

**An Improved Arc Algorithm for Detecting  
Definite Hermitian Pairs**

Chun-Hua Guo, Nicholas J. Higham and  
Françoise Tisseur

November 2008

MIMS EPrint: **2008.115**

Manchester Institute for Mathematical Sciences  
School of Mathematics

The University of Manchester

Reports available from: <http://www.manchester.ac.uk/mims/eprints>

And by contacting: The MIMS Secretary  
School of Mathematics  
The University of Manchester  
Manchester, M13 9PL, UK

ISSN 1749-9097

# AN IMPROVED ARC ALGORITHM FOR DETECTING DEFINITE HERMITIAN PAIRS\*

CHUN-HUA GUO<sup>†</sup>, NICHOLAS J. HIGHAM<sup>‡</sup>, AND FRANÇOISE TISSEUR<sup>‡</sup>

**Abstract.** A 25-year old and somewhat neglected algorithm of Crawford and Moon attempts to determine whether a given Hermitian matrix pair  $(A, B)$  is definite by exploring the range of the function  $f(x) = x^*(A + iB)x/|x^*(A + iB)x|$ , which is a subset of the unit circle. We revisit the algorithm and show that with suitable modifications and careful attention to implementation details it provides a reliable and efficient means of testing definiteness. A clearer derivation of the basic algorithm is given that emphasizes an arc expansion viewpoint and makes no assumptions about the definiteness of the pair. Convergence of the algorithm is proved for all  $(A, B)$ , definite or not. It is shown that proper handling of three details of the algorithm is crucial to the efficiency and reliability: how the midpoint of an arc is computed, whether shrinkage of an arc is permitted, and how directions of negative curvature are computed. For the latter, several variants of Cholesky factorization with complete pivoting are explored and the benefits of pivoting demonstrated. The overall cost of our improved algorithm is typically just a few Cholesky factorizations. Applications of the algorithm are described to testing the hyperbolicity of a Hermitian quadratic matrix polynomial, constructing conjugate gradient methods for sparse linear systems in saddle point form, and computing the Crawford number of the pair  $(A, B)$  via a quasiconvex univariate minimization problem.

**Key words.** definite pair, pencil, Hermitian generalized eigenvalue problem, direction of negative curvature, Crawford number, hyperbolic quadratic eigenvalue problem, saddle point linear system

**AMS subject classifications.** 15A18, 65F15, 65F30

**1. Introduction.** For given Hermitian matrices  $A, B \in \mathbb{C}^{n \times n}$ , the pair  $(A, B)$  is said to be a definite pair if  $x^*(A + iB)x \neq 0$  for all nonzero  $x \in \mathbb{C}^n$ , and otherwise indefinite. It is known that  $(A, B)$  is definite if and only if there exists a real number  $t$  such that the matrix

$$(1.1) \quad B(t) = A \sin t + B \cos t$$

is positive definite [1], [25]. If  $(A, B)$  is definite then both theoretical and computational advantages accrue. In the associated generalized eigenvalue problem  $Ax = \lambda Bx$  the eigenvalues are real,  $A$  and  $B$  are simultaneously diagonalizable, and if  $t$  is known then the eigenvalues can be computed via those of an associated pair  $(A(t), B(t))$ , where

$$(1.2) \quad A(t) + iB(t) \equiv A \cos t - B \sin t + i(A \sin t + B \cos t) = e^{it}(A + iB),$$

by methods that exploit the definiteness of  $B(t)$  [9].

Interest in definite pairs also arises in contexts other than the generalized eigenvalue problem. For an  $n \times n$  Hermitian quadratic matrix polynomial with positive

---

\*Version of November 28, 2008. This work was supported by a Royal Society-Wolfson Research Merit Award to the second author.

<sup>†</sup>Department of Mathematics and Statistics, University of Regina, Regina, SK S4S 0A2, Canada (chguo@math.uregina.ca, <http://www.math.uregina.ca/~chguo/>). The research of this author was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

<sup>‡</sup>School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK (higham@ma.man.ac.uk, <http://www.ma.man.ac.uk/~higham/>, ftisseur@ma.man.ac.uk, <http://www.ma.man.ac.uk/~ftisseur/>). The work of both authors was supported by Engineering and Physical Sciences Research Council grant EP/D079403.

definite leading coefficient matrix, hyperbolicity is equivalent to definiteness of a related  $2n \times 2n$  Hermitian matrix pair, so testing for hyperbolicity of the quadratic can be done by testing for definiteness of the pair. The question of definiteness of a pair also arises with saddle point linear systems, for which the coefficient matrix is symmetric but indefinite. Liesen and Parlett [18] show that if a certain pair of symmetric matrices is definite then there exists a well-defined conjugate gradient iteration for the saddle point system.

Unfortunately, determining whether a given pair is definite is not straightforward. What is required is an efficient algorithm that takes as input a Hermitian pair  $(A, B)$  and finds a real number  $t$  such that  $B(t)$  is positive definite or determines that no such  $t$  exists. To the best of our knowledge, the first such algorithm was suggested by Moon [21], and this algorithm was later refined by Crawford and Moon [7]. The algorithm of Crawford and Moon has received relatively little attention in the literature, perhaps due to a lack of clarity in the derivation and statement of the algorithm and in the explanation of its properties. In particular, the presentations in [6] and [7] can lead the reader to interpret the algorithm as a linearly convergent bisection procedure that potentially requires  $n$  iterations each of  $O(n^3)$  flops cost.

In this work we re-examine the algorithm of Crawford and Moon, emphasizing the mechanism by which it performs arc expansion on the unit circle until the (in)definiteness of the pair  $(A, B)$  is determined. We give a clearer derivation that makes no assumptions about the definiteness of the pair and formulate an improved algorithm that is better suited to floating point arithmetic. We prove convergence of the algorithm for all  $(A, B)$ , definite or not, and justify the algorithm's determinations using backward error analysis. We compute the midpoint of an arc in a more numerically reliable way and allow the arc to shrink to compensate for the effects of rounding errors. We also improve on the Cholesky-based implementation proposed in [7] and realized in [6] of the computation of directions of negative curvature. We show that the suitable use of complete pivoting, possibly with a different choice of negative curvature direction, leads to faster convergence of the algorithm. Overall, our modifications improve the efficiency of the Crawford–Moon algorithm—often at least halving the number of iterations—and greatly improve the reliability of the algorithm in floating point arithmetic.

The outline of the paper is as follows. The modified algorithm is derived and analyzed in Section 2, where a description of our improvements is given along with a justification of the algorithm via backward error analysis. How to test definiteness of a matrix and compute directions of negative curvature is considered in Section 3. Section 4 treats the application of the algorithm to hyperbolic quadratic matrix polynomials and sparse linear systems in saddle point form. In this section it is also shown that once the algorithm has determined that a pair is definite the Crawford number (which measures how far a definite pair is from the nearest indefinite one [16]) can be computed via a quasiconvex univariate minimization problem. Detailed numerical experiments on a variety of problems are given in Section 5, followed by conclusions in Section 6.

**2. A modification of the Crawford–Moon algorithm.** We first derive a modified version of the Crawford–Moon algorithm and then explain how it differs from the original algorithm. For a Hermitian pair  $(A, B)$  with  $A$  and  $B$  not both zero we define

$$(2.1) \quad f(x) = \frac{x^*(A + iB)x}{|x^*(A + iB)x|}, \quad x \in \mathbb{C}^n, \quad x^*(A + iB)x \neq 0.$$

Thus  $f(x)$  lies on the unit circle. The range of  $f$  is described in the following result.

LEMMA 2.1 ([1], [2]). *The range of  $f$  is of one of the following types:*

- (i) *A closed arc on the unit circle of length less than  $\pi$ .*
- (ii) *Two diametrically opposite points on the unit circle.*
- (iii) *The whole unit circle.*
- (iv) *A half circle with or without one or both endpoints.*

*If  $(A, B)$  is definite then (i) is the only possibility, while if  $(A, B)$  is indefinite any of (i)–(iv) can hold.*

For any complex numbers  $a$  and  $b$  on the unit circle such that  $a \neq -b$ , we denote by  $\text{arc}[a, b]$  the shorter arc on the unit circle connecting  $a$  and  $b$ . The length of the arc, or equivalently the angle subtended by the arc at the origin, is denoted by  $\theta[a, b]$ . When  $a = -b$ , we define  $\theta[a, b] = \pi$  and  $\text{arc}[a, b]$  can be understood as either of the two arcs connecting  $a$  and  $b$ . For a definite pair we will denote the range of  $f$  in (2.1) by  $\text{arc}[\tilde{a}, \tilde{b}]$ , where  $\theta[\tilde{a}, \tilde{b}] < \pi$  by Lemma 2.1.

In [6] and [7] the Crawford–Moon algorithm is viewed as a bisection-like search for the midpoint of  $\text{arc}[\tilde{a}, \tilde{b}]$ , assuming  $(A, B)$  is definite. This viewpoint is not conducive to understanding the algorithm’s behavior for indefinite pairs  $(A, B)$ , and indeed [6] and [7] give an incomplete description of what happens for such pairs. We take a different viewpoint in deriving and analyzing our modified algorithm.

We start with a modified statement of [7, Lem. 2.5]. Recall that  $B(t)$  is defined in (1.1).

LEMMA 2.2. *Let  $c = \sin t + i \cos t$ . Assume that  $B(t)$  is not positive definite and take  $x \neq 0$  such that  $x^* B(t)x \leq 0$ . If  $x^*(A + iB)x \neq 0$  then for  $d = f(x)$  we have  $\theta[c, d] \geq \pi/2$ .*

*Proof.* It suffices to note that the inner product of the two nonzero vectors  $\hat{d} = [x^* Ax, x^* Bx]^T$ ,  $\hat{c} = [\sin t, \cos t]^T \in \mathbb{R}^2$  is

$$\hat{d}^T \hat{c} = (x^* Ax) \sin t + (x^* Bx) \cos t = x^* B(t)x \leq 0. \quad \square$$

In the statement and proof of Lemma 2.5 in [7] it is implicitly assumed that the pair  $(A, B)$  is definite. The definiteness is not needed in our revised lemma.

We can now explain the algorithm. To determine the definiteness of the pair we start with any point  $c = \sin t + i \cos t$  in the range of  $f$ . If the conditions of Lemma 2.2 do not both hold then the (in)definiteness of  $(A, B)$  is determined immediately. Otherwise, Lemma 2.2 produces another point  $d$  on the unit circle such that  $\theta[c, d] \geq \pi/2$ . Let  $a = c$  and  $b = d$ . If  $b = -a$ , then the pair  $(A, B)$  is indefinite by Lemma 2.1. Otherwise,  $\text{arc}[a, b]$  must be part of the range of  $f$ , by Lemma 2.1 again. We then apply Lemma 2.2 with  $c = \sin t + i \cos t$  the midpoint of  $\text{arc}[a, b]$ . Unless a determination of definiteness is made at this time, a point  $d$  outside  $\text{arc}[a, b]$  is found, and the arc is extended (in the range of  $f$ ) by replacing the closer endpoint by  $d$ . If the length of the new arc is  $\pi$  or greater, the pair  $(A, B)$  is indefinite by Lemma 2.1. Otherwise, the above process can be repeated on the new arc.

As the experiments in Section 5 will show, it is often the case that a positive definite  $B(t)$ , or an arc exceeding  $\pi$  in length, is produced after just a few iterations.

The reasoning above leads to the following modified version of the algorithm of Crawford and Moon. Figure 2.1 illustrates the main loop of the algorithm. We denote by  $e_i$  the  $i$ th column of the identity matrix, and  $u$  denotes the unit roundoff.

ALGORITHM 2.3. *Given Hermitian  $A, B \in \mathbb{C}^{n \times n}$  and a convergence tolerance  $\text{tol}$  this algorithm determines whether the pair  $(A, B)$  is definite and, if it is, returns  $t \in \mathbb{R}$  such that  $B(t)$  in (1.1) is positive definite.*

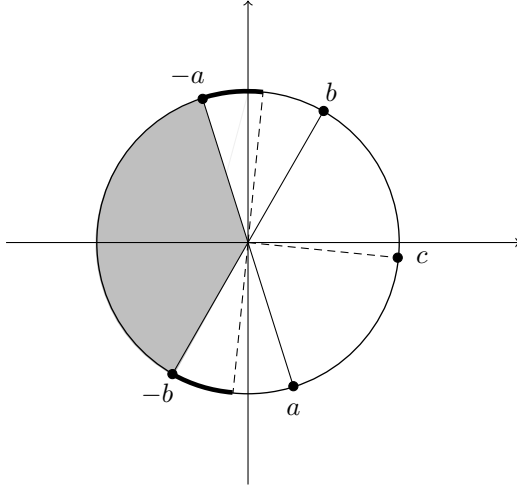


FIG. 2.1. The figure depicts the unit circle in the complex plane and the situation during the main loop of Algorithm 2.3. The point  $c = \sin t + i \cos t$  is the midpoint of  $\text{arc}[a, b]$ . If  $B(t)$  is positive definite then  $(A, B)$  is definite. Otherwise, if indefiniteness is not detected on line 17 then Lemma 2.2 implies that  $f(x)$  lies at an angle at least  $\pi/2$  from  $c$ . If  $f(x)$  is located in the arc corresponding to the gray shaded sector then  $(A, B)$  is not definite, by Lemma 2.1. If  $f(x)$  lies on one of the arcs shown as thick black lines then the pair is possibly definite and the algorithm continues with  $a$  or  $b$  replaced by  $f(x)$ .

```

% Starting phase
1 Take  $x$  with  $\|x\|_2 = 1$ , say  $x = e_1$ .
2 if  $x^*(A + iB)x = 0$ , quit (pair is indefinite), end
3  $a = f(x) = \sin t + i \cos t$ .
4 if  $B(t)$  is positive definite, quit (pair is definite), end
5 Find  $x$  with  $\|x\|_2 = 1$  such that  $x^*B(t)x \leq 0$ .
6 if  $x^*(A + iB)x = 0$ , quit (pair is indefinite), end
7  $b = f(x)$  % We have  $\theta[a, b] \geq \pi/2$  by Lemma 2.2.
8 if  $b = -a$ 
9   quit (pair is indefinite) % By Lemma 2.1.
10 else
11    $\theta = \theta[a, b]$ , and interchange  $a$  and  $b$  if necessary so that  $b = ae^{i\theta}$ .
12 end
13 if  $\theta \geq \pi - \text{tol}$ , quit (pair is within distance tol of an indefinite pair), end
% Main loop
14  $c = ae^{i\theta/2} = \sin t + i \cos t$  %  $c$  is the midpoint of  $\text{arc}[a, b]$ .
15 if  $B(t)$  is positive definite, quit (pair is definite), end
16 Find  $x$  with  $\|x\|_2 = 1$  such that  $x^*B(t)x \leq 0$ .
17 if  $x^*(A + iB)x = 0$ , quit (pair is indefinite), end
18  $d = f(x)$  % We have  $\theta[c, d] \geq \pi/2$  by Lemma 2.2.
19  $\theta = \theta/2 + \theta[c, d]$ 
20 if  $\theta \geq \pi - \text{tol}$ 
21   quit % (pair is indefinite if  $\theta \geq \pi$  by Lemma 2.1,
      or otherwise within distance tol of an indefinite pair)
22 elseif  $\theta[a, d] < \theta[b, d]$ 
23    $a = d$ , goto line 14

```

```

24 else
25    $b = d$ , goto line 14
26 end

```

The changes we have made to the original algorithm of Crawford and Moon, as presented in [7], include the following.

1. The original algorithm starts by finding  $a \neq b$  in  $\text{arc}[\tilde{a}, \tilde{b}]$ . Thus there is an assumption that the pair is definite, so that  $\tilde{a}$  and  $\tilde{b}$  are defined. But even if the pair is definite it may not be possible to find  $a \neq b$  in  $\text{arc}[\tilde{a}, \tilde{b}]$ . A simple example is  $A = B = I$ , for which the arc is a single point. Our revised algorithm makes no assumptions on the definiteness of the pair and correctly determines (in the starting phase) that the pair  $(I, I)$  is definite. The code attached to [6] does not assume the definiteness of  $(A, B)$  to start with. However, it first generates a point on the unit circle that is not necessarily in the range of  $f$ . In our revised algorithm we start with the generation of a point on the unit circle that is in the range of  $f$ , which is immediately used in the expansion of an arc in the range of  $f$ .

2. When a nonzero vector  $x$  is generated (as on our lines 5 and 16), the original algorithm proceeds with the computation of  $f(x)$ . Here again the pencil is implicitly assumed to be definite so that  $f(x)$  is defined. Our revised algorithm tests for indefiniteness by checking whether  $x^*(A + iB)x = 0$ .

3. We terminate on line 13 or 21 when  $\theta[a, b]$  is at least  $\pi - \text{tol}$ . As shown later in this section this implies the pair is within distance  $\text{tol}$  of an indefinite pair and so, assuming  $\text{tol}$  is approximately the maximum of the unit roundoff and the uncertainty in the data, iterating further will not provide any more useful information.

4. The original algorithm computes the midpoint  $c$  of  $\text{arc}[a, b]$  as  $c = (a+b)/|a+b|$ . However, as  $a$  approaches  $-b$  this formula loses more and more significant digits, thereby bringing existing errors in  $a$  and  $b$  into prominence, and when  $a = -b$  the formula breaks down. The solution is to work with the arguments of the complex numbers. Algorithm 2.3 does this and obtains  $c$  by rotating  $a$  or  $b$  through  $\theta[a, b]/2$ .

5. The way we update  $\text{arc}[a, b]$  on lines 22–25 is equivalent to that in [7] in exact arithmetic. In floating point arithmetic, however, the algorithm in [7] declares failure whenever  $\text{arc}[a, b]$  is not expanded (due to rounding errors). In our algorithm we allow  $\text{arc}[a, b]$  to shrink now and expand later. We return to this point at the end of Section 3.

6. We have left open how to implement lines 4, 5, 15, and 16 and consider this in detail in Section 3.

While Crawford and Moon state that there is no guarantee of convergence (in exact arithmetic) when  $(A, B)$  is indefinite, it is easy to show that the sequence  $\text{arc}[a_k, b_k]$ , where  $a_k$  and  $b_k$  are generated after the  $k$ th test of the positive definiteness of  $B(t)$ , is always convergent in Algorithm 2.3. We need the following lemma.

LEMMA 2.4. *On the  $k$ th step of Algorithm 2.3,  $\theta[a_k, b_k] \geq \pi(1 - 2^{-k})$ , regardless of whether the pair  $(A, B)$  is definite or not.*

*Proof.* In the first step we have  $\theta[a_1, b_1] \geq \pi/2$ . By Lemma 2.2 and the procedure in the algorithm, we know that

$$\begin{aligned}
\theta[a_k, b_k] &\geq \frac{\pi}{2} + \frac{1}{2}\theta[a_{k-1}, b_{k-1}] \geq \dots \\
&\geq \frac{\pi}{2} \left( 1 + \frac{1}{2} + \dots + \left(\frac{1}{2}\right)^{k-2} \right) + \left(\frac{1}{2}\right)^{k-1} \theta[a_1, b_1] \\
&\geq \frac{\pi}{2} \left( 1 + \frac{1}{2} + \dots + \left(\frac{1}{2}\right)^{k-1} \right) = \pi(1 - 2^{-k}). \quad \square
\end{aligned}$$

The convergence of  $\text{arc}[a_k, b_k]$  can now be seen as follows. Unless the algorithm converges in a finite number of steps, we have  $\text{arc}[a_k, b_k] \in [\pi(1 - 2^{-k}), \pi]$  for all  $k$ . But in this case the sequence is clearly convergent to  $\pi$  and the pair is not definite.

To explore further the implications of Lemma 2.4, consider the two possible cases for  $(A, B)$ , first that it is not definite. If the range of  $f$  is of type (ii) in Lemma 2.1, the indefiniteness of the pair is detected in the starting phase (on the first step) of Algorithm 2.3. If a determination is not made on the  $k$ th step of Algorithm 2.3 then the range of  $f$  must be an arc on the unit circle of length  $J \geq \theta[a_k, b_k] \geq \pi(1 - 2^{-k})$ , by Lemma 2.4, regardless of whether the pair  $(A, B)$  is definite or not. Thus the statement at the end of Section 3 in [7] that no information is gained in the indefinite case is incorrect. In the second case, where  $(A, B)$  is definite,  $\tilde{a}$  and  $\tilde{b}$  are defined and

$$\pi > \theta[\tilde{a}, \tilde{b}] \geq \theta[a_k, b_k] \geq \pi(1 - 2^{-k}).$$

So if we need to carry out  $k$  steps of the algorithm, we must have

$$\pi - \theta[\tilde{a}, \tilde{b}] \leq 2^{-k}\pi.$$

This is essentially [7, Thm. 3.2].

The possibility of Algorithm 2.3 failing to terminate cannot be ruled out in floating point arithmetic. But whenever a determination is made it is numerically stable in the sense that it is correct for a perturbed pair  $(A + \Delta A, B + \Delta B)$  with

$$(2.2) \quad \|\begin{bmatrix} \Delta A & \Delta B \end{bmatrix}\|_2 \leq c_n u \|\begin{bmatrix} A & B \end{bmatrix}\|_2,$$

where  $c_n$  is a modest constant. We now justify this claim. Suppose the pair is determined to be definite on line 4 or 15 and that positive definiteness is checked in a numerically stable way, that is, the correct answer is returned for  $B(t) + E$  with  $\|E\|_2 \leq c_n u \|B(t)\|_2$ . We can write

$$B(t) + E = (A + E \sin t) \sin t + (B + E \cos t) \cos t =: (A + \Delta A) \sin t + (B + \Delta B) \cos t,$$

where

$$\begin{aligned} \|\begin{bmatrix} \Delta A & \Delta B \end{bmatrix}\|_2 &= \|E \begin{bmatrix} \sin t I & \cos t I \end{bmatrix}\|_2 = \|E\|_2 \leq c_n u \|B(t)\|_2 \\ &= c_n u \left\| \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} \sin t I \\ \cos t I \end{bmatrix} \right\|_2 \leq c_n u \|\begin{bmatrix} A & B \end{bmatrix}\|_2. \end{aligned}$$

Thus the test is numerically stable for the pair. If the pair is determined to be indefinite on line 2, 6, or 17 then standard error analysis for complex arithmetic [14, Chap. 3] can be used to show that  $(A + \Delta A, B + \Delta B)$  is indefinite for a perturbation  $(\Delta A, \Delta B)$  satisfying (2.2). If the pair is deemed to be indefinite or within a distance  $\text{tol}$  of an indefinite pair on line 13 or 21 we justify the decision as follows. Suppose the pair is actually definite. Then we must have  $\theta[\tilde{a}, \tilde{b}] \geq \pi - \max(c'_n u, \text{tol})$ , since the computed  $a_k$  and  $b_k$  are points on the unit circle (up to machine precision) and  $\theta[a_k, b_k]$  can be computed accurately for the computed  $a_k$  and  $b_k$ . We now need to explain why  $\theta[\tilde{a}, \tilde{b}]$  close to  $\pi$  implies nearness of the definite pair  $(A, B)$  to an indefinite pair.

Higham, Tisseur, and Van Dooren [16, Thm. 2.2] show that the distance

$$(2.3) \quad d(A, B) = \min \left\{ \|\begin{bmatrix} \Delta A & \Delta B \end{bmatrix}\|_2 : z^*(A + \Delta A + i(B + \Delta B))z = 0, \text{ some } z \neq 0 \right\}$$

from a Hermitian pair (definite or indefinite) to the nearest indefinite pair is equal to the Crawford number

$$(2.4) \quad \gamma(A, B) = \min_{\substack{z \in \mathbb{C}^n \\ \|z\|_2=1}} |z^*(A + iB)z|,$$

which is the distance from the origin to the field of values  $F(A + iB) = \{z^*(A + iB)z : z \in \mathbb{C}^n, \|z\|_2 = 1\}$ . Suppose that  $(A, B)$  is definite with  $\theta[\tilde{a}, \tilde{b}] = \pi - \delta > 0$  and that

$$\tilde{a} = \frac{z_1^*(A + iB)z_1}{|z_1^*(A + iB)z_1|}, \quad \tilde{b} = \frac{z_2^*(A + iB)z_2}{|z_2^*(A + iB)z_2|}$$

with  $\|z_1\|_2 = \|z_2\|_2 = 1$  and  $z_1 \neq z_2$ . From the convexity of the field of values the line connecting  $|z_1^*(A + iB)z_1|\tilde{a}$  and  $|z_2^*(A + iB)z_2|\tilde{b}$  is in  $F(A + iB)$ . Since  $|z_k^*(A + iB)z_k| \leq \sqrt{\|A\|_2^2 + \|B\|_2^2} =: \alpha$ ,  $\gamma(A, B)$  is at most the distance from the origin to the line connecting  $\alpha\tilde{a}$  and  $\alpha\tilde{b}$ . The latter distance is  $\alpha \sin(\delta/2)$ . Thus

$$(2.5) \quad \begin{aligned} \gamma(A, B) &\leq \alpha \sin(\delta/2) \leq \frac{1}{2}\alpha\delta = \frac{1}{2}\sqrt{\|A\|_2^2 + \|B\|_2^2} (\pi - \theta[\tilde{a}, \tilde{b}]) \\ &\leq 2^{-1/2} \| [A \quad B] \|_2 (\pi - \theta[\tilde{a}, \tilde{b}]), \end{aligned}$$

and so the relative distance from  $(A, B)$  to the nearest indefinite pair is bounded by  $2^{-1/2}(\pi - \theta[\tilde{a}, \tilde{b}])$ . We conclude that the algorithm can wrongly diagnose indefiniteness only when  $(A, B)$  is close in a normwise relative sense to an indefinite pair.

In the next section we consider how to implement the definiteness tests and the computation of the vectors  $x$ , and we introduce further improvements over [6] and [7].

**3. Testing definiteness and computing a direction of negative curvature.** Critical to Algorithm 2.3 is the ability to test whether  $C \equiv B(t)$  is positive definite and, if it is not positive definite, to compute a nonzero vector  $x$  such that  $x^*Cx \leq 0$ . This task arises commonly in Newton methods for optimization, with  $C$  a Hessian matrix, and in this context such an  $x$  is called a direction of negative curvature<sup>1</sup> of the underlying function [10], [22].

A vector  $x$  that gives the greatest arc expansion in the current step is one for which

$$(3.1) \quad [\sin t, \cos t] \begin{bmatrix} x^*Ax \\ x^*Bx \end{bmatrix} / d(x) = x^*Cx/d(x)$$

is minimized, where  $d(x) = ((x^*Ax)^2 + (x^*Bx)^2)^{1/2}$  satisfies  $\gamma(A, B) \leq d(x)/\|x\|_2 \leq (\|A\|_2^2 + \|B\|_2^2)^{1/2}$ . However, this minimization problem is too expensive to solve in the context of our problem and so we will pursue the less ambitious aim of developing an inexpensive procedure to find a unit vector  $x$  such that  $x^*Cx$  is small.

The idea we explore is to run the Cholesky algorithm until it either completes, in which case  $C$  is declared positive definite, or it encounters a negative or zero pivot, in which case  $C$  is declared not positive definite. It can be shown that this test is numerically stable, in the sense described in the previous section [13]. When non-definiteness is declared the partial factorization is used to compute a direction of negative curvature.

<sup>1</sup>Strictly speaking we should write ‘‘nonpositive curvature’’, since we are allowing  $x^*Cx = 0$ .

Suppose that  $k$  steps of the outer product form of Cholesky factorization have been successfully completed, optionally with symmetric pivoting. Then we have

$$(3.2) \quad P^T C P = \begin{bmatrix} R_{11}^* \\ R_{12}^* \end{bmatrix} \begin{matrix} k & n-k \\ [R_{11} & R_{12}] \end{matrix} + \begin{matrix} k & n-k \\ n-k & \\ 0 & S_k \end{matrix} \begin{bmatrix} 0 & 0 \\ 0 & S_k \end{bmatrix},$$

where  $P$  is a permutation matrix,  $R_{11}$  is upper triangular with real, positive diagonal elements, and  $S_k$  is a Schur complement. Suppose also that  $P$  incorporates interchanges for the next step and that the (1,1) element  $s_{11}^{(k)}$  of  $S_k$  is nonpositive, so that the factorization breaks down. Then a vector  $x$  such that  $x^* C x = s_{11}^{(k)} \leq 0$  can be constructed as

$$(3.3) \quad x = P Z e_1, \quad Z = \begin{bmatrix} R_{11}^{-1} R_{12} \\ -I \end{bmatrix}.$$

This is a well-known technique (see, e.g., [14, Prob. 10.9]), and it is used by Crawford and Moon with  $P = I$ .

Two nontrivial choices of  $P$  are of interest. First, complete pivoting, in which at each stage the pivot is chosen as the largest diagonal element in the active part of the matrix. With this choice the factorization continues as long as possible. When the factorization terminates early because no positive pivot is available we bring the smallest diagonal element into the pivot position prior to computing  $x$  from (3.3). The second choice is a modified form of complete pivoting that we will call early-exit complete pivoting. This is the same as complete pivoting except that if there are any nonpositive diagonal elements in the active part of the matrix then the smallest one is brought to the pivot position; thus the factorization is terminated as soon as the diagonal reveals that the matrix is not positive definite.

Note that  $x$  in (3.3) is built solely from information on the diagonal of  $S_k$ . Another choice of  $x$  is possible that incorporates information from the off-diagonal elements. For notational simplicity write  $S_k \equiv S = (s_{ij}) \in \mathbb{C}^{(n-k) \times (n-k)}$ . Define

$$(3.4a) \quad y = \begin{cases} e_p, & p = q, \\ \frac{1}{\sqrt{2}}(e_p - \text{sign}(s_{pq})e_q), & p \neq q, \end{cases}$$

$$(3.4b) \quad |s_{pq}| := \max\{|s_{ij}| : i \neq j, \text{ or } i = j \text{ and } s_{ii} \leq 0\},$$

where  $p = q$  is chosen if the maximum is attained for both a diagonal element and an off-diagonal element. That  $y$  is a direction of negative curvature when  $p \neq q$  follows from  $y^* S y = \frac{1}{2}(s_{pp} + s_{qq} - 2|s_{pq}|) \leq 0$ . This choice is used by Forsgren, Gill, and Murray [10] in conjunction with Cholesky factorization with complete pivoting in modified Newton methods. The following lemma shows that when  $s_{ii} \leq 0$  for all  $i$ , as is the case when complete pivoting terminates,  $y$  is a nearly optimal direction of negative curvature for  $S$ , in the sense that it nearly achieves  $\min\{z^* S z : z^* z = 1\} = \lambda_{\min}(S)$ .

LEMMA 3.1. *If  $\lambda_{\min}(S) \leq 0$  and  $s_{ii} \leq 0$  for all  $i$  then  $y$  in (3.4) satisfies  $y^* S y \leq \lambda_{\min}(S)/(n-k)$ .*

*Proof.* If  $p = q$  in (3.4) then  $y^* S y = s_{pp} = -\max_{i,j} |s_{ij}|$ , and otherwise,  $y^* S y = \frac{1}{2}(s_{pp} + s_{qq} - 2|s_{pq}|) \leq -|s_{pq}| = -\max_{i,j} |s_{ij}|$ . But  $\max(-\lambda_{\min}(S), \lambda_{\max}(S)) = \|S\|_2 \leq (n-k) \max_{i,j} |s_{ij}|$ , so  $-\max_{i,j} |s_{ij}| \leq \lambda_{\min}(S)/(n-k)$ .  $\square$

The actual direction of negative curvature is

$$(3.5) \quad x = P Z y,$$

where  $Z$  is given in (3.3), and so unfortunately the bound of the lemma worsens when translated for  $x$  if  $Z$  has large norm.

The potential benefit of (3.5) over (3.3) is easy to see. For example if  $C = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$  then (3.3) returns zero curvature whereas (3.5) yields (optimal) curvature of  $-1$ .

Two aspects need to be considered when choosing from among the options above. First is the effect the choice of  $x$  has on the number of iterations required by Algorithm 2.3. As noted above we would really like to minimize (3.1), and it is hard to predict which of the above options comes closest to achieving this aim; the numerical experiments in Section 5 shed light on this. The second aspect is the effect of rounding errors—and this is subtle. The algorithm aims to expand the arc on each iteration and it guarantees to achieve this if, for the computed quantities,  $[\sin t, \cos t][x^*Ax, x^*Bx]^T/d(x) = x^*Cx/d(x) \leq 0$ . Although the above choices of  $x$  ensure this inequality holds in exact arithmetic, in practice rounding errors in computing  $x$  could vitiate the inequality. We will consider the effects such an event may have on the algorithm shortly, but first we investigate what can be said about the accuracy of  $x$ .

We begin by noting that  $R_{11}$  is the Cholesky factor of the positive definite matrix  $\tilde{C}_{11} = (P^T C P)_{11}$ , and while the eigenvalues of  $\tilde{C}_{11}$  interlace those of  $C$ , this does not constrain the conditioning of  $\tilde{C}_{11}$  when  $C$  is indefinite, even if pivoting is used. Hence  $R_{11}$  can be arbitrarily ill conditioned, while  $R_{12}$  is arbitrary. In computing  $x$  in (3.3) or (3.5) we need to solve a triangular system of the form  $R_{11}w = b$ . We know from the rounding error analysis for triangular systems [14, Sec. 8.2] that the computed  $\hat{w}$  satisfies

$$(3.6) \quad \frac{\|w - \hat{w}\|_\infty}{\|w\|_\infty} \leq \frac{\text{cond}(R_{11})\gamma_n}{1 - \text{cond}(R_{11})\gamma_n}, \quad \text{cond}(R_{11}) = \| |R_{11}^{-1}| |R_{11}| \|_\infty,$$

where  $\gamma_n = nu/(1 - nu)$ . For a general triangular matrix  $U$ ,  $\text{cond}(U)$  is unbounded. However, for  $R_{11} = (r_{ij})$  from complete pivoting it holds that [14, Sec. 10.3]

$$(3.7) \quad r_{ii}^2 \geq \sum_{\ell=i}^{\min(j,k)} r_{\ell j}^2, \quad j = i+1:k, \quad i = 1:k$$

which implies [14, Lem. 8.6] that  $\text{cond}(R_{11}) \leq 2^k - 1$ , and this bound is approximately attainable [14, Sec. 8.3]. Hence, for complete pivoting the error is small if  $k$  is small, but nevertheless can still potentially be large unless  $n$  is small. However, in practice triangular systems tend to be solved to higher accuracy than the bounds predict [14, p. 140].

An alternative approach for computing  $x$  is to use the eigensystem of  $C$ . We first unitarily reduce  $C$  to Hermitian tridiagonal form:  $Q^*CQ = T$ , which requires  $4n^3/3$  flops. Then we compute the smallest eigenvalue  $\lambda_n$  and its corresponding unit 2-norm eigenvector  $x$  by the bisection method and inverse iteration applied to  $T$ , which costs just  $O(n^2)$  flops. If  $\lambda_n > 0$  then  $C$  is positive definite, otherwise  $x$  is a direction of negative curvature with  $x^*Cx = \lambda_n \leq 0$ . This approach is backward stable [17]. However, Cholesky factorization costs only  $n^3/3$  flops, and in practice the Cholesky factorization has a flop count advantage of more than a factor 4 because for non-positive definite matrices Cholesky may break down well before  $n$  steps have been completed.

We now return to the effect of an inaccurate  $x$  on Algorithm 2.3. As already noted, the main danger is that the arc does not expand, or indeed shrinks, and this

is most likely when  $\text{arc}[a_k, b_k]$  is close to  $\pi$ . In this circumstance we can regard the algorithm as restarting with a different arc. In our experience (see Section 5) non-expansion of the arc is rare, and when it happens the algorithm recovers if allowed to continue; therefore we do not test for it in Algorithm 2.3.

**4. Applications.** In this section we discuss the application of Algorithm 2.3 to three quite different problems.

**4.1. Testing hyperbolicity of Hermitian quadratics.** A Hermitian quadratic matrix polynomial  $Q(\lambda) = \lambda^2 A + \lambda B + C$ , where  $A, B, C \in \mathbb{C}^{n \times n}$  are Hermitian and  $A$  is positive definite is said to be hyperbolic if [11], [15]

$$(x^* B x)^2 > 4(x^* A x)(x^* C x) \text{ for all nonzero } x \in \mathbb{C}^n.$$

It is well known that  $Q$  is hyperbolic if and only if  $Q(\mu) < 0$  for some  $\mu \in \mathbb{R}$  [20, Lem. 31.15]. It is also known [16, Thm. 3.6] that  $Q$  is hyperbolic if and only if the pair  $(A_1, B_1)$  is definite, where

$$(4.1) \quad A_1 = \begin{bmatrix} -C & 0 \\ 0 & A \end{bmatrix}, \quad B_1 = - \begin{bmatrix} B & A \\ A & 0 \end{bmatrix}.$$

The equivalence of these two characterizations of hyperbolicity is seen through the congruence

$$(4.2) \quad \alpha A_1 + \beta B_1 = \begin{bmatrix} -\alpha C - \beta B & -\beta A \\ -\beta A & \alpha A \end{bmatrix}$$

$$(4.3) \quad = \begin{bmatrix} I & -(\beta/\alpha)I \\ 0 & I \end{bmatrix} \begin{bmatrix} -\alpha Q(\beta/\alpha) & 0 \\ 0 & \alpha A \end{bmatrix} \begin{bmatrix} I & 0 \\ -(\beta/\alpha)I & I \end{bmatrix}.$$

Algorithm 2.3 can be used to test the hyperbolicity of  $Q$  and, in the case of a positive test result, compute  $\mu \in \mathbb{R}$  so that  $Q(\mu) < 0$ . The key part of the algorithm is to determine whether  $B(t) \equiv \alpha A_1 + \beta B_1$  is positive definite, where  $\alpha = \sin t$  and  $\beta = \cos t$ , and, when  $B(t)$  is not positive definite, find a nonzero vector  $x$  such that  $x^* B(t) x \leq 0$ . For efficiency, we would prefer to carry out this task by working only with  $n \times n$  matrices, instead of working directly on the  $2n \times 2n$  matrix  $B(t)$ ; how to do so was mentioned briefly by us in [11] and we now give more details.

The structure of  $A_1$  and  $B_1$  implies that if  $\alpha \leq 0$  then  $B(t)$  is not positive definite, and in this case  $e_j$  is a direction of negative curvature for  $j = n + 1:2n$ . We can therefore assume that  $\alpha > 0$  and in view of (4.3) the question is then whether  $-Q(\beta/\alpha)$  is positive definite. Suppose the answer is “no” and that a direction of negative curvature  $y$  for  $-Q(\beta/\alpha)$  is obtained. The corresponding direction of negative curvature for  $B(t)$  is

$$x = \begin{bmatrix} I & 0 \\ -(\beta/\alpha)I & I \end{bmatrix}^{-1} \begin{bmatrix} y \\ 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ (\beta/\alpha)I & I \end{bmatrix} \begin{bmatrix} y \\ 0 \end{bmatrix} = \begin{bmatrix} y \\ (\beta/\alpha)y \end{bmatrix}.$$

The algorithm then computes

$$v = x^*(A_1 + iB_1)x = y^*(-C + (\beta/\alpha)^2 A)y + iy^*(-B - 2(\beta/\alpha)A)y$$

and thence  $f(x) = v/|v|$ . If  $|\beta/\alpha| \gg 1$  then  $v$  is dominated by  $A$  and the contributions from  $B$  and  $C$  are small. Yet it is clear from (4.2) that the contribution from  $B$  is still important when  $|\beta/\alpha| \gg 1$ . Hence this approach could have numerical instability

problems through loss of significance in floating point arithmetic. The underlying reason is the use of a potentially ill conditioned congruence transformation. Note that the congruence can be avoided by working with the  $2n \times 2n$  pair  $(A_1, B_1)$ , but at a significant increase in cost. However the chance that  $\beta/\alpha$  is very large is relatively small in practice. We will return to the behavior of the congruence approach in Experiment 5 of Section 5.

**4.2. Linear systems in saddle point form.** The matrix of a linear system in saddle point form has the block structure

$$(4.4) \quad \mathcal{A} = \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix},$$

where  $A = A^T \in \mathbb{R}^{n \times n}$  is positive definite,  $B \in \mathbb{R}^{m \times n}$  with  $m \leq n$  and  $C = C^T \in \mathbb{R}^{m \times m}$  is positive semidefinite. The matrix  $\mathcal{A}$  is usually large and sparse and is easily shown to be indefinite with  $n$  positive eigenvalues and  $\text{rank}(C + BA^{-1}B^T)$  (which is typically close to  $m$ ) negative eigenvalues. In practice the indefiniteness of  $\mathcal{A}$  tends to slow down iterative solvers for linear systems (e.g., Krylov subspace methods). Liesen and Parlett [18] show that if a real scalar  $\mu$  is known such that  $\mathcal{M}(\mu) = \mathcal{A} - \mu\mathcal{J}$  with  $\mathcal{J} = \begin{bmatrix} I_n & 0 \\ 0 & -I_m \end{bmatrix}$  is positive definite then one can construct a conjugate gradient (CG) iteration for solving the linear system  $\mathcal{J}\mathcal{A}x = \mathcal{J}b$ . They prove [18, Thm. 2.2] that  $\mathcal{M}(\mu)$  is positive definite if and only if

$$(4.5) \quad \lambda_{\min}(A) > \mu > \lambda_{\max}(C) \quad \text{and} \quad \|(\mu I - C)^{-1/2}B(A - \mu I)^{-1/2}\|_2 < 1.$$

Finding a  $\mu$  satisfying these two conditions can be difficult in practice when the matrices are large. Liesen and Parlett show that if  $\lambda_{\min}(A) > \lambda_{\max}(C)$  then  $\mathcal{M}(\tilde{\mu})$  with  $\tilde{\mu} = (\lambda_{\min}(A) + \lambda_{\max}(C))/2$  is positive definite if

$$(4.6) \quad 2\|B\|_2 < \lambda_{\min}(A) - \lambda_{\max}(C),$$

which is a sufficient condition that is easier to check but is often too restrictive, as illustrated by numerical examples in [18].

We note that checking whether  $\mathcal{A} - \mu\mathcal{J}$  is positive definite for some  $\mu \in \mathbb{R} \cup \{\infty\}$  is equivalent to checking whether  $\mathcal{A} \sin t + \mathcal{J} \cos t$  is positive definite for some  $t \in [0, 2\pi]$ . In other words, Algorithm 2.3 armed with a sparse Cholesky factorization routine to test definiteness and to compute directions of negative curvature can be used to test if the pair  $(\mathcal{A}, \mathcal{J})$  is definite and to find  $\mu = -\cos t / \sin t$  such that  $\mathcal{M}(\mu)$  is positive definite, thus avoiding the need to work with (4.5) or the sufficient condition (4.6). Experiment 6 in Section 5 shows the effectiveness of this approach.

**4.3. Computing the Crawford number.** Once Algorithm 2.3 has been applied to  $(A, B)$  we know that the pair is indefinite, in which case  $\gamma(A, B) = 0$ , or that it is definite, in which case we know that  $\gamma(A, B)$  is positive but do not know its value. We now consider how to compute  $\gamma(A, B)$  efficiently for a definite pair by making use of the information provided by Algorithm 2.3.

To compute  $\gamma(A, B)$  for definite  $(A, B)$  it is useful to reformulate (2.4) as a single variable optimization problem (see, e.g., [16, Sec. 2.2]):

$$(4.7) \quad \gamma(A, B) = \max_{-\pi \leq \theta \leq \pi} g(\theta), \quad g(\theta) = \lambda_{\min}(A \cos \theta + B \sin \theta).$$

For definite pairs Algorithm 2.3 provides upper bounds for  $\gamma(A, B)$  for free since it computes  $f(x) = x^*(A + iB)x/d(x)$  with  $d(x) = |x^*(A + iB)x|$  and by definition,

$\gamma(A, B) \leq d(x)$ . We can simply return the smallest bound at the end of the computation. Note that once  $t$  is known such that  $B(t)$  is positive definite,  $\lambda_{\min}(B(t))$  provides a lower bound for  $\gamma(A, B)$ , although this requires extra computation.

In the rest of this section we show that Algorithm 2.3 can be used to simplify the univariate non-convex optimization problem over  $[-\pi, \pi]$  in (4.7) into one that is quasiconvex on a smaller interval  $(t_1, t_2)$  of length less than  $\pi$  and thereby easier to solve numerically. The following result of Veselić [27, Thm. A1] is needed.

**THEOREM 4.1.** *Suppose that the Hermitian pair  $(A, B)$  is definite, where  $B \in \mathbb{C}^{n \times n}$  is nonsingular with  $m$  positive eigenvalues and  $n-m$  negative eigenvalues. Then there exists a nonsingular matrix  $Z$  such that*

$$(4.8a) \quad Z^*AZ = \begin{bmatrix} \Lambda_+ & 0 \\ 0 & -\Lambda_- \end{bmatrix}, \quad Z^*BZ = \begin{bmatrix} I_m & 0 \\ 0 & -I_{n-m} \end{bmatrix},$$

$$(4.8b) \quad \Lambda_+ = \text{diag}(\lambda_1^+, \dots, \lambda_m^+), \quad \Lambda_- = \text{diag}(\lambda_1^-, \dots, \lambda_{n-m}^-),$$

$$(4.8c) \quad \lambda_1^+ \geq \dots \geq \lambda_m^+, \quad \lambda_1^- \geq \dots \geq \lambda_{n-m}^-.$$

Moreover for  $\mu \in \mathbb{R}$ ,  $A - \mu B$  is positive definite if and only if

$$(4.9) \quad \lambda_m^+ > \mu > \lambda_1^-,$$

with the convention that  $\lambda_1^- = -\infty$  when  $m = n$  and  $\lambda_m^+ = +\infty$  when  $m = 0$ .

*Proof.* Since  $(A, B)$  is definite there exists a nonsingular  $Z_1 \in \mathbb{C}^{n \times n}$  such that  $Z_1^*AZ_1 = D_1$  and  $Z_1^*BZ_1 = D_2$ , where  $D_1$  and  $D_2$  are diagonal and the first  $m$  diagonal elements of  $D_2$  are positive and the last  $n-m$  diagonal elements are negative. The ordering of the entries of  $\Lambda_+$  and  $\Lambda_-$  is arbitrary, so we can assume (4.8c). Setting  $Z = Z_1|D_2|^{1/2}$  we have (4.8). Then  $Z^*(A - \mu B)Z = \text{diag}(\Lambda_+ - \mu I_m, -\Lambda_- + \mu I_{n-m})$ , and this matrix is positive definite if and only if the inequalities (4.9) hold.  $\square$

Under the conditions of Theorem 4.1 the Hermitian matrix  $A - \mu B$  is positive definite precisely when  $\mu \in (\lambda_1^-, \lambda_m^+)$  or, equivalently,  $A \cos \theta + B \sin \theta > 0$  with  $\cos \theta = 1/\sqrt{1 + \mu^2}$  and  $\sin \theta = -\mu/\sqrt{1 + \mu^2}$  precisely when  $\theta$  is in the interval  $(t_1, t_2)$  with endpoints  $-\tan^{-1}(\lambda_1^-)$  and  $-\tan^{-1}(\lambda_m^+)$ . Hence it follows that

$$\gamma(A, B) = \max_{-\pi \leq \theta \leq \pi} g(\theta) = \max_{t_1 < \theta < t_2} g(\theta).$$

If  $t$  is the particular angle at which all the eigenvalues of  $A \cos \theta + B \sin \theta$  become positive then these eigenvalues become all negative at  $t + \pi$  since  $A \cos(t + \pi) + B \sin(t + \pi) = -(A \cos t + B \sin t)$ . Hence the length of  $(t_1, t_2)$  is at most  $\pi$ .

We now show that on the interval  $(t_1, t_2)$  the function  $g(\theta)$  is quasiconcave (or equivalently that  $-g(\theta)$  is quasiconvex or unimodal) [5, Sec. 3.4], [8, Sec. 6.4], that is, each sublevel set

$$\mathcal{S}_\xi = \{\theta \in (t_1, t_2) : g(\theta) \geq \xi\}, \quad \xi \in \mathbb{R}$$

is an interval. We just need to consider  $\mathcal{S}_\xi$  for  $\xi \in (0, \gamma(A, B)]$  since  $g(\theta) = \lambda_{\min}(A \cos \theta + B \sin \theta) > 0$  for  $t \in (t_1, t_2)$  so that  $\mathcal{S}_\xi = (t_1, t_2)$  for all  $\xi \leq 0$  and, because  $g(\theta) \leq \gamma(A, B)$ ,  $\mathcal{S}_\xi = \emptyset$  for all  $\xi > \gamma(A, B)$ . Now for  $\xi \in (0, \gamma(A, B))$ , there exists an angle  $\theta_\xi \in (t_1, t_2)$  at which all  $n$  eigenvalues of the Hermitian matrix  $A \cos \theta + B \sin \theta - \xi I = B(\frac{\pi}{2} - \theta) - \xi I$  become positive. Higham, Tisseur, and Van Dooren [16, Sec.2.2] studied the eigenvalues of  $A \cos \theta + B \sin \theta - \xi I$  as a function of  $\theta$  and in particular how these eigenvalues cross the zero line. They showed that when the pair  $(A, B)$  is definite the

matrix  $A \cos \theta + B \sin \theta - \xi I$  has exactly  $n$  consecutive strictly increasing zero crossings followed by  $n$  consecutive strictly decreasing zero crossings in  $[\theta_\xi - \pi, \theta_\xi + \pi)$ . Hence for each level  $\xi \in (0, \gamma(A, B))$  the  $n$  eigenvalues of  $A \cos \theta + B \sin \theta - \xi I$  have exactly two zero crossings. In particular  $g(\theta) = \lambda_{\min}(A \cos \theta + B \sin \theta)$  crosses the  $\xi$  level at an increasing crossing for some  $\theta \in (t_1, \theta_\xi)$  followed by a decreasing crossing for some  $\theta \in (\theta_\xi, t_2)$ , with no other crossing, so that the sublevel  $\mathcal{S}_\xi$  is always an interval. Note that for  $\xi = \gamma(A, B)$ ,  $\mathcal{S}_\xi$  is reduced to a single point  $t_{\text{opt}}$ . Hence we conclude that  $-g(\theta)$  is quasiconvex on  $(t_1, t_2)$ .

This makes possible an efficient calculation of the Crawford number  $\gamma(A, B)$  upon termination of Algorithm 2.3. Indeed if  $t$  is known such that  $B(t) > 0$  then the computation of  $\lambda_m^+$  and  $\lambda_1^-$  requires  $10n^3/3$  operations assuming that the Cholesky factorization of  $B(t)$  is returned at the end of Algorithm 2.3. This can be done as follows. First form the definite pencil  $A(t) - \mu B(t)$  and then compute its eigenvalues by transforming it to a Hermitian standard eigenvalue problem  $Hv = \mu v$  [9] by using the Cholesky factorization of  $B(t)$ . Then tridiagonalize  $H$  and compute its eigenvalues  $\mu_j$  with the bisection algorithm. The eigenvalues  $\lambda_j$  are then given by

$$\lambda_j = \frac{\mu_j \cos t + \sin t}{\cos t - \mu_j \sin t}$$

and it is easy to identify  $\lambda_m^+$  and  $\lambda_1^-$ . As shown above, the endpoints of the interval  $(t_1, t_2)$  are given by  $-\tan^{-1}(\lambda_1^-)$  and  $-\tan^{-1}(\lambda_m^+)$ . Experiment 7 in Section 5 shows that the length of  $(t_1, t_2)$  can be much less than  $\pi$ .

The quasiconcavity of  $g$  on  $(t_1, t_2)$  implies that  $g(\theta)$  increases from 0 at  $\theta = t_1$  to  $\gamma(A, B)$  at  $\theta = t_{\text{opt}}$  and then decreases towards 0 as  $\theta$  approaches  $t_2$ . This property can be used to further reduce the length of  $(t_1, t_2)$ . Indeed Algorithm 2.3 returns  $t$  such that  $B(t) = A \cos \tilde{t} + B \sin \tilde{t}$  with  $\tilde{t} = \frac{\pi}{2} - t$  is positive definite and the sign of the derivative of  $g(\theta)$  at  $\theta = \tilde{t}$  determines whether  $g$  increases or decreases for  $\theta > \tilde{t}$ . If  $g(\tilde{t})$  is a simple eigenvalue with normalized eigenvector  $v$ , the derivative of the eigenvalue is given by

$$\frac{\partial}{\partial \theta} g(\theta)|_{\tilde{t}} = v^* \frac{\partial}{\partial \theta} (A \cos \theta + B \sin \theta)|_{\tilde{t}} v = v^* (-A \sin \tilde{t} + B \cos \tilde{t}) v$$

and then

$$(4.10) \quad \gamma(A, B) = \max_{\theta \in \mathcal{I}} g(\theta) = -\min_{\theta \in \mathcal{I}} -g(\theta), \quad \mathcal{I} = \begin{cases} (t_1, \tilde{t}], & \text{if } \frac{\partial}{\partial \theta} g(\theta)|_{\tilde{t}} \leq 0, \\ [\tilde{t}, t_2), & \text{otherwise.} \end{cases}$$

The minimum in (4.10) can be found using, for example, a combination of golden section search and parabolic interpolation [8, Chap. 6] or bundle trust methods [23].

**5. Numerical experiments.** We now describe a comprehensive collection of numerical experiments designed to give insight into Algorithm 2.3, the implementation issues, its performance in floating point arithmetic, and its behavior on the applications described in Section 4. Our computations were done in MATLAB 7.6 (R2008a) under Windows XP (SP3) with a Pentium E6850, for which  $u = 2^{-53} \approx 1.1 \times 10^{-16}$ , and we set  $\text{tol} = nu$  in Algorithm 2.3. In our tests we obtained our reference answer of whether a pair  $(A, B)$  is definite by computing the global maximum over  $t \in [0, 2\pi]$  of the function  $\lambda_{\min}(A \sin t + B \cos t)$ ; the pair is definite if and only

TABLE 5.1  
Results for  $10 \times 10$  pair  $(A, B)$  in Experiment 1.

	iterations	curvatures (3.1)
PDFIND	8	$-1.84\text{e-}16, 1.33\text{e-}17, -1.91\text{e-}1, -3.33\text{e-}16, -1.60\text{e-}16, -1.63\text{e-}16,$ $-3.12\text{e-}3, -2.50\text{e-}1$
Chol	6	$2.67\text{e-}17, -3.83\text{e-}1, -6.11\text{e-}16, -3.64\text{e-}2, -6.33\text{e-}3, -4.93\text{e-}1$
Chol(cp)	2	$-9.97\text{e-}1, -4.68\text{e-}1$
Chol(cp2)	2	$-9.97\text{e-}1, -4.68\text{e-}1$
Chol(ecp)	2	$-9.97\text{e-}1, -2.37\text{e-}1$
eig	2	$-9.75\text{e-}1, -1.95\text{e-}1$

if the maximum is positive. A very convenient way to obtain the global maximum is with the Chebfun package [3], [26], using the statement `max(chebfun(@(t) min(eig(sin(t)*A+cos(t)*B)), [0 2*pi]))`.

We compare several variants of Algorithm 2.3 differing in how they carry out the definiteness test and compute the direction of negative curvature:

**Chol:** Cholesky without pivoting and (3.3),

**Chol(cp):** Cholesky with complete pivoting and (3.3),

**Chol(cp2):** Cholesky with complete pivoting and (3.4), (3.5),

**Chol(ecp):** Cholesky with early exit complete pivoting and (3.3),

**eig:** MATLAB's `eig` function (which implements the QR algorithm), taking as direction an eigenvector of  $B(t)$  corresponding to the smallest eigenvalue.

We also use PDFIND from [6].

*Experiment 1.* Our first example is a  $10 \times 10$  indefinite pair generated by the subroutine GETMAT provided with [6], with parameter KPAR equal to 6. Table 5.1 shows the number of iterations and the values of the curvature (3.1) on successive iterations. PDFIND and Chol require 3 or 4 times as many iterations as the other methods and the curvature values reveal why: whereas the other methods achieve curvature of almost  $-1$  on the first iteration, immediately obtaining an arc of length close to  $\pi$ , PDFIND and Chol only gradually expand the arc beyond length  $\pi$ . The difference is entirely due to the way in which the direction of negative curvature is generated.

*Experiment 2.* Our next example concerns a class of pairs suggested by Moon [21, Example 3.0, p. 80], which is most easily defined by the MATLAB code that generates the  $n \times n$  instance,  $(A_n, B_n)$ :

```
V = gallery('triu',n,1,2);
theta = zeros(n,1);
for i = 2:n, theta(i) = theta(i-1) + pi/2^(i-1); end
A = V'*diag(sin(theta))*V;
B = V'*diag(cos(theta))*V;
```

$A_n$  is positive semidefinite and singular,  $B_n$  is singular and indefinite, and the pair  $(A_n, B_n)$  is definite. Figure 5.1 plots the field of values  $F(A_{50} + iB_{50})$  as well as the unit circle. It is clear that  $\text{arc}[\tilde{a}, \tilde{b}]$ , which is the projection of the field of values onto the unit circle, is an arc of length approximately  $\pi$ , and indeed  $(A_n, B_n)$  is within distance of order  $u$  of an indefinite pair for  $n \geq 54$ . This is a problem where the tolerance `tol` in Algorithm 2.3 plays a key role. Table 5.2 shows the numbers of iterations for  $n = 64$  and  $n = 80$  with `tol = 0` and with `tol = nu`. An iteration count of 100 indicates that the algorithm had not converged after 100 iterations, and in the four such cases there were many instances of positive curvature and arc shrinkage. In all other cases an indefinite pair was signalled. Each variant of Algorithm 2.3



TABLE 5.3

Number of iterations for the pair  $(A_1, B_1)$  defined by (4.1) and (5.1) for  $\beta = 0.51961524227066xy$ , where “ $xy$ ” denotes the 15th and 16th decimal places of  $\beta$ .

$xy$	20	22	24	26	28	30	32	34	36	38	40
PDFIND	26	26	27	27	27	28	25	25	25	25	25
Chol	27	26	26	26	26	27	26	24	24	24	24
Chol(cp)	17	17	18	18	17	17	19	18	18	18	17
Chol(cp2)	17	17	18	18	17	17	19	18	18	18	17
Chol(ecp)	18	19	19	20	21	19	19	19	17	19	17
eig	18	18	18	18	18	19	20	19	18	20	20

TABLE 5.4

Number of iterations for the pair  $(A_1, B_1)$  defined by (4.1) and (5.1).

$\beta$	0.500	0.504	0.508	0.512	0.516	0.520	0.524	0.528
PDFIND	7	7	7	7	8	3	3	3
Chol	6	6	6	6	7	2	2	2
Chol(cp)	2	2	2	2	2	2	2	2
Chol(cp2)	2	2	2	2	2	2	2	2
Chol(ecp)	6	5	5	5	5	5	2	2
eig	3	3	3	4	4	5	2	2

MATLAB toolbox [4] via `nlevp('spring', 100, 1, 10*ones(100, 1))`. We know that the definiteness of  $(A_1, B_1)$  changes for some  $\beta$  near 0.51961524227066. So we take  $\beta = 0.51961524227066xy$ , where the last two digits  $xy$  are to be specified (and we note that rounding causes our actual  $\beta$  values to be tiny perturbations of these). Table 5.3 reports the number of iterations taken by each method.

The methods all correctly determined that the pair is indefinite for the first six  $\beta$  and definite for the last five  $\beta$ ,

Shrinkage of the arc occurred for eig for two different values of  $\beta$ , once for each  $\beta$ , but not at all for the Chol variants. The number of times a positive value of (3.1) was observed was 3 for PDFIND, 4 for Chol, 1 each for Chol(cp) and Chol(cp2), none for Chol(ecp), and 4 for eig, with the value of (3.1) being at most of order  $10^{-7}$ .

We note that the behavior of PDFIND is very sensitive to the computing environment: under MATLAB 7.5 (R2007b) on a Windows XP (SP3) machine with Athlon X2 processor PDFIND aborts with failure for three of the eleven values of  $\beta$ .

The main points to note from this test are

1. The three Chol variants using Cholesky with pivoting require significantly fewer iterations than that using no pivoting.
2. The maximum values of the condition number  $\text{cond}(R_{11})$  in (3.6) are of order  $10^8$  for Chol and 10 for the other Chol variants. Thus the triangular systems that are solved to obtain the direction of negative curvature are much better conditioned when pivoting is used.
3. Eig requires a comparable number of iterations to the Cholesky with pivoting variants but is much more expensive overall, since one iteration of the eig variant costs as much as four or more iterations of the Cholesky variants.

Table 5.4 reports the results for several  $\beta$  not close to a point where the definiteness of  $(A_1, B_1)$  changes. These results are more representative of the typical behavior of the algorithms. There were no arc shrinkages or instances of positive curvature, and the possible benefits of pivoting in the Cholesky factorization are again apparent.

*Experiment 4.* Our next example shows a small improvement brought by Chol(cp2)’s

TABLE 5.5

For the pair  $(A_1, B_1)$  defined by (4.1) and (5.1) along with the scaling  $A \leftarrow \alpha^2 A$ ,  $B \leftarrow \alpha B$ ,  $\alpha = 10^{-7}$ , determinations of definiteness (1 = definite, 0 = indefinite, -1 = failure), with number of iterations in parentheses.

$\beta$	Chol	Chol*	Chol-cong	Chol-cong*	PDFIND-cong	PDFIND
0.51965	1 (2)	0 (2)	1 (1)	0 (5)	0 (7)	0 (7)
0.51966	1 (2)	0 (2)	1 (1)	0 (5)	0 (7)	-1(7)
0.51967	1 (2)	0 (2)	1 (1)	1 (6)	0 (7)	0 (7)
0.51968	1 (2)	0 (2)	1 (1)	0 (5)	0 (7)	0 (7)
0.51969	1 (2)	0 (2)	1 (1)	0 (5)	0 (7)	-1(7)
0.51970	1 (2)	0 (2)	1 (1)	0 (6)	0 (7)	-1(7)
0.51971	1 (2)	0 (2)	1 (1)	0 (5)	1 (3)	1 (3)

choice of direction of negative curvature. Let

$$R = \begin{bmatrix} 2 & -1/3 & -1/3 & -1/3 \\ 0 & 1 & -1/3 & -1/3 \end{bmatrix}, \quad A = R^*R + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad B = \text{diag}(0, 1, 1, 1).$$

For this  $A$  and  $B$ , the first few lines of Algorithm 2.3 form  $a = 1 = \sin(\pi/2)$  and attempt to Cholesky factorize  $B(\pi/2) = A$ , which is an indefinite matrix with  $\lambda_{\min}(A) = -1$ . In exact arithmetic the factorization terminates at the start of stage 3 with remaining Schur complement  $S = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ . In floating point arithmetic we obtain a zero computed  $\hat{s}_{11}$ , but the computed  $\hat{x}$  for Chol, Chol(cp), and Chol(ecp) is such that positive curvature on the order of  $10^{-17}$  is generated; nevertheless, each variant correctly identifies definiteness in 3 iterations without any shrinkages occurring. Chol(cp2), by contrast, generates curvature of  $-0.7$  and converges in 2 iterations; eig behaves in exactly the same way. Here, the Crawford number  $\gamma = 0.75$ , so the pair is safely definite. If our main goal was to generate directions of negative curvature—perhaps for use in a Newton algorithm for optimization, say—then the use of (3.4) might be important. However, in our testing we have not seen any significant advantage of (3.4) over (3.3) in the context of Algorithm 2.3.

*Experiment 5.* In [11] we showed experimentally on some scaled quadratic matrix polynomials  $Q$  that yield ill-conditioned congruences in (4.3) that the code PDFIND from [6] is unreliable as a means for testing for hyperbolicity of  $Q$ , whether it is used on the  $2n \times 2n$  pair  $(A_1, B_1)$  or in conjunction with the congruence. We repeated the experiments of [11, Sec. 5.2] with Algorithm 2.3 and a modification of Algorithm 2.3 that exploits the congruence to work only with the quadratic  $Q$ . We found that these algorithms return the *correct* determination on these experiments. To gain insight we consider a particular example in detail. We take the same problem as in Experiment 3 but we scale  $A \leftarrow \alpha^2 A$ ,  $B \leftarrow \alpha B$  with  $\alpha = 10^{-7}$ , and  $\beta$  takes 7 equally spaced values from 0.51965 to 0.51971. We run the Chol variant of Algorithm 2.3 with the use of the congruence (denoted Chol-cong) and without the congruence, PDFIND, and PDFIND modified to use the congruence (denoted PDFIND-cong). We also run modified versions of Algorithm 2.3 with and without the use of the congruence, denoted Chol-cong\* and Chol\*, that compute the midpoint by  $c = (a + b)/|a + b|$  on line 14 of the algorithm.

The results, given in Table 5.5, are striking. Chol and Chol-cong give the correct result of “definite” in every case, but Chol\* and Chol-cong\* give the incorrect result in all but one case. It is easily checked that  $\theta[a, b]$  approaches  $\pi$  in this example. Thus the use of the midpoint formula  $c = (a + b)/|a + b|$  can clearly destroy the reliability

TABLE 5.6  
Results for  $834 \times 834$  pair  $(\mathcal{A}(\alpha), \mathcal{J})$  from Stokes equations.

$\alpha$	definite pair	iterations	stage at which Cholesky terminates								
0.1	no	5	579	56	56	57	57				
0.5	no	8	579	75	111	161	230	580	264	581	
1.0	yes	5	579	82	129	200	834				
5.0	yes	2	579	834							
10.0	yes	2	579	834							

of Algorithm 2.3. PDFIND suffers from the use of this formula in this example, and has additional problems causing it to abort with failure 3 times when applied to the pair.

*Experiment 6.* We now consider a saddle point system with a matrix  $\mathcal{A}$  of the form (4.4) generated by the MATLAB package Incompressible Flow Iterative Solution Software (IFISS), version 2.2 [24]. The sparse matrices  $A$ ,  $B$ , and  $C$  are constructed by running the MATLAB script file `stokes_testproblem`. We used the default options, which set up a stabilized discretization of a Stokes equation model problem. The matrix  $A$  is of dimension  $n = 578$  and  $C$  is of dimension  $m = 256$ . We find that  $\lambda_{\min}(A) = 0.0764 > \lambda_{\max}(C) = 0.0156$  and that  $\|B\|_2 = 0.2476$ , so that the sufficient condition (4.6) for definiteness of  $\mathcal{A} - \tilde{\mu}\mathcal{J}$  with  $\mathcal{J} = \begin{bmatrix} I_n & 0 \\ 0 & -I_m \end{bmatrix}$  and  $\tilde{\mu} = (\lambda_{\min}(A) + \lambda_{\max}(C))/2 = 0.046$  is not satisfied, as already noticed in [18]. Our MATLAB implementation of Algorithm 2.3 with definiteness test and direction of negative curvature computed with an attempted sparse Cholesky factorization (using MATLAB's `chol` function, without pivoting for sparsity) detects that the pair is definite in 5 iterations and returns  $\mu = 0.0589$ .

Let  $\alpha > 0$  be a real parameter and consider the scaled matrices

$$\mathcal{A}(\alpha) = \begin{bmatrix} \alpha^2 A & \alpha B^T \\ \alpha B & -C \end{bmatrix} = \begin{bmatrix} \alpha I_n & 0 \\ 0 & -I_m \end{bmatrix} \begin{bmatrix} A & B^T \\ B & C \end{bmatrix} \begin{bmatrix} \alpha I_n & 0 \\ 0 & -I_m \end{bmatrix}.$$

Note that the sufficient condition (4.6) guarantees that the pair  $(\mathcal{A}(\alpha), \mathcal{J})$  is definite for  $\alpha > (\|B\|_2 + \sqrt{\|B\|_2^2 + \lambda_{\min}(A)\lambda_{\max}(C)})/\lambda_{\min}(A) \approx 6.52$ . Table 5.6 reports the output of the definiteness test, the number of iterations and the stage at which the sparse Cholesky factorization terminates (i.e., the values of  $k$  in (3.2)) for several values of  $\alpha$ . Note that the sparse Cholesky factorizations often terminate at a stage  $k \ll n + m = 834$ , which means that the iterations are often executed at low computational cost.

*Experiment 7.* We now consider the computation of the Crawford number of  $25 \times 25$  definite pairs  $(A_k, B_k)$ ,  $k = 1:4$  generated by the subroutine GETMAT of [6] with parameter KPAR ranging from 1 to 4. The plots of  $g_k(\theta) = \lambda_{\min}(A_k \cos \theta + B_k \sin \theta)$  in Figure 5.2 illustrate the nonconvexity of  $-g_k(\theta)$  over  $\theta \in [-\pi, \pi]$  and its unimodality on the interval where  $g_k(\theta)$  is positive. Table 5.7 displays  $\gamma(A, B)$  and the corresponding optimal angle  $t_{\text{opt}}$ , lower and upper bounds  $\gamma_\ell$  and  $\gamma_u$  for  $\gamma(A, B)$  that can easily be computed via a small modification of Algorithm 2.3 as explained in Section 4.3, and the reduced interval  $\mathcal{I}$  in (4.10). Note that in three of the examples  $(\gamma_\ell, \gamma_u)$  provides a good estimate of the magnitude of  $\gamma(A, B)$  and that the length of the reduced interval  $\mathcal{I}$  is much smaller than  $2\pi$  in each case, showing that Algorithm 2.3 enables the interval of maximization to be greatly reduced.

We used MATLAB's function `fminbnd` and the Fortran code BTCLC from [28] which implements a bundle trust method, both with tolerance  $10^{-5}$ , to compute the Crawford number  $\gamma(A, B)$  by minimizing over the reduced interval  $\mathcal{I}$ . Both approaches

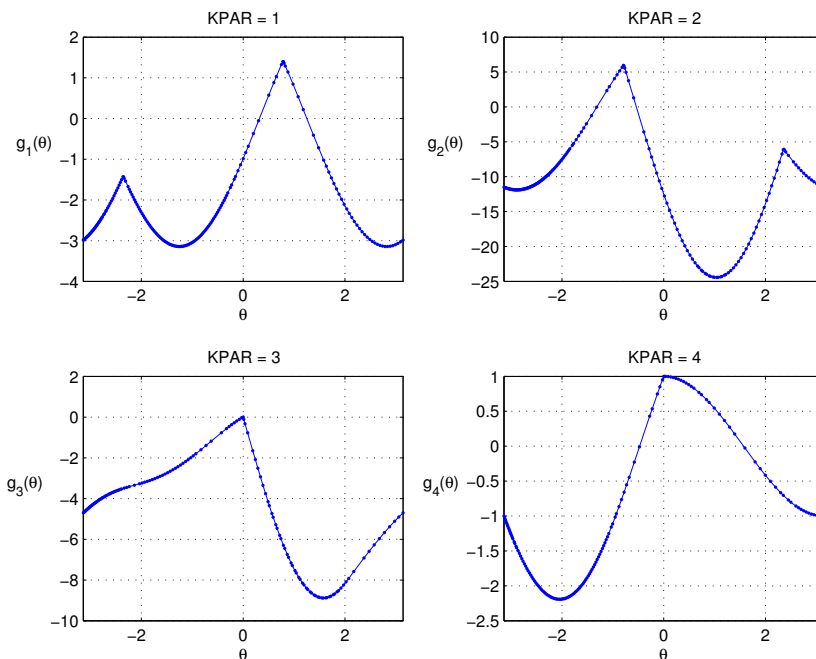


FIG. 5.2. Plots of  $g_k(\theta) = \lambda_{\min}(A_k \cos \theta + B_k \sin \theta)$  for the definite pairs  $(A, B)$  in Experiment 7.

TABLE 5.7

For the definite pairs  $(A, B)$  in Experiment 7, Crawford number  $\gamma(A, B)$ , optimal angle  $t_{\text{opt}}$ , bounds  $\gamma_\ell$  and  $\gamma_u$  on  $\gamma(A, B)$ , reduced interval  $\mathcal{I}$ , and number of evaluations of  $g(\theta)$  needed by `fminbnd` and `BTCLC` when called on the reduced interval.

KPAR	$\gamma(A, B)$	$t_{\text{opt}}$	$(\gamma_\ell, \gamma_u)$	$\mathcal{I}$	# $g(\theta)$ <code>fminbnd</code>	# $g(\theta)$ <code>BTCLC</code>
1	1.4	0.79	(1.4, 1.4)	(0.79, 1.3)	24	3
2	6.0	-0.79	(5.5, 8.1)	(-0.84, -0.54)	23	5
3	7e-9	1.6e-8	(4.7e-9, 5.9e-2)	(0.0, 9.4e-8)	1	1
4	1.0	2.7e-13	(0.71, 1.0)	(-0.47, 0.79)	25	7

return the Crawford number with at least 5 correct significant digits, but the number of evaluations of  $g(\theta) = \lambda_{\min}(A \cos \theta + B \sin \theta)$  required by `BTCLC` is much smaller for  $\text{KPAR} = 1, 2, 4$  than that required by `fminbnd`. For the  $834 \times 834$  matrix pair  $(\mathcal{A}, \mathcal{J})$  associated with the saddle point system in Experiment 6, `fminbnd` and `BTCLC` find that  $\gamma(\mathcal{A}, \mathcal{J}) = 0.0124$  using 18 function evaluations for the former and only 8 function evaluations for the latter.

Finally, we assess how the arc algorithm compares with other methods for testing definiteness of a Hermitian pair. Higham, Tisseur, and Van Dooren [16] present a bisection algorithm and a level set algorithm, both of which determine whether a Hermitian pair  $(A, B)$  is definite and also compute the Crawford number  $\gamma(A, B)$ . Each iteration of the bisection algorithm requires the solution of a quadratic eigenvalue problem and up to  $2n$  Hermitian eigenproblems, so this algorithm is much more expensive than Algorithm 2.3 and the procedure for calculating  $\gamma(A, B)$  described in Section 4.3. For testing definiteness, the level set algorithm needs to compute the eigensystem of a (single) non-Hermitian generalized eigenvalue problem, which

costs about  $46n^3$  flops if done by the QZ algorithm. This corresponds to the cost of at least 138 iterations of Algorithm 2.3, so again Algorithm 2.3 is clearly the more efficient approach to testing definiteness. Another test can be based on Veselić’s  $J$ -orthogonal Jacobi algorithm [27], but this is based on hyperbolic transformations and so is potentially unstable.

**6. Conclusions.** In this work we revisited the algorithm of Crawford and Moon for testing whether a Hermitian pair  $(A, B)$  is definite. By reworking the derivation and emphasizing the arc expansion mechanism we have obtained a fuller understanding of its behavior—in particular for indefinite pairs. Moreover, we have shown that the details of how the algorithm is initialized and terminated, how the angles are computed, and how directions of negative curvature are computed are all crucial to the efficiency of the algorithm and its behavior in floating point arithmetic. Our modified algorithm incorporates improvements to all the above aspects, and our backward error analysis justifies the algorithm’s determination of whether the pair is definite or not. Our recommended implementation is Algorithm 2.3 with Chol(cp)—Cholesky decomposition with complete pivoting. This combination has performed well in all our tests and has the advantage over the Chol(ecp) early-exit variant that the efficient code that is available in LAPACK Version 3.2 [12], [19] can be used without change.

In summary, Algorithm 2.3 with Chol(cp) is remarkably efficient in general, the computational cost being just a few (partial) Cholesky factorizations. The efficiency stems from the fact that the algorithm can quickly find a  $t$  for which  $B(t)$  is definite, or determine that the range of  $f$  is an arc of length greater than  $\pi$ , well before it has accurately determined the range of  $f$ .

## REFERENCES

- [1] Y.-H. AU-YEUNG, *A theorem on a mapping from a sphere to the circle and the simultaneous diagonalization of two Hermitian matrices*, Proc. Amer. Math. Soc., 20 (1969), pp. 545–548.
- [2] ———, *Some theorems on the real pencil and simultaneous diagonalization of two Hermitian bilinear functions*, Proc. Amer. Math. Soc., 23 (1969), pp. 246–253.
- [3] Z. BATTLES AND L. N. TREFETHEN, *An extension of MATLAB to continuous functions and operators*, SIAM J. Sci. Comput., 25 (2004), pp. 1743–1770.
- [4] T. BETCKE, N. J. HIGHAM, V. MEHRMANN, C. SCHRÖDER, AND F. TISSEUR, *NLEVP: A collection of nonlinear eigenvalue problems*. <http://www.mims.manchester.ac.uk/research/numerical-analysis/nlevp.html>.
- [5] S. BOYD AND L. VANDENBERGHE, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.
- [6] C. R. CRAWFORD, *ALGORITHM 646 PDFIND: A routine to find a positive definite linear combination of two real symmetric matrices*, ACM Trans. Math. Software, 12 (1986), pp. 278–282.
- [7] C. R. CRAWFORD AND Y. S. MOON, *Finding a positive definite linear combination of two Hermitian matrices*, Linear Algebra Appl., 51 (1983), pp. 37–48.
- [8] G. DAHLQUIST AND Å. BJÖRCK, *Numerical Methods in Scientific Computing*, vol. I, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.
- [9] P. I. DAVIES, N. J. HIGHAM, AND F. TISSEUR, *Analysis of the Cholesky method with iterative refinement for solving the symmetric definite generalized eigenproblem*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 472–493.
- [10] A. FORSGREN, P. E. GILL, AND W. MURRAY, *Computing modified Newton directions using a partial Cholesky factorization*, SIAM J. Sci. Comput., 16 (1995), pp. 139–150.
- [11] C.-H. GUO, N. J. HIGHAM, AND F. TISSEUR, *Detecting and solving hyperbolic quadratic eigenvalue problems*, MIMS EPrint 2007.117, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Sept. 2007. Revised August 2008. To appear in SIAM J. Matrix Anal. Appl.
- [12] S. HAMMARLING, N. J. HIGHAM, AND C. LUCAS, *LAPACK-style codes for pivoted Cholesky and QR updating*, in Applied Parallel Computing. State of the Art in Scientific Computing. 8th International Workshop, PARA 2006, B. Kågström, E. Elmroth, J. Dongarra, and J. Waśniewski, eds., no. 4699 in Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2007, pp. 137–146.
- [13] N. J. HIGHAM, *Computing a nearest symmetric positive semidefinite matrix*, Linear Algebra Appl., 103 (1988), pp. 103–118.
- [14] ———, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second ed., 2002.
- [15] N. J. HIGHAM, D. S. MACKAY, AND F. TISSEUR, *Definite matrix polynomials and their linearization by definite pencils*, MIMS EPrint 2007.97, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, Apr. 2008. Revised September 2008. To appear in SIAM J. Matrix Anal. Appl.
- [16] N. J. HIGHAM, F. TISSEUR, AND P. M. VAN DOOREN, *Detecting a definite Hermitian pair and a hyperbolic or elliptic quadratic eigenvalue problem, and associated nearness problems*, Linear Algebra Appl., 351–352 (2002), pp. 455–474.
- [17] I. C. F. IPSEN, *Computing an eigenvector with inverse iteration*, SIAM Rev., 39 (1997), pp. 254–291.
- [18] J. LIESEN AND B. N. PARLETT, *On nonsymmetric saddle point matrices that allow conjugate gradient iterations*, Numer. Math., 108 (2008), pp. 605–624.
- [19] C. LUCAS, *LAPACK-style codes for level 2 and 3 pivoted Cholesky factorizations*, Numerical Analysis Report No. 442, Manchester Centre for Computational Mathematics, Manchester, England, Feb. 2004.
- [20] A. S. MARKUS, *Introduction to the Spectral Theory of Polynomial Operator Pencils*, American Mathematical Society, Providence, RI, USA, 1988.
- [21] Y.-S. MOON, *On the Numerical Solution of the Definite Generalized Eigenvalue Problem*, PhD thesis, University of Toronto, Toronto, Canada, Feb. 1979.
- [22] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, Springer-Verlag, New York, second ed., 2006.
- [23] H. SCHRAMM AND J. ZOWE, *A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results*, SIAM J. Optim., 2 (1992), pp. 121–152.

- [24] D. J. SILVESTER, H. C. ELMAN, AND A. RAMAGE, *Incompressible flow and iterative solver software (IFISS) version 2.2*. <http://www.maths.manchester.ac.uk/~djs/ifiss/>.
- [25] G. W. STEWART, *Perturbation bounds for the definite generalized eigenvalue problem*, *Linear Algebra Appl.*, 23 (1979), pp. 69–85.
- [26] L. N. TREFETHEN, R. PACHÓN, R. B. PLATTE, AND T. A. DRISCOLL, *Chebfun version 2*. <http://www.comlab.ox.ac.uk/chebfun/>.
- [27] K. VESELIĆ, *A Jacobi eigenreduction algorithm for definite matrix pairs*, *Numer. Math.*, 64 (1993), pp. 241–269.
- [28] J. ZOWE, M. KOCVARA, J. OUTRATA, AND H. SCHRAMM, *Bundle trust methods: Fortran codes for nondifferentiable optimization. User's guide*, Preprint No. 259, Institute of Applied Mathematics, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2000.