# SOLVING THE TRUST-REGION SUBPROBLEM BY A GENERALIZED EIGENVALUE PROBLEM [*]

SATORU ADACHI , SATORU IWATA [†], YUJI NAKATSUKASA [‡], AND AKIKO TAKEDA [§]

**Abstract.** The state-of-the-art algorithms for solving the trust-region subproblem are based on an iterative process, involving solutions of many linear systems, eigenvalue problems, subspace optimization, or line search steps. A relatively underappreciated fact, due to Gander, Golub and von Matt in 1989, is that trust-region subproblems can be solved by one generalized eigenvalue problem, with no outer iterations. In this paper we rediscover this fact and discover its great practicality, which exhibits good performance both in accuracy and efficiency. Moreover, we generalize the approach in various directions, namely by allowing for an ellipsoidal constraint, dealing with the so-called hard case, and obtaining approximate solutions efficiently when high accuracy is unnecessary. We demonstrate that the resulting algorithm is a general-purpose TRS solver, effective both for dense and large-sparse problems, including the so-called hard case. Our algorithm is easy to implement: its essence is a few lines of MATLAB code.

**Key words.** Trust-region subproblem, generalized eigenvalue problem, elliptic inner product, hard case

**AMS subject classifications.** 49M37, 65K05, 90C25, 90C30

## 1. Introduction.

The trust-region subproblem (TRS) [4], [26, Ch. 4] is to

$$\underset{\|p\|_B \le \Delta}{\text{minimize}} \; g^\top p + \frac{1}{2} p^\top A p, \tag{1.1}$$

where $A, B \in \mathbb{R}^{n\times n}$ are symmetric, $g \in \mathbb{R}^n$, $\Delta > 0$, and $B$ is symmetric positive definite. Trust-region methods are a popular approach to dealing with general nonlinear optimization problems to minimize $f(x)$, in which each iteration requires an (approximate) solution for TRS (1.1). In a trust-region method, the objective function of TRS (1.1) is a quadratic model of $f$ near the current approximate solution $\tilde{x}$, in which $A$ is the Hessian and $g$ is the gradient of $f$ at $\tilde{x}$. Note that we allow for the constraint $\|p\|_B \le \Delta$ in an ellipsoidal norm, defined by $\|p\|_B = \|B^{1/2}p\| = \sqrt{p^\top B p}$ for a positive definite matrix $B$, not necessarily equal to the identity $I$. An appropriate and nontrivial choice $B \ne I$ can be important for example when working in a properly scaled trust region to solve the nonlinear problem efficiently [23]. For example, in [4] it is argued that a good choice would be to take $B = |A|$, the Hermitian polar factor [17] of $A$. For more on trust-region methods and TRS, see the book [4].

The necessary and sufficient optimality condition for TRS (1.1) is the following:

THEOREM 1.1. *A vector $p_*$ is an optimal solution to the TRS* (1.1) *if and only*

[*]This research is supported by Grants-in-Aid for Scientific Research (Challenging Exploratory Research, No. 26540007) from Japan Society for Promotion of Science.

[†]Department of Mathematical Informatics, The University of Tokyo, Tokyo 113-8656, Japan ({satoru_adachi, iwata}@mist.i.u-tokyo.ac.jp)

[‡]Mathematical Institute, University of Oxford, Oxford OX2 6GG, UK (nakatsukasa@maths.ox.ac.uk) Supported by JSPS as Postdoctoral Fellow for Research Abroad.

[§]Department of Mathematical Analysis and Statistical Inference, The Institute of Statistical Mathematics, 10-3 Midori-cho, Tachikawa, Tokyo 190-8562, Japan (atakeda@ism.ac.jp)

*if there exists* $\lambda_* \geq 0$ *such that*

$$\|p_*\|_B \leq \Delta, \tag{1.2}$$

$$(A + \lambda_* B)p_* = -g, \tag{1.3}$$

$$\lambda_*(\Delta - \|p_*\|_B) = 0, \tag{1.4}$$

$$A + \lambda_* B \succeq 0. \tag{1.5}$$

The equations (1.2)-(1.5) are mentioned in [4, Thm. 7.4.1] and [15] as a necessary condition. Theorem 1.1 is well known when $B = I$, and it can be obtained by using a change of variables in the discussion in Gay [11] and Moré and Sorensen [25] (see also [26, Ch. 4]) to allow for a general $B \succ 0$.

Most existing algorithms for the TRS focus on $B = I$ and attempt to find a TRS solution $(\lambda_*, p_*)$ in Theorem 1.1 by an iterative process, during which a parameter such as an estimate for $\lambda_*$ is updated. Algorithms for TRS can be classified into three broad categories as we summarize below, together with the representative studies.

1. *Accurate methods for dense problems.* The classical algorithm by Moré and Sorensen [25] iteratively solves symmetric positive-definite linear systems via the Cholesky factorization. During the iteration the estimate for $\lambda_*$ is adjusted. Safeguard techniques are sometimes necessary to ensure convergence to the solution. This is the standard approach for moderate-sized problems, say $n \leq 1000$.
2. *Accurate methods for large-sparse problems.* Sorensen's algorithm [35], refined and implemented by Rojas et al. [32, 33], iteratively computes the smallest eigenvalue of a parameterized matrix of the form $\left[\begin{smallmatrix} \alpha & g^\top \\ g & A \end{smallmatrix}\right]$, where $\alpha$ is a parameter adjusted during the iteration to find the solution. Again, some safeguard technique is needed to guarantee convergence. Another approach due to Rendl and Wolkowicz [30] solves TRS via semidefinite programming (SDP), for which the standard solver based on the interior-point method also involves an iteration of linear systems [2]. A modification of the Moré-Sorensen algorithm using Taylor series is presented by Gould, Robinson and Thorne [15], and other accurate algorithms include subspace projection methods, such as Hager [16] and Erway and Gill [6].
3. *Approximation methods.* The Steihaug-Toint algorithm [36, 39] is a well-known method that employs a truncated conjugate gradient step (see [4, Sec. 7.5]). Also known is the approach by Gould et al. [13] using truncated Lanczos steps. These algorithms allow one to pursue the practical goal of reducing the overall cost of solving the nonlinear optimization problem via TRS, in which a sufficient reduction in the objective value (1.1) is usually adequate, and finding the exact minimizer of TRS is unnecessary. Other approaches that iteratively solve the TRS approximately include the dogleg method [26, Sec. 4.1] and DC-based algorithms [38].

All these algorithms require iterations of a computational routine, and the number of iterations is often unpredictable and potentially large.

One exception to the iteration-based methods was proposed in 1989 by Gander, Golub and von Matt [10], which reduces TRS to a *single* quadratic eigenvalue problem, which they linearize to obtain a standard eigenvalue problem of size $2n$. However, in that paper they report that their eigenvalue-based approach is slower and less accurate than Moré-Sorensen's algorithm. This is perhaps why this approach appears to have received less attention by TRS researchers than those mentioned above; another reason may be that the paper [10] revolves around (constrained) eigenvalue problems rather than TRS.

The algorithm we advocate here, however, results in an extension of [10], which turns out to be both efficient and accurate. Apparently, the slow speed and loss of accuracy reported in [10] was largely due to the relatively unrefined eigenvalue solver available those days. We demonstrate that the accuracy and speed are both dramatically improved by the highly developed eigensolvers available today, such as the shift-and-invert Arnoldi method [20]. Our MATLAB experiments demonstrate that our algorithm has excellent accuracy and competitive speed compared with existing algorithms in the standard case $B = I$ with large-sparse $A$.

Our approach, based on a different derivation from [10], further extends the algorithm by allowing $B \neq I$ and preserving symmetry. Moreover, the paper [10] does not discuss how to deal with the "hard case", which are TRS whose $\lambda_*$ is equal to the largest eigenvalue of the pencil $A + \lambda B$ (which implies by (1.3) that $g \perp \mathcal{N}(A + \lambda_* B)$, where $\mathcal{N}(A+\lambda_* B)$ denotes the null space of $A+\lambda_* B$). In this paper we treat the hard case in detail, and illustrate that our algorithm solves the hard case efficiently with high accuracy. We also demonstrate its effectiveness for computing an approximate solution efficiently.

Note that the lack of iterations makes our algorithm easy to implement (its essence is a few lines of MATLAB code), and the runtime predictable. Moreover, difficulties associated with iterations–such as the need for safeguard techniques–simply disappear. Overall, this paper introduces a practical, general-purpose TRS algorithm, useful for a wide range of problems, from dense-medium sized to large-sparse; we have experimented successfully with problems of size up to $10^7$.

The algorithms in [25, 30] are designed for $B = I$, and so are most publically available codes [30, 33]. Algorithms applicable to $B \neq I$ are discussed in [4, Sec. 7.5.6], [13, 15], but these algorithms are approximate and/or iterative. The traditional approach to deal with $B \neq I$ is to reduce the problem to an equivalent one with $B = I$ by a change of variables: defining $\tilde{p} := B^{\frac{1}{2}} p$, one can reduce (1.1) to an equivalent TRS with $B = I$, i.e., the minimization of $(B^{-\frac{1}{2}} g)^\top \tilde{p} + \frac{1}{2}\tilde{p}^\top (B^{-\frac{1}{2}} A B^{-\frac{1}{2}})\tilde{p}$ subject to $\|\tilde{p}\| \leq \Delta$. However, this reduction involves computing the matrix square root or the Cholesky factor of $B$ and their inverse, which can be expensive and numerically unstable. Furthermore, the conversion to $B^{-\frac{1}{2}} A B^{-\frac{1}{2}}$ generally destroys the problem structure: e.g., the inverse of an irreducible tridiagonal matrix is dense (matrix-free algorithms such as [32] avoids forming $B^{-\frac{1}{2}} A B^{-\frac{1}{2}}$ by working with the (sparse) Cholesky factor of $B$). Our algorithm works directly with $A, B$ and thus takes full advantage of the sparsity.

Besides choosing $B$ to reflect the geometry of the problem such as $B \approx |A|$, another situation where an ellipsoidal norm arises is when a standard TRS with $B = I$ is solved via the Steihaug-Toint conjugate gradient-based algorithm [36, 39] with preconditioning. We note that in [5] it is suggested that precautions are needed when such change of the inner product is employed, as it can alter the quality of the solution. This also indicates the importance of the choice of $B$. This paper makes no claims on how to choose $B$, but rather takes it as given, and we focus on solving the TRS (1.1).

Our algorithm can be regarded as an extension of a related eigenvalue-based algorithm in [19] for the point-ellipsoid distance problem, which minimizes $\|x\|_2$ subject to the constraint that $x$ lies on the boundary of an ellipsoid. This problem has a convex objective but a nonconvex constraint, and a generalized eigenproblem is derived in [19] that yields a global solution. We note that Rimon and Boyd [31] also suggests using the algorithm by Gander, Golub and von Matt [10] mentioned above for computing point-ellipsoid distances.

Along with [19], this work was initially inspired by the fact that some quadratic optimization problems lead to polynomial equations, as follows. Consider TRS in the simple case $B = I$ and let $A = VDV^\top$ be the eigenvalue decomposition with eigenvalues $d_i, i = 1, \dots, n$. We focus on the solution with $\|p_*\| = \mathit{\Delta}$, which is generally the more difficult case than interior solutions $\|p_*\| < \mathit{\Delta}$ (see Section 2.1). Then by (1.3) we have $p_* = -V(D + \lambda I)^{-1}V^\top g$, and writing out the condition $\|p_*\| = \mathit{\Delta}$ and defining $\hat{g} = V^\top g$, one sees that the solution can be obtained via a rational equation in $\lambda$ of the form

$$\mathit{\Delta}^2 = \sum_{j=1}^{n} \frac{\hat{g}_j^2}{(d_j + \lambda)^2}. \tag{1.6}$$

This equation has been presented in the literature [11, 25, 26], but is usually treated as a "difficult" nonlinear equation that needs to be solved through an iterative process. Nonetheless, (1.6) is nothing else than a rational rootfinding problem, which can mathematically be reduced to a polynomial rootfinding problem by multiplying out $\prod_{j=1}^{n}(d_j + \lambda)^2$, which can then be solved by a *single* eigenvalue problem via linearization such as the companion matrix, without iterations (except those used within the eigensolver). Although this "polynomialization" is usually not recommended due to numerical instability (and we will not pursue it), this observation does suggest that a method based on iterations is perhaps unnecessary.

This paper is organized as follows. In Section 2 we detail the optimality conditions and the KKT conditions. We then describe our main algorithm in Section 3. Section 4 discusses how to deal with the hard case. We summarize the algorithm in pseudocode in Section 5. In Section 6 we compare our algorithm with existing methods. Section 7 shows numerical experiments to illustrate and compare the performance.

**Notation.** Throughout this paper, $\mathcal{R}(X)$ denotes the range of a matrix $X$, and $\mathcal{N}(X)$ the null space. In addition, $X \succeq 0$ indicates $X$ is a positive semidefinite matrix, $I_n$ is the $n \times n$ identity, and $O_n, O_{m\times n}$ are the $n \times n$ and $m \times n$ zero matrices; we simply write $I, O$ if the dimensions are clear from the context. We always denote a TRS solution by $p_*$ with associated Lagrange multiplier $\lambda_*$. Computed approximants wear a hat, so for example $\hat{p}_*$ is a computed approximant to $p_*$. We denote by $u$ the unit roundoff, $u \approx 1.1 \times 10^{-16}$ in double precision arithmetic.

## 2. Interior and boundary solutions.

Equation (1.4) shows that a TRS solution $(\lambda_*, p_*)$ belongs to either (or both) of the following two cases, known as complementary slackness: $\lambda_* = 0$, or $\mathit{\Delta} - \|p_*\|_B = 0$. In view of this, we separate the TRS solution into the following two types:

(i) an interior solution with $\|p_*\|_B < \mathit{\Delta}$,
(ii) a boundary solution $\|p_*\|_B = \mathit{\Delta}$.

We must have $\lambda_* = 0$ in case (i). The "hard case" arises in the boundary case (ii), if we further have $\det(A + \lambda_* B) = 0$; see Section 4.

Below we first discuss how to deal with an interior solution, and then treat the boundary solutions (ii). Our algorithm by default proceeds by executing both steps to find one or two candidates, and obtains a TRS solution by comparing the objective value.

**2.1. Interior solutions.** An interior solution clearly satisfies $\lambda_* = 0$ by (1.4). Obtaining an interior solution is done by solving the linear system $Ap_0 = -g$ for $p_0$. We then check if the constraint $\|p_0\|_B < \varDelta$ is satisfied. If it is, then $p_0$ is a candidate for an interior TRS solution.

If further $A \succ 0$, then $p_0$ is indeed the TRS solution, If $A$ has negative eigenvalues, by (1.5), $p_0$ is not a TRS solution. Similarly, if $A \succeq 0$ has a zero eigenvalue, then solving the linear system $Ap_0 = -g$ becomes nontrivial, but a TRS solution must exist on the boundary (case (ii)). In practice, checking the positive definiteness of $A$ may be a costly operation (requiring $\mathcal{O}(n^3)$ if $A$ is dense), so instead, we simply attempt to solve $Ap_0 = -g$ using a direct solver or preconditioned CG, and keep $p_0$ as a candidate if $\|p_0\|_B < \varDelta$ and discard it if not.

**2.2. Boundary solutions.** The first three (1.2)–(1.4) of the TRS optimality conditions in Theorem 1.1 represent the KKT conditions. The last (1.5) specifies which of the many (up to $2n$, as we discuss in Section 3.2) KKT multipliers corresponds to the solution; we will show it is the largest real one.

By (1.3), for any KKT multiplier $\lambda$, unless the matrix $A + \lambda B$ is singular we can write $p$ as a function of $\lambda$ as

$$p(\lambda) = -(A + \lambda B)^{-1} g. \tag{2.1}$$

Since we focus on the case $\|p\|_B = \varDelta$, plugging (2.1) into this equation we obtain

$$g^\top (A + \lambda B)^{-1} B (A + \lambda B)^{-1} g = \varDelta^2. \tag{2.2}$$

Now we regard $A + \lambda B$ as a matrix pencil with eigenvalues $\mu_1 \le \mu_2 \le \cdots \le \mu_n$, and let $W$ be the matrix that achieves the simultaneous diagonalization by congruence [12, § 8.7]

$$W^\top (A, B) W = (\varLambda, I), \tag{2.3}$$

where $\varLambda = -\mathrm{diag}(\mu_1, \dots, \mu_n)$; note the minus sign as here we define $\mu_i$ as the eigenvalues of the pencil $A + \lambda B$, not $A - \lambda B$. Then (2.2) can be written as

$$(W^\top g)^\top (\varLambda + \lambda I)^{-2} (W^\top g) - \varDelta^2 = 0. \tag{2.4}$$

We write $W = [w_1, \dots, w_n]$ and define

$$h(\lambda) := \sum_{j=1}^{n} \frac{(w_j^\top g)^2}{(\lambda - \mu_j)^2} - \varDelta^2. \tag{2.5}$$

Equation (2.4) can be written as a zero-finding problem $h(\lambda) = 0$, which is a rational equation with respect to $\lambda$, analogous to (1.6). This equation has $2n$ solutions in $\mathbb{C}$ (counting multiplicities), and it is possible that they are all real.

We do not work directly with $h(\lambda) = 0$ as in (2.5), because this would require the computation of the eigenvector matrix $W$, which is quite expensive, and the resulting method is not as accurate as the one we propose. Our aim is to construct a generalized eigenvalue problem whose eigenvalues contain the values of $\lambda$ satisfying (2.5), as we describe next.

**3. Eigenvalue-based algorithm for boundary TRS solutions.** We now come to the heart of the paper where we discuss finding a boundary TRS solution via one generalized eigenvalue problem.

**3.1. Key matrix pencils.** The starting point is to introduce two matrix pencils whose eigenvalues include the desired KKT multiplier $\lambda_*$. Define the $(2n+1)\times(2n+1)$ matrix pencil

$$M(\lambda) = \begin{bmatrix} \mathit{\Delta}^2 & 0 & g^\top \\ 0 & -B & A+\lambda B \\ g & A+\lambda B & O_n \end{bmatrix} \tag{3.1}$$

and the $2n \times 2n$ matrix pencil

$$\tilde{M}(\lambda) = \begin{bmatrix} -B & A+\lambda B \\ A+\lambda B & -\frac{gg^\top}{\mathit{\Delta}^2} \end{bmatrix}. \tag{3.2}$$

The crucial facts that we prove are (i) the eigenvalues of these pencils include the values of $\lambda$ satisfying the KKT conditions, and (ii) we can obtain the solution $(\lambda_*, p_*)$ via the largest real eigenpair. We first examine the connection between the eigenvalues and $\lambda_*$, and then discuss how to obtain $p_*$.

**3.2. Finding $\lambda_*$.** We show that all the Lagrange multipliers at the KKT points $(\lambda, p)$ on the boundary $\|p\|_B = \mathit{\Delta}$, including the desired $\lambda_*$, are contained in the eigenvalues of $M(\lambda)$ and $\tilde{M}(\lambda)$.

LEMMA 3.1. *For any $(\lambda, p)$ satisfying $(A+\lambda B)p = -g$ and $\|p\|_B = \mathit{\Delta}$, we have* $\det M(\lambda) = \det \tilde{M}(\lambda) = 0$.

*Proof.* If $\det(A+\lambda B) = 0$ at $\lambda$ with $(A+\lambda B)x = 0$, then it follows from $g \in \mathcal{R}(A+\lambda B)$ that $M(\lambda)\left[\begin{smallmatrix} 0_{n+1} \\ x \end{smallmatrix}\right] = 0$ and $\tilde{M}(\lambda)\left[\begin{smallmatrix} 0_n \\ x \end{smallmatrix}\right] = 0$, hence $\det M(\lambda) = \det \tilde{M}(\lambda) = 0$.

We now deal with $\lambda$ such that $\det(A+\lambda B) \neq 0$. Define $p(\lambda) = -(A+\lambda B)^{-1}g$ as in (2.1), and $X(\lambda) = \begin{bmatrix} 1 & & \\ p(\lambda) & I & \\ & & I \end{bmatrix}$. Then $X(\lambda)$ is unimodular, i.e., $\det X(\lambda) \equiv 1$, and we have

$$\begin{aligned} \det M(\lambda) &= \det X(\lambda)^T M(\lambda) X(\lambda) \qquad (3.3)\\ &= \det \begin{bmatrix} \mathit{\Delta}^2 - p(\lambda)^\top B p(\lambda) & -p(\lambda)^\top B & 0 \\ -Bp(\lambda) & -B & A+\lambda B \\ 0 & A+\lambda B & O \end{bmatrix} \\ &= (-1)^n \det(A+\lambda B)^2 \{\mathit{\Delta}^2 - p(\lambda)^\top B p(\lambda)\}. \qquad (3.4) \end{aligned}$$

If $(\lambda, p(\lambda))$ satisfies $\|p(\lambda)\|_B = \mathit{\Delta}$ and $\det(A+\lambda B) \neq 0$, then recalling (2.2) we have $\mathit{\Delta}^2 - p(\lambda)^\top B p(\lambda) = 0$, and so $\det M(\lambda) = 0$. This proves $\det M(\lambda) = 0$.

To prove $\det \tilde{M}(\lambda) = 0$, we define $T = \begin{bmatrix} 1 & & -\frac{1}{\mathit{\Delta}^2}g^\top \\ & I_n & \\ & & I_n \end{bmatrix}$ and see that

$$T^\top M(\lambda) T = \begin{bmatrix} \mathit{\Delta}^2 & & \\ & -B & A+\lambda B \\ & A+\lambda B & -\frac{1}{\mathit{\Delta}^2}gg^\top \end{bmatrix} = \begin{bmatrix} \mathit{\Delta}^2 & \\ & \tilde{M}(\lambda) \end{bmatrix}. \tag{3.5}$$

It follows that $\det M(\lambda) = \mathit{\Delta}^2 \det \tilde{M}(\lambda)$, and hence together with the above result, $\det \tilde{M}(\lambda) = \det M(\lambda) = 0$. ∎

Fortunately, both matrix pencils $M(\lambda)$ and $\tilde{M}(\lambda)$ are *regular* matrix pencils, that is, their determinants are nonzero for some $\lambda$ and the number of eigenvalues is equal to their size. To see this, observe that $\tilde{M}(\infty) := \left[\begin{smallmatrix} O_n & B \\ B & O_n \end{smallmatrix}\right]$ is nonsingular, and that $\tilde{M}(\lambda)$ is obtained from $M(\lambda)$ by taking its Schur complement. Therefore the number of eigenvalues is finite, more precisely, $2n+1$ and $2n$, respectively, and $2n$ of them match the $2n$ roots of the rational equation (2.5). Note that $2n$, the size of the matrix pencil $\tilde{M}(\lambda)$, is the smallest possible, since (2.5) is a rational equation that can be reduced to a degree-$2n$ polynomial equation with $2n$ solutions. Each real eigenvalue $\lambda$ of $M(\lambda)$ corresponds to a KKT point $p = (A + \lambda B)^{-1}g$ with KKT multiplier $\lambda$.

Lemma 3.1 shows that the TRS solution on the boundary satisfies $\det M(\lambda) = 0$ and $\det \tilde{M}(\lambda) = 0$, both representing a generalized eigenvalue problem. The eigenvalues $\lambda$ contain the Lagrange multipliers at the KKT points, so the multiplier for the TRS solution must be one of the $2n$ finite eigenvalues (note that $M(\lambda)$ has one eigenvalue at infinity, which is not the one of interest).

We next show that among these $2n$ finite eigenvalues, $\lambda_*$ corresponds to the largest real one lying in $[\mu_n, \infty)$.

THEOREM 3.2. *For a TRS solution $(\lambda_*, p_*)$ on the boundary $\|p_*\|_B = \Delta$ satisfying* (1.2)–(1.5)*, the multiplier $\lambda_*$ is equal to the largest real eigenvalue of $M(\lambda)$ (excluding $\lambda = \infty$) and $\tilde{M}(\lambda)$, and $\lambda_* \in [\mu_n, \infty)$, where $\mu_n$ is the largest eigenvalue of the pencil $A + \lambda B$.*

*Proof.* The fact $\lambda_* = \lambda_{\max} \in [\mu_n, \infty)$ has been shown in [7, 19] for special cases: [7] for $B = I$ and [19] for $A = I$; see also [24]. By a change of variables we can extend these results to the TRS (1.1).

Alternatively, we can directly obtain the result as follows. First, from $A + \lambda_* B \succeq 0$ in (1.5) we must have $\lambda_* \geq \mu_n$.

Recall from (2.5) that $\|p(\lambda_*)\|_B = \Delta$ is equivalent to $h(\lambda_*) = 0$, unless $\lambda_* = \mu_n$. To prove that $\lambda_*$ is the largest real eigenvalue of $M(\lambda)$, we consider the two cases $\lambda_* > \mu_n$ and $\lambda_* = \mu_n$ separately. First, when $\lambda_* > \mu_n$, we have $h(\lambda_*) = 0$ and $\det(A + \lambda_* B) > 0$ in (3.4). Note from (2.5) that $h(\lambda)$ is strictly decreasing on $(\mu_n, \infty)$. Hence, $\tilde{M}(\lambda)$ has exactly one real eigenvalue larger than $\mu_n$, which must be $\lambda_*$.

Next, when $\lambda_* = \mu_n$ (this is the "hard case"), from (1.3) we see that $g \perp \mathcal{N}(A + \lambda_* B)$, and so $h(\lambda)$ has no pole at $\lambda = \mu_n$, and strictly decreasing on $(\mu_{n-\ell}, \infty)$, where $\ell (\geq 1)$ is the integer such that $\mu_{n-\ell} < \mu_{n-\ell+1} = \cdots = \mu_n$. Hence $h(\lambda_*)$ is formally defined as $h(\lambda_*) = \sum_{j=1}^{n-\ell} \frac{(w_j^\top g)^2}{(\mu_n - \mu_j)^2} - \Delta^2$. Now $h(\lambda_*) + \Delta^2 = \sum_{j=1}^{n-\ell} \frac{(w_j^\top g)^2}{(\mu_n - \mu_j)^2}$ is equal to the smallest $\|p\|_B^2$ such that $(A + \lambda B)p = -g$ (we prove this below in (4.2)). Therefore, if $h(\lambda_*) > 0$, then no solution exists with (1.3) and $\|p_*\|_B = \Delta$. Hence $h(\lambda_*) \leq 0$, and since $h(\lambda)$ is strictly decreasing on $(\mu_{n-\ell}, \infty)$, there is no solution for $h(\lambda) = 0$ with $\lambda > \mu_n$, that is, $\lambda_*$ is the largest real eigenvalue of $M(\lambda)$ and $\tilde{M}(\lambda)$ also in this case. ∎

## 3.3. Obtaining $p_*$.

We next show that generically, the TRS solution $p_*$ can be obtained from the eigenvector of $M(\lambda)$ and $\tilde{M}(\lambda)$ corresponding to $\lambda_*$.

THEOREM 3.3. *The eigenvectors of $M(\lambda)$ and $\tilde{M}(\lambda)$ for the largest real eigenvalue $\lambda = \lambda_*$ correspond one-to-one as*

$$\tilde{M}(\lambda_*) \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 0 \Leftrightarrow M(\lambda_*) \begin{bmatrix} -\frac{1}{\Delta^2} g^\top y_2 \\ y_1 \\ y_2 \end{bmatrix} = 0. \tag{3.6}$$

*Moreover, if $g^\top y_2 \neq 0$ then a TRS solution $p_*$ can be obtained by*

$$p_* = -\frac{\Delta^2}{g^\top y_2} y_1. \tag{3.7}$$

*Proof.* We prove that (3.6) holds for every finite eigenvalue $\lambda$ of $M(\lambda)$. By the congruence relation (3.5), we see that if $\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right]$ is an eigenvector of $\tilde{M}(\lambda)$ then

$$\tilde{M}(\lambda)\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 0 \quad \Rightarrow \quad M(\lambda)\begin{bmatrix} -\frac{1}{\Delta^2}g^\top y_2 \\ y_1 \\ y_2 \end{bmatrix} = 0. \tag{3.8}$$

For the converse, suppose that $M(\lambda)\left[\begin{smallmatrix} t \\ y_1 \\ y_2 \end{smallmatrix}\right] = 0$. Then since $T$ in (3.5) is nonsingular with $T^{-1} = \begin{bmatrix} 1 & & \frac{1}{\Delta^2}g^\top \\ & I_n & \\ & & I_n \end{bmatrix}$, we have

$$M(\lambda)\begin{bmatrix} t \\ y_1 \\ y_2 \end{bmatrix} = T^{-\top}\begin{bmatrix} \Delta^2 & \\ & \tilde{M}(\lambda) \end{bmatrix} T^{-1}\begin{bmatrix} t \\ y_1 \\ y_2 \end{bmatrix} = T^{-\top}\begin{bmatrix} \Delta^2(t + \frac{1}{\Delta^2}g^\top y_2) \\ \tilde{M}(\lambda)\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \end{bmatrix} = 0,$$

thus $\tilde{M}(\lambda)\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = 0$ and $t = -\frac{1}{\Delta^2}g^\top y_2$. Now noting that $M(\lambda)\left[\begin{smallmatrix} t \\ y_1 \\ y_2 \end{smallmatrix}\right] = 0$ with $\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right] \neq 0$ can hold only for $\lambda = \infty$, we obtain the first statement (3.6).

We next prove (3.7). From $\tilde{M}(\lambda_*)\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right] = 0$, that is,

$$\begin{bmatrix} -B & A \\ A & -\frac{gg^\top}{\Delta^2} \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -\lambda_*\begin{bmatrix} O & B \\ B & O \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \tag{3.9}$$

the lower block gives $(A+\lambda_* B)y_1 = \frac{g(g^\top y_2)}{\Delta^2}$, which is a scalar multiple of $g$. Therefore, in view of $(A+\lambda_* B)p_* = -g$ from (1.3), we obtain (3.7) as required, provided that $g^\top y_2 \neq 0$. ∎

**Practical extraction of $p_*$.** In practice, computing the solution $p_*$ from the normalization (3.7) may introduce unnecessary numerical errors, and we choose the simpler normalization

$$p_* = -\mathrm{sign}(g^\top y_2)\Delta\frac{y_1}{\|y_1\|_B}, \tag{3.10}$$

which is on the boundary to working precision: $\|\hat{p}_*\|_B = \Delta + \mathcal{O}(u)$. The normalization (3.10) is obtained by scaling the vector $y_1$ (which is parallel to the solution $p_*$) to have $B$-norm $\Delta$.

When $g^\top y_2 = 0$, finding the solution is not straightforward; this is described in Section 4.

### 3.4. Observations for a practical implementation.

**3.4.1. The rightmost eigenvalue is real.** We have seen that $\lambda_*$ at the TRS solution is equal to the largest real eigenvalue of $M(\lambda)$ and $\tilde{M}(\lambda)$. For the eigensolver it helps to further know that $\lambda_*$ is indeed the rightmost eigenvalue, that is, there is no nonreal eigenvalue that lies to the right of $\lambda_*$.

PROPOSITION 3.4. *The rightmost eigenvalues of $\tilde{M}(\lambda)$ and $M(\lambda)$ (excluding $\infty$) are both real and equal to $\lambda_*$.*

*Proof.* It suffices to show that the rightmost eigenvalue of $\tilde{M}(\lambda)$ is real. Suppose that $\tilde{\lambda} = \alpha + \beta\mathrm{i}$ where $\alpha, \beta \in \mathbb{R}$, $\alpha > \mu_n$ and $\beta > 0$ is a nonreal eigenvalue. Then $\alpha - \beta\mathrm{i}$ must also be an eigenvalue. Now if $\alpha > \lambda_* \geq \mu_n$, then recalling (2.5) we have $h(\lambda) = \sum_{j=1}^{n} \frac{(w_j^\top g)^2}{(\lambda-\mu_j)^2} - \Delta^2$, and since the imaginary part of $\frac{1}{(\tilde{\lambda}-\mu_j)^2}$ is strictly negative for all $j$, we conclude that the imaginary part of $h(\tilde{\lambda})$ must also be strictly negative. Hence $h(\tilde{\lambda})$ cannot be 0, so $\tilde{\lambda}$ is not an eigenvalue of $\tilde{M}(\lambda)$.

The above discussion proves the statement except when $\alpha = \mu_n = \lambda_*$, that is, to exclude $\tilde{\lambda} = \lambda_* + \beta\mathrm{i}$ in the hard case. In this case, we note that the terms $w_j^\top g$ must be zero for all indices $j$ such that $\lambda_j = \lambda_*$. For the remaining indices, $w_j^\top g$ cannot be all zeros (since $W$ is nonsingular), and again we have $\mathrm{Im}(\frac{1}{(\tilde{\lambda}-\mu_j)^2}) < 0$. ∎

We have shown that the TRS solution can be obtained from the rightmost (which is the largest real) eigenpair of $M(\lambda)$ or $\tilde{M}(\lambda)$, which is generally much cheaper to compute than the whole eigenvalues via the standard QR or QZ algorithms; for example the Arnoldi algorithm provides an effective means for computing extremal eigenvalues of large-sparse matrices.

**3.4.2. Which matrix pencil to use, $M(\lambda)$ or $\tilde{M}(\lambda)$?.** The generalized eigenvalue problem $M(\lambda)$ has one additional eigenvalue at $\infty$, but the matrices involved are explicitly sparse. On the other hand, $\tilde{M}$ is one size smaller with no eigenvalue at infinity, but contains $gg^\top$, which is rank-one but dense. Thus the choice between $M$ and $\tilde{M}$ should be made based on the properties of the eigensolver available. Some eigensolvers, such as MATLAB's `eigs`, which is based on (shift-and-invert) Arnoldi [20], allow the user to provide just a routine that multiplies the matrices to a vector; in this case the rank-one structure can be exploited. For this reason we use $\tilde{M}$ in our MATLAB experiments.

**3.4.3. Comparison with the algorithm by Gander, Golub and von Matt.** The algorithm by Gander, Golub and von Matt [10] considers the case $B = I$ and finds the largest eigenvalue of the $2n \times 2n$ standard but nonsymmetric eigenvalue problem

$$\begin{bmatrix} A & -I \\ -\frac{gg^\top}{\Delta^2} & A \end{bmatrix} \begin{bmatrix} y_2 \\ y_1 \end{bmatrix} = -\lambda_* \begin{bmatrix} y_2 \\ y_1 \end{bmatrix}. \tag{3.11}$$

This is a nonsymmetric matrix, with a dense bottom-left block. We can obtain (3.11) when $B = I$ by the equivalence transformation of right-multiplying $\left[\begin{smallmatrix} O & I_n \\ I_n & O \end{smallmatrix}\right]$ to $\tilde{M}$. Therefore, we arrived at a different derivation of (3.11) and generalized it to $B \neq I$. In [10] solving TRS via (3.11) is not recommended over Moré-Sorensen's secular equation approach [22], observing the inefficiency and inaccuracy with (3.11) in their experiments.

As we shall see in our experiments, the poor performance seems to have been largely due to the relatively undeveloped eigenvalue solvers available at the time: with state-of-the-art eigensolvers the algorithm is both fast and accurate. Between solving $\tilde{M}(\lambda_*)\left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right] = 0$ and (3.11), the speed comparison depends on the architecture etc: in our experiments (3.11) was often slightly faster, so a reasonable option is to turn to (3.11) when $B = I$. In any case, our experiments illustrate that by solving (3.9), our algorithm has efficiency comparable to the state-of-the-art algorithms. We do not claim to have a faster algorithm than [10] when $B = I$ (when employing the same eigensolver), but instead we observe its practicality and extend it to $B \neq I$, and treat the hard case.

## 4. Dealing with the "hard case".

We saw in Theorem 3.3 that a TRS solution $(\lambda_*, p_*)$ can be obtained via the rightmost eigenpair of $M(\lambda)$ provided that $g^\top y_2 \neq 0$. We now discuss the case where this does not hold, i.e., $g^\top y_2 = 0$. In this case $\lambda_*$ is still obtained as the rightmost eigenvalue of $\widetilde{M}(\lambda)$ by Theorem 3.2, but finding $p_*$ requires more work.

We show that the condition $g^\top y_2 = 0$ can happen only if $\lambda_* = \mu_n$.

PROPOSITION 4.1. *In the setting of Theorem* 3.3, $g^\top y_2 = 0$ *only if* $\lambda_* = \mu_n$.

*Proof.* Recall from (3.9) that $g^\top y_2 = 0$ implies $(A + \lambda_* B)y_1 = 0$. From the first block of (3.9), we obtain

$$-By_1 + (A + \lambda_* B)y_2 = 0, \tag{4.1}$$

and hence $(A+\lambda_* B)B^{-1}(A+\lambda_* B)y_2 = 0$. Using the diagonalization $W^\top(A, B)W = (\Lambda, I)$, this yields $W^\top(\lambda_* I+\Lambda)^2 W y_2 = 0$, from which we obtain $W^\top(\lambda_* I+\Lambda)W y_2 = 0$. This is equivalent to $(A + \lambda_* B)y_2 = 0$, which, by (4.1), also implies $y_1 = 0$. Hence $y_2$ is a nonzero eigenvector of $A + \lambda B$ corresponding to the eigenvalue $\lambda_* = \mu_n$. ∎

Note that the proof also shows that if $g^\top y_2 = 0$ then $y_1$, which we usually use to extract the solution $p_*$, is zero. Also note from (1.3) that $\lambda_* = \mu_n$ implies $g \perp \mathcal{N}(A+\lambda_* B)$. Hence $g^\top y_2 = 0$ implies $\lambda_* = \mu_n$, which in turn implies $g \perp \mathcal{N}(A+\lambda_* B)$. This is the so-called "hard case", a difficulty that is known to arise in the standard $B = I$ case, and it is of course present also when $B \succ 0$ is a general positive definite matrix. We restate the definition for general $B \succ 0$.

DEFINITION 4.2. *A TRS* (1.1) *is said to be in the "hard case", if* $\lambda_* = \mu_n$, *the largest eigenvalue of the pencil* $A + \lambda B$.

We repeat that $\lambda_* = \mu_n$ implies $g \perp \mathcal{N}(A + \lambda_* B)$, which some papers take to be the definition.

It is worth noting that the last condition pertains the orthogonality between $g$ and the eigenvectors corresponding to the largest eigenvalue of $A+\lambda B$ in the standard inner product, not $B$-orthogonal $w_i^\top Bg = 0$ as one might expect since the TRS (1.1) employs the $B$-norm. We also note that a TRS solution lies on the boundary in the hard case, since we have either $\lambda_* > 0$ (trivially a boundary solution), or $A \succeq 0$ with $A$ having null vectors, so again a TRS solution is on the boundary.

Although mathematically the hard case represents only a set of TRS instances of measure zero, it can happen for matrices with special structures, and numerically there are "nearly hard cases", in which $g^\top y_2 \approx 0$ and hence $\lambda_* \approx \mu_n$. These can be equally challenging. Indeed a number of studies such as [25, 32] have focused on the hard case with $B = I$.

In our approach, the reason the hard case is difficult is that recalling (3.7), the eigenvectors of $M$ or $\widetilde{M}$ for $\lambda_*$ do not give us the TRS solution. Indeed the linear system $(A + \lambda_* B)x = -g$ has infinitely many solutions. The challenge is therefore to find the solution $p_*$ such that $(A + \lambda_* B)p_* = -g$ and $\|p_*\|_B = \Delta$. We manage to do so by modifying an approach in [8] to adapt to a general positive definite $B$.

### 4.1. Solution for the hard case.

We form a nonsingular linear system using $\mathcal{N}(A + \lambda_* B)$ such that we can obtain $p_*$ from its solution.

THEOREM 4.3. *Suppose the TRS problem* (1.1) *belongs to the "hard case" and* $(\lambda_*, p_*)$ *satisfies* (1.3)–(1.5) *and* $\|p_*\|_B = \Delta$ *with* $\lambda_* = \mu_n$. *Let* $d = \dim(\mathcal{N}(A+\lambda_* B))$ *and* $V := [v_1, \cdots, v_d]$ *be a basis of* $\mathcal{N}(A+\lambda_* B)$ *that is* $B$*-orthogonal, i.e.,* $V^\top BV = I_d$.

*For an arbitrary $\alpha > 0$, define*

$$H := (A + \lambda_* B + \alpha \sum_{i=1}^{d} B v_i v_i^\top B).$$

*Then $H$ is positive definite. Moreover, $q := -H^{-1}g$ is the minimum-norm solution to the linear system $(A + \lambda_* B)p = -g$ in the $B$-norm, that is,*

$$q = \operatorname{argmin}_p \{ \|p\|_B | (A + \lambda_* B)p = -g \}. \tag{4.2}$$

*Furthermore, for any nonzero $v \in \mathcal{N}(A + \lambda_* B)$ there exists a scalar $\eta \in \mathbb{R}$ such that $p_* = q + \eta v$ is a TRS solution.*

*Proof.* First, we prove that $H \succ 0$. Let $W$ be the matrix that simultaneously diagonalize $A, B$ as in (2.3). Then $W^\top (A+\lambda_* B)W = \operatorname{diag}(\lambda_* - \mu_1, \cdots, \lambda_* - \mu_{n-d}, 0, \cdots, 0)$ is a diagonal matrix and $W^\top B W = I$.

We claim that defining $G = \sum_{i=1}^{d} B v_i (B v_i)^\top = B V V^\top B$, we have

$$W^\top G W = \begin{bmatrix} O_{n-d} & \\ & I_d \end{bmatrix}.$$

To see this, writing $W = [W_1 \ W_2]$ we have $(A + \lambda_* B)W_2 = (A + \lambda_* B)V = O_{n\times d}$ and $W_2^\top B W_2 = V^\top B V = I_d$. Since $V, W_2$ are of the same size, these two equalities imply that $V, W_2$ are both $B$-orthonormal matrices that span the same subspace, namely the space spanned by the eigenvectors of $A + \lambda B$ corresponding to the eigenvalue $\lambda_*$. Hence there exists an orthogonal matrix $Q \in \mathbb{R}^{d\times d}$ such that $V = W_2 Q$. Therefore,

$$\begin{aligned} W^\top (B V V^\top B) W &= [W_1 \ W_2]^\top B W_2 Q (W_2 Q)^\top B [W_1 \ W_2] \\ &= \left( [W_1 \ W_2]^\top B W_2 \right) W_2^\top B [W_1 \ W_2] \\ &= \begin{bmatrix} O_{(n-d)\times d} \\ I_d \end{bmatrix} \begin{bmatrix} O_{d\times(n-d)} & I_d \end{bmatrix} = \begin{bmatrix} O_{n-d} & \\ & I_d \end{bmatrix}, \end{aligned}$$

where we have used the fact $W^\top B W = [W_1 \ W_2]^\top B [W_1 \ W_2] = I_n$. Therefore,

$$\begin{aligned} W^\top H W &= (W^\top A W + \lambda_* I_n + \alpha \sum_{i=1}^{d} W^\top B v_i v_i^\top B W) \\ &= \operatorname{diag}(\lambda_* - \mu_1, \cdots, \lambda_* - \mu_{n-d}, \alpha, \cdots, \alpha) \end{aligned}$$

is positive definite, so by Sylvester's law of inertia it follows that $H$ is also positive definite.

We next show that $q := -H^{-1}g$ is a solution to the singular linear system $(A + \lambda_* B)p = -g$, which necessarily has infinitely many solutions. To see that $(A+\lambda_* B)q = -g$, define $\tilde{\Lambda} = -\operatorname{diag}(\mu_1, \ldots, \mu_{n-d})$ (recall $\Lambda$ in (2.3)) and note that $-g = (A+\lambda_* B)p$ implies (for the remainder of the proof, $I = I_{n-d}$ and $O = O_d$)

$$-g = W^{-\top} \begin{bmatrix} \tilde{\Lambda} + \lambda_* I & \\ & O \end{bmatrix} W^{-1} p,$$

and also that

$$q := -H^{-1} g = -W \begin{bmatrix} \tilde{\Lambda} + \lambda_* I & \\ & \alpha I \end{bmatrix}^{-1} W^\top g \tag{4.3}$$

from which we obtain

$$\begin{aligned}(A+\lambda_* B)q &= -W^{-\top}\begin{bmatrix}\tilde{\Lambda}+\lambda_* I & \\ & O\end{bmatrix}W^{-1}W\begin{bmatrix}\tilde{\Lambda}+\lambda_* I & \\ & \alpha I\end{bmatrix}^{-1}W^{\top}g\\ &= -W^{-\top}\begin{bmatrix}I & \\ & O\end{bmatrix}W^{\top}g = W^{-\top}\begin{bmatrix}I & \\ & O\end{bmatrix}W^{\top}W^{-\top}\begin{bmatrix}\tilde{\Lambda}+\lambda_* I & \\ & O\end{bmatrix}W^{-1}p\\ &= W^{-\top}\begin{bmatrix}\tilde{\Lambda}+\lambda_* I & \\ & O\end{bmatrix}W^{-1}p = -g.\end{aligned}$$

To prove that $q = -H^{-1}g$ is the minimum $B$-norm solution to $(A+\lambda_* B)p = -g$, note that we can write a general solution to $(A+\lambda_* B)p = -g$ as $p = q+v$ where $v \in \mathcal{N}(A+\lambda_* B)$. We shall show that $\|q\|_B \le \|q+v\|_B$ for any such $v$. We have

$$\|q+v\|_B^2 = (q+v)^\top B(q+v) = q^\top Bq + 2v^\top Bq + v^\top Bv.$$

Since $B$ is positive definite the third term is nonnegative, it suffices to show that $v^\top Bq = 0$ for all $v \in \mathcal{N}(A+\lambda_* B)$. To see this, using $q = -H^{-1}g = H^{-1}(A+\lambda_* B)q$ we obtain $Bq = BH^{-1}(A+\lambda_* B)q$, and note that

$$BH^{-1}(A+\lambda_* B) = (A+\lambda_* B)H^{-1}B,$$

which we can also verify using the decomposition with respect to $W$ (essentially because diagonal matrices commute):

$$\begin{aligned}BH^{-1}(A+\lambda_* B) &= W^{-\top}W^{-1}\left(W\begin{bmatrix}\tilde{\Lambda}+\lambda_* I & \\ & \alpha I_d\end{bmatrix}^{-1}W^{\top}\right)\left(W^{-\top}\begin{bmatrix}\tilde{\Lambda}+\lambda_* I & \\ & O\end{bmatrix}W^{-1}\right)\\ &= W^{-\top}\begin{bmatrix}I & \\ & O\end{bmatrix}W^{-1}\\ &= \left(W^{-\top}\begin{bmatrix}\tilde{\Lambda}+\lambda_* I & \\ & O\end{bmatrix}W^{-1}\right)\left(W\begin{bmatrix}\tilde{\Lambda}+\lambda_* I & \\ & \alpha I_d\end{bmatrix}^{-1}W^{\top}\right)W^{-\top}W^{-1}\\ &= (A+\lambda_* B)H^{-1}B.\end{aligned}$$

Therefore we obtain for all $v \in \mathcal{N}(A+\lambda_* B)$

$$v^\top Bq = v^\top(BH^{-1}(A+\lambda_* B)q) = v^\top((A+\lambda_* B)H^{-1}Bq) = (v^\top(A+\lambda_* B))H^{-1}Bq = 0,$$

as required. Note from (4.3) that $\|q\|_B^2 = \sum_{j=1}^{n-1}\frac{(w_j^\top g)^2}{(\mu_n-\mu_j)^2}$, which we used in the proof of Theorem 3.2.

It remains to establish the final statement. Since in the hard case the solution is on the boundary with Lagrange multiplier $\lambda_*$ equal to $\mu_n$, we are able to obtain a vector $q+\eta v$ on the boundary by finding the scalar $\eta$ by solving a scalar quadratic equation $\|q+\eta v\|_B^2 = \Delta^2$, to obtain a global solution to the TRS satisfying (1.2)–(1.5). □

In the last paragraph of the proof, it may appear that $\|q\|_B > \Delta$ is possible, in which case there is no $\eta$ such that $q+\eta v$ is on the boundary. However, by Theorem 1.1, there must be a boundary solution in the hard case. In other words, if even the minimum-norm solution had norm $\|q\|_B > \Delta$ then this implies the situation was not the hard case, and the process in the previous section solves the TRS.

The proof above shows that in the hard case the TRS solution $p_*$ is generally not unique; the goal is to find one solution, which we denote simply by $p_*$.

Theorem 4.3 shows that $p_*$ can be computed by finding the null vectors of $A+\lambda_* B$, forming $H$, and solving the nonsingular linear system $Hx+g=0$. Finding $\eta$ is then an easy task of solving a scalar quadratic equation. Note that due to the low-rank term $\alpha\sum_{i=1}^{d} Bv_iv_i^\top B$, $H$ can be dense even when $A, B$ are sparse. Nonetheless, the linear system $Hx+g=0$ can be solved efficiently by the CG algorithm (employing an appropriate preconditioner if available), which (as with any Krylov-type algorithm) only requires a routine for computing matrix-vector multiplications with respect to the coefficient matrix.

**4.2. Detecting the hard case in our algorithm.** As noted after Proposition 4.1, the hard case is indicated by $y_1 = 0$, that is, the vector $y_1$ that we usually obtain the solution from becomes zero; indeed, otherwise we obtain the solution by (3.10).

This suggests that we can detect the hard case by looking at the first half elements $y_1$ of the computed eigenvector $y = \left[\begin{smallmatrix} y_1 \\ y_2 \end{smallmatrix}\right]$, which we take to have unit norm $\|y\| = 1$: the hard case is when $y_1 = 0$. In practice, due to roundoff errors the computed vector $\hat{y}_1$ (recall that quantities wearing a hat represent computed approximations) has nonzero but small elements in the (near) hard case. To distinguish the hard and generic cases, we set a threshold $\tau < 1$ such that TRS with $\|y_1\| < \tau$ is treated as hard case. To set an appropriate value for $\tau$, note that if the top part $\hat{y}_1$ is small in norm, denoting it by $\epsilon$, we have

$$\tilde{M}(\hat{\lambda}_*)\begin{bmatrix}\epsilon \\ \hat{y}_2\end{bmatrix} = \begin{bmatrix} -B & A+\hat{\lambda}_*B \\ A+\hat{\lambda}_*B & -\frac{gg^\top}{\Delta^2}\end{bmatrix}\begin{bmatrix}\epsilon \\ \hat{y}_2\end{bmatrix}. \tag{4.4}$$

For this to be numerically zero, by the first block row we have

$$(A+\hat{\lambda}_*B)\hat{y}_2 = B\epsilon.$$

Since the right-hand side is $\mathcal{O}(\epsilon)$, it follows that $(\hat{\lambda}_*, \hat{y}_2)$ can be regarded as an approximate eigenpair for $(A, B)$.

To choose an appropriate threshold we analyze the accuracy of the computed $\hat{y}_1$ as an approximation to $p_*$. Let the computed eigenvector $\hat{y} = \left[\begin{smallmatrix} \hat{y}_1 \\ \hat{y}_2 \end{smallmatrix}\right]$ be normalized to have unit norm. Assuming the matrix of eigenvectors is well-conditioned, the accuracy of $\hat{y}$ is known to be $\mathcal{O}(\frac{\text{residual}}{\text{gap}})$ [37, Ch. 5]. Here the residual is $\|M(\hat{\lambda}_*)\hat{y}\|$, which is generally $\mathcal{O}(u)$ with a numerically stable algorithm[1], where $u$ is the unit roundoff, and gap is the smallest distance between $\lambda_*$ and the rest of the eigenvalues of $M(\lambda)$. Moreover, the loss of accuracy in extracting a vector of norm $\|y_1\|$ as a part of a unit-norm vector is a factor $\mathcal{O}(\frac{1}{\|y_1\|})$. Since the solution $p_*$ is obtained by normalizing $y_1$ as in (3.10), overall the accuracy in $p_*$ is estimated to be $\mathcal{O}(\frac{u}{\|y_1\|\text{gap}})$.

On the other hand, if we treat the problem as the hard case, recalling Theorem 4.3, we need to compute the null vectors $\mathcal{N}(A+\hat{\lambda}_*B)$. Numerically these are the vectors $v$ for which $\|(A+\hat{\lambda}_*B)v\|$ is negligible. With the tolerance $\tau$ for detecting the hard case, recalling (4.4), we expect the vectors we consider will have $\|(A+\hat{\lambda}_*B)v\| = \mathcal{O}(\|y_1\|)$, suggesting this entails an error of size $\mathcal{O}(\|y_1\|)$.

[1]To avoid unnecessary cluttering we assume $\|A\|, \|B\|$ are $\mathcal{O}(1)$. This causes no loss of generality as we can scale the matrices without changing the TRS solution: $A \leftarrow c_1A$ by taking $g \leftarrow c_1 g$, and $B \leftarrow c_2B$ by taking $\Delta \leftarrow c_2\Delta$.

We suggest choosing the threshold $\tau$ for $\|y_1\|$ based on which is likely to give the more accurate solution, according to the above discussion: $\frac{u}{\tau \text{gap}} = \tau$, that is,

$$\tau = \sqrt{\frac{u}{\text{gap}}}. \tag{4.5}$$

In double precision arithmetic $u \approx 10^{-16}$, and we choose $\tau$ to be about $10^{-8}\sqrt{1/\text{gap}}$. We can estimate the gap by computing two rightmost eigenvalues of $M(\lambda)$ instead of one. Alternatively and more efficiently, since a rough estimate for gap would suffice, we can use the inverse power method for $M(\lambda) = M_0 + \lambda M_1$ by starting with a random vector $w$ that is $M_1$-orthogonal to the eigenvector $v$ corresponding to $\lambda_*$, and taking a few iterations of power method with the matrix $(M_0 + \lambda_* M_1)^{-1} M_1$, executed by solving the linear system $(M_0 + \lambda_* M_1) w_{\text{new}} = M_1 w$ for $w_{\text{new}}$. Here we solve the linear system only crudely using an iterative solver such as MINRES [28].

Below is the process to deal with the hard case TRS in pseudocode.

**Algorithm 4.1** Algorithm for hard-case TRS.

```
1: Compute the eigenvectors of $(A, B)$ corresponding to $\lambda_*$ (i.e., the null vectors of $A + \lambda_* B$).
2: Solve $Hq + g = 0$ for $q$ by the CG method.
3: Take an eigenvector $v \in \mathcal{N}(A + \lambda_* B)$ computed above, and find $\eta \in \mathbb{R}$ such that $\|q + \eta v\|_B = \Delta$ by a quadratic equation.
4: Return $q + \eta v$ as a candidate for the global TRS solution.
```

## 5. Pseudocode.

We summarize our TRS algorithm in pseudocode (Algorithm 5.1).

**Algorithm 5.1** Solve the TRS (1.1).

```
1: (Consider the case $\lambda_* = 0$) Solve $Ap_0 = -g$ by the CG algorithm, and keep $p_0$ if satisfies $\|p_0\|_B < \Delta$.
2: Compute the rightmost eigenvalue $\lambda_*$ of $\tilde{M}(\lambda)$ and an eigenvector $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ such that
   $$\begin{bmatrix} -B & A \\ A & -\frac{gg^\top}{\Delta^2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -\lambda_* \begin{bmatrix} 0 & B \\ B & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}. \tag{5.1}$$
3: Estimate the gap between $\lambda_*$ and the other eigenvalues via a power method. If $\|y_1\| \le \sqrt{\frac{u}{\text{gap}}}$, then treat as hard case: run Algorithm 4.1 to obtain $p_1$. Otherwise, obtain $p_1$ by $p_1 := -\text{sign}(g^\top y_2)\Delta \frac{y_1}{\|y_1\|_B}$.
4: The solution $p_*$ is either $p_1$ or $p_0$ (if it exists), whichever gives the smallest objective value.
```

The core part of the algorithm is (5.1). We note that some steps can be skipped if additional information is available. For example, we can conclude that $p_0$ is the (unique) optimal TRS solution if (i) it is feasible $\|p_0\|_B \le \Delta$, and (ii) $A \succ 0$. Therefore, when the positive definiteness of $A$ is known or easily verifiable, if $\|p_0\|_B \le \Delta$ after step 1 in Algorithm 5.1, then we can dismiss the remaining steps in Algorithm 5.1 and take $p_0$ as the TRS solution.

Note that the CG algorithm is originally designed for positive definite linear systems, and if CG does not converge then this implies that $A$ is numerically indefinite. However, in practice CG often converges even when $A$ is indefinite [27], so we cannot conclude that $A$ is positive definite just because the linear system $Ap_0 = -g$ was solved by CG.

Another situation worth mentioning is when $A$ is known (or easily verifiable) to be indefinite with one or more negative eigenvalues. In this case a TRS solution must lie on the boundary, and so there is no point in executing step 1; we directly proceed to step 2.

## 6. Comparison with existing methods.

Here we compare our Algorithm 5.1 with previous methods and argue that ours has attractive properties in terms of efficiency and simplicity. Moreover, numerical experiments illustrate the excellent accuracy as we see in Section 7.

### 6.1. Efficiency.

The computational cost of our algorithm is $\mathcal{O}(n^3)$ flops for dense $A, B$. When $A, B$ are large and sparse, it is essentially the cost of an Arnoldi-type method for computing one rightmost eigenpair, which is typically in the order of a constant times the cost of a matrix-vector product, or a shifted-and-inverted linear system $(M_0 + \sigma M_1)x = b$ for some $\sigma \in \mathbb{C}$; the cost also depends on the separation of eigenvalues etc. This is in the same ballpark as the cost of the existing algorithms [13, 25, 35] when $B = I$, both in the dense and large-sparse cases.

Let us compare the cost in more detail. Recall that most conventional algorithms for an accurate TRS solution involve $n \times n$ linear systems or eigenvalue problems iteratively, whereas ours solves a double-sized $2n \times 2n$ eigenproblem once. The advantage of a noniterative approach becomes significant especially when the eigenproblem scales mildly with $n$. For example, in the dense case where eigensolvers (and linear systems) require $\mathcal{O}(n^3)$ cost, our approach is roughly comparable to a conventional algorithm that iterates 8 times. If the eigensolver needs $\mathcal{O}(n^p)$ cost with $p < 3$, then this number reduces to $2^p$.

We note that (when $B = I$) it is possibly less than twice as expensive to solve the $2n \times 2n$ eigenvalue problem $\tilde{M}$ rather than the size-$(n+1)$ eigenproblem with respect to the matrix $N(\alpha) = \left[\begin{smallmatrix} \alpha & g^\top \\ g & A \end{smallmatrix}\right]$, which is used in [35, 32, 33]. This is because the dominant cost in the Arnoldi iteration is in matrix-vector multiplication, and multiplying the matrix (3.11) to a vector $\left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]$ can be done by computing $A[x_1\ x_2]$ and a vector-vector multiplication $g^\top x_1$. Now, computing $A[x_1, x_2]$ is usually faster than two independent matrix-vector multiplications $Ax_1, Ax_2$ due to the use of a higher level blas routine [12].

### 6.2. Ease of implementation.

As mentioned before, the main feature of our approach is that the TRS can be solved essentially by one generalized eigenvalue problem.

Besides the aesthetic advantage of directly giving a solution without iterations, another advantage of our approach is its ease of implementation. For example, in MATLAB we can execute (5.1), the main part of Algorithm 5.1, in just four lines (here $\tilde{M}(\lambda) = M_0 + \lambda M_1$):

```
M0 = [-B  A;A -g*g'/Delta^2];
M1 = [zeros(n) B;B zeros(n)];
lam = max(eig(M0,-M1));
p = -(A+lam*B)\g;
```

This is strikingly simple when compared with those of the existing algorithms, such as the codes of [9, 14, 15, 33]. When $A, B$ are large-scale and sparse, it is advised to replace `eig` with `eigs` and $\tilde{M}$ with $M$. Specifically, in the large-scale case, after the second line the code should be

```
[v,lam] = eigs(@(x)M0x(x),2*n,-M1,1,'lr');
p = v(n+1:end);
p = p/sqrt(p'*B*p)*Delta;
```

The last line is a normalization to force $\|p\|_B = \Delta$, because the computed eigenvector is normalized so that $\|v\|_2 = 1$. Here `M0x(x)` is a function handle that left-multiplies $M_0$ to an input vector `x`. This saves memory over storing the matrix $M_0$ because this way we essentially only need to store the matrices $A$ and $B$, along with the vectors $g, v$. This way, despite their doubled size, storing $M, \tilde{M}$ requires no more memory than $A, B$ and $g$.

**6.3. TRS as a subproblem.** We argue that Algorithm 5.1 is the first algorithm that solve the TRS accurately and is suited to the large-scale sparse case with $B \neq I$ without requiring a change of variables or outer iterations. However, its performance for efficiently solving the overall nonlinear optimization problem using TRS as a subproblem is not easily predictable, in which solving the TRS exactly is not necessary, and it suffices if the TRS solution results in sufficient reduction in the original nonlinear function $f$.

To gain an idea of the performance in such cases, we present experiments where we obtain approximate TRS solutions by stopping the Arnoldi iteration early when computing the eigenpair (5.1).

## 7. Numerical experiments.

We now turn to experiments to examine the performance of the proposed methods for a variety of TRS instances. All experiments were carried out in MATLAB 2013A on a Blade server machine with Xeon CPU and 64 GB memory.

In the figures below, Algorithm 5.1 is shown as GEP (standing for generalized eigenvalue problem). We compare GEP with three other codes that are publically available, implementing existing algorithms: (i) the code by Fortin-Wolkowicz [9] which is based on the algorithm by Rendl and Wolkowicz [30] (shown as FRW), (ii) Rojas, Santos and Sorensen [33], based on [35] (shown as RSS), and (iii) the `galahad_gltr` in the Galahad library [14] (shown as GLTR), based on the algorithm by Gould, Lucidi, Roma and Toint [15]; this code is written in Fortran, and here we used its MATLAB interface. We used the default parameters for FRW and RSS, and for GLTR, we used the options `extra_vectors=100` and `stop_relative=1e-15` (suggested by Nick Gould, chosen to obtain high-accuracy solutions).

We examine the performance in the following cases: (a) $B = I$ and $A$ is sparse, (b) $B = I$ and $A$ is dense, (c) $B \neq I$ and $A, B$ are sparse, and (d) the hard case with $B = I$.

In each set of experiments, we ran the codes 20 times and report the average runtime and accuracy. To measure the accuracy, we have computed the relative objective function difference as follows:

$$\frac{f(\hat{p}_*) - f(\hat{p}_{\text{best}})}{|f(\hat{p}_{\text{best}})|}. \tag{7.1}$$

Here $\hat{p}_*$ denotes the computed solution of each method and $\hat{p}_{\text{best}}$ is the solution with the smallest objective value among the four algorithms. The reason the accuracy measure is usually positive in the plots below (exceptions include Figure 7.1 (right) for

GLTR) is that the algorithm that achieves $\hat{p}_{\text{best}}$ varies from problem to problem, and we report the average of 20 runs. Throughout, unless otherwise specified, we take[2] $\varDelta = 1$ and the vector $g$ is randomly generated by `randn(n,1)`. To ensure the computed solutions are always feasible, after each algorithm we applied the normalization $\hat{p} := \varDelta\hat{p}/\|\hat{p}\|_B$ if the computed $\hat{p}$ violated the constraint $\|\hat{p}\|_B \le \varDelta$.

*When $B = I$ and $A$ is sparse.* We first set $B = I$ and let $A$ be large-sparse matrices, varying the matrix size $n$ from $10^3$ (moderate) to $10^7$ (large).

In Figure 7.1 we show the results with $A = L - 5I$, where $L$ is the discrete 2D Laplacian (Figure 7.1); this follows the test matrices in [33]. For $n > 10^6$ we ran only GEP and GLTR, which complete the computation in a reasonable amount of time.

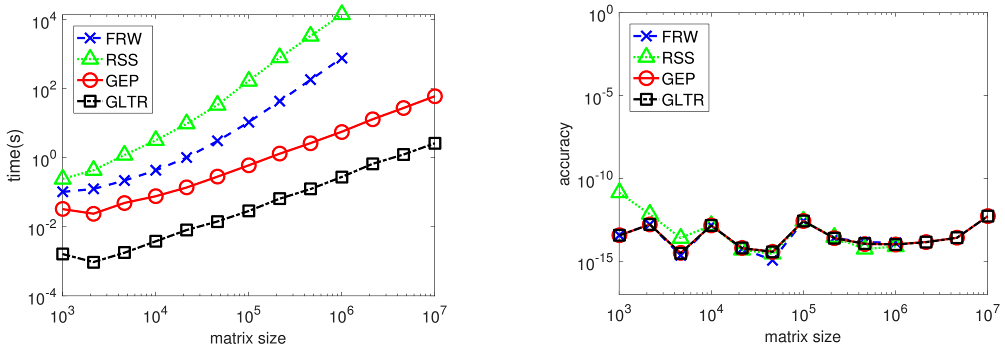

Fig. 7.1. *Runtime (left) and accuracy (right) for $A = L - 5I$ and $B = I$.*

We also experiment with a random symmetric sparse matrix (Figure 7.2), generated by

$$\texttt{A} = \texttt{sprandsym}(\texttt{n}, \texttt{density}), \tag{7.2}$$

where we took `density=1e-2`.

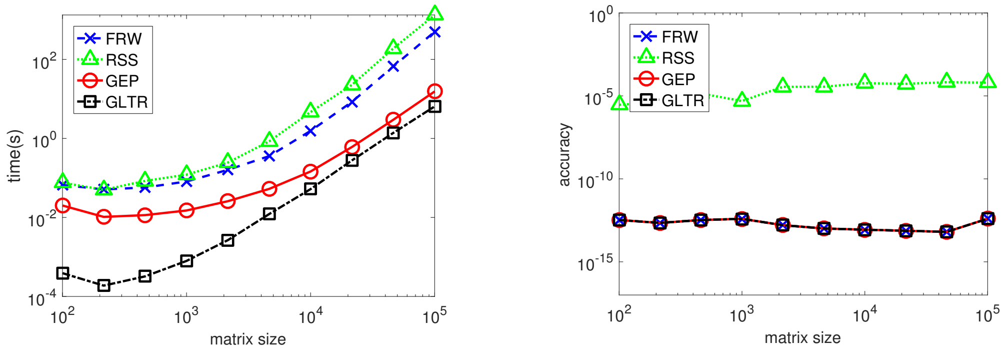

Fig. 7.2. *Runtime (left) and accuracy (right) for $\texttt{A} = \texttt{sprandsym}(\texttt{n}, \texttt{1e}-2)$ and $B = I$.*

The left figures show the runtime in seconds. We see that while GLTR was the fastest in these examples, our algorithm was notably faster than FRW and RSS, the

[2]The choice of $\varDelta$ is usually made adaptively during the overall trust-region method [4, Sec.6.1]. It is worth noting that the volume of the unit ball in $\mathbb{R}^n$ shrinks factorially with $n$ [18], so a geometrically appropriate choice may be far from $\varDelta = 1$. Our experiments suggest that the performance of GEP dependent only mildly on the value of $\varDelta$.

codes natively programmed in MATLAB. Figure 7.1 indicates that GEP scales like $\mathcal{O}(n)$ for this problem. Recalling the discussion in Section 6.1, this is favorable for our non-iterative algorithm, because the iterative algorithms often need to solve more than 10 eigenproblems of half the size.

The right plots of Figures 7.1 and 7.2, which show the difference in the objective values, illustrate that our algorithm GEP obtained solutions within about $10^{-15}$ of the optimal for every problem, which are accurate enough to be regarded as exact solutions in double precision arithmetic. These results suggest that with the eigensolvers available today, when $A, B$ are sparse, our algorithm is applicable and effective for very large problems.

*When $B = I$ and $A$ is dense.* We next examine the dense case. We generate $A$ by forming $n$ random real numbers $\mu_i$ via `randn(n,1)`, then generating a random orthogonal matrix $Q$, formed by `orth(randn(n))` and setting $A = Q^\top \mathrm{diag}(\mu_i)Q$. Clearly we are limited to much smaller matrix size $n$ than in the previous sparse case; here we take $n \leq 10^4$.

The results are shown in Figure 7.3. The accuracy behaved much the same as in the sparse case. For the speed, the difference is more benign than in the sparse case. We can explain this qualitatively as follows: in the dense case all the algorithms require $\mathcal{O}(n^3)$ operations, and recalling the discussion in Section 6, our algorithm is expected to be fast especially when the generalized eigenproblem can be solved efficiently. In the sparse case its cost is often much less than $\mathcal{O}(n^3)$ as we saw above, and this is a favorable situation for our algorithm. In the dense case, for which all algorithms are expected to scale like $\mathcal{O}(n^3)$, our algorithm still has efficiency comparable with other approaches. GLTR appears to scale slightly worse than the other algorithms here. Another alternative for dense problems is GALAHAD's `galahad_trs`, which implements the algorithm in [15], although it did not perform better in the experiments here.

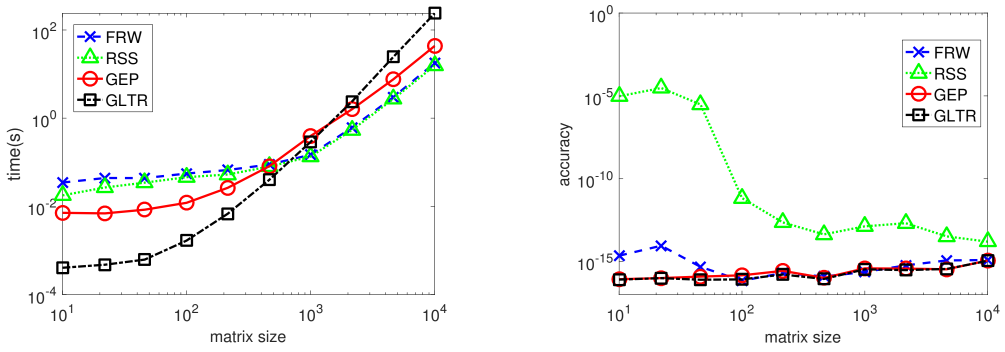

FIG. 7.3. *Dense $A$, $B = I$. Runtime (left) and accuracy (right).*

*When $B \succ 0$ and $A$ is sparse.* We now examine problems with $B \neq I$. We let $A$ be sparse as in and $B$ be a tridiagonal matrix, defined as `A = sprandsym(n,0.01)` as in (7.2) and `B = tridiag(1,3,1)`. $g$ is a random vector as before.

Since FRW and RSS are not directly applicable when $B \neq I$, to invoke these algorithms we first compute the Cholesky factorization $B = LL^\top$, then form $L^{-1}AL^{-\top}$ and apply the codes to $A \leftarrow L^{-1}AL^{-\top}, g \leftarrow L^{-1}g$ and $B \leftarrow I$. RSS is a matrix-free algorithm that allows the input for the matrix $A$ to be a routine for matrix-vector multiplication, so we can use RSS without forming the (dense) matrix $L^{-1}AL^{-\top}$ by solving a linear system with $L, L^\top$ for each matrix-vector multiply, thus allows us to work only with sparse matrices; in the above example, $L$ here has bandwidth 2. The code FRW does not allow such input, which makes the code too costly in memory and hence inapplicable for large $n$. Our algorithm GEP and GLTR do not require modified treatment when $B \neq I$; they handle the $B \neq I$ case exactly the same way as when $B = I$.

Figure 7.4 shows the results. As before, our algorithm ourperforms FRW and RSS. GLTR is the most efficient here, although the difference shrinks with $n$ as GEP appears to scale better.

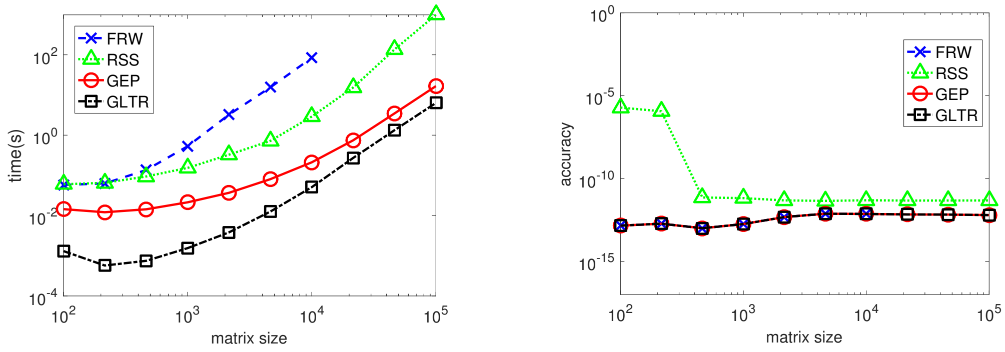

FIG. 7.4. *Sparse $A, B$, $B \neq I$. Runtime (left) and accuracy (right) for* `A = sprandsym`*, $B$ tridiagonal.*

*Hard case.* We now turn to the hard case. The standard process is then insufficient, and algorithms usually employ a remedy for the hard case, such as Algorithm 4.1 for GEP. An exception is RSS, in which the hard case is treated within the main iteration of the algorithm.

We let $B = I$ and form a tridiagonal $A$, with 2 on the diagonal and random $N(0, 1)$ elements on the off-diagonals; the Hessian $A$ is tridiagonal if the objective function in the nonlinear optimization problem depends only on adjacent variables $x_{i-1}, x_i, x_{i+1}$. To generate a "hard case", we first set $g$ to be a random vector and calculate the largest eigenvalue $\lambda_*$ of the pencil $A + \lambda B$ with a corresponding eigenvector $v$. We then update $g$ as $g \leftarrow g - (v^\top g/||v||^2)v$. To enforce the hard case we set $\Delta$ to be large: $\Delta = 10^3$. Indeed we verified that $\lambda_* = \mu_n$ in all the examples thus generated indicating the hard case.

We set the convergence criteria of the CG method as follows: the relative residual is less than $10^{-12}$, or the maximum allowed number of iterations 3000 is reached. Since `galahad_gltr` appeared to return an incorrect output in the hard case, here we do not present its accuracy. The results are shown in Figure 7.5.

Observe that especially in the hard case, our algorithm is fast and reliable, giving the most accurate solutions. We note that while GALAHAD's GLTR underperformed in this example, the latest pre-released version 3.0 has addressed the inefficiency and accuracy issue for the hard case.

*Problems whose exact solution is known.* We can generate a TRS problem in the hard case with known exact solutions as follows: first, set

$$A = \mathrm{diag}(-1, 2, 3, \cdots, n), \quad g = (0, -3\alpha\Delta, 0, \cdots, 0)^\top,$$

$B = I$, $\Delta = 1$ and $\alpha = 10^{-2}$. In this case, the optimal value of TRS (1.1) is $-(1+3\alpha^2)\Delta^2/2$. We then generate an orthogonal matrix $Q$ by the MATLAB command

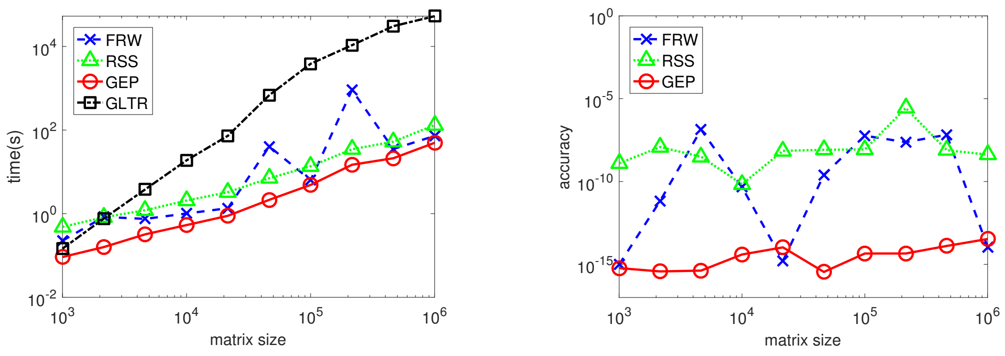

FIG. 7.5. *Hard case. Runtime (left) and accuracy (right), $B = I$ and $A$ sparse.*

TABLE 7.1
*Error in computed objective value.*

| | FRW | RSS | GEP |
|---|---|---|---|
| $n = 100$ | 1.50e-12 | 9.37e-14 | 1.44e-15 |
| $n = 1000$ | 1.49e-12 | 8.90e-14 | -6.22e-15 |
| $n = 10000$ | 4.39e-14 | 8.32e-14 | 3.87e-14 |

`qr(rand(n))` and update $A, g$ as $A \leftarrow QAQ^\top$, $g \leftarrow Qg$, which does not change the optimal objective value.

We measured the difference between the computed optimal value of each method and the exact optimal value. Note that we sometimes obtained objective value slightly smaller than the exact optimal value; this is caused by roundoff error (recall that we impose that the obtained solution is feasible to $O(u)$). The result is shown in Table 7.1, which confirms that our algorithm indeed computes solutions with high accuracy, and the previous accuracy plots give reliable estimates for the errors.

*Approximate solutions.* Recalling the discussion in Section 6.3, here we consider using our approach to efficiently obtain an approximate TRS solution by solving the generalized eigenvalue problem (5.1) approximately instead of to high accuracy $\mathcal{O}(u)$. A natural strategy would be to terminate the iteration for computing the eigenpair before convergence to full precision is attained.

To illustrate this idea, we formed an $n = 1000$ example and took $B = I$ and let $A$ be a random matrix defined by `A=randn(n);A=A'+A`, and we apply $k$ steps of the Arnoldi process [1, Ch. 7] to $-M_1^{-1}M_0 = \begin{bmatrix} -A & \frac{gg^\top}{\Delta^2} \\ B & -A \end{bmatrix}$ and approximate the eigenvector in (5.1) by the Ritz vector (e.g. [1, Ch. 3]) corresponding to the rightmost Ritz value, from which we obtain approximate solutions $x_k$ for each $k$.

Figure 7.6 shows the accuracy of the resulting approximate solutions $p_k$ as the dimension of the Arnoldi subspace $k$ varies. Here we quantify the accuracy from three aspects: denoting by $p_*$ the solution (obtained by Algorithm 5.1) the angular error $\angle(p_k, p_*) = \arccos \frac{|p_k^\top p_*|}{\|p_k\|\|p_*\|}$, the distance from the optimal objective value $f(p_k) - f(p_*)$, and the eigenpair residual $\|M_0x - \lambda M_1x\|_2$, which measures the quality of the approximate eigenpair.

Observe in Figure 7.6 that the computed TRS solution $p_k$ improves its accuracy as the quality of the Ritz vector improves. Since Ritz vectors are known to typically converge geometrically, which we can see in the figure, this suggests that our approach

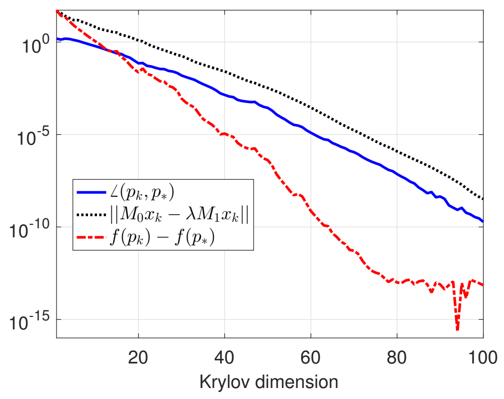

FIG. 7.6. *Accuracy in solution $\angle(p_k, p_*)$ and objective value $f(p_k) - f(p_*)$ as Krylov subspace dimension $k$ varies. Also shown is the residual $\|M_0x - \lambda M_1x\|_2$ of the approximate eigenpair.*

is attractive also for computing an approximate TRS solution.

**Summary of experiments.** The results of our experiments can be summarized as follows.

- Algorithm 5.1 (GEP) based on one generalized eigenproblem is consistently reliable and its accuracy is often among the highest, including the hard case.
- Algorithm 5.1 has speed comparable to the state-of-the-art methods, largely independent of the problem: dense/sparse and/or $B \neq I$.

While GLTR was often the most efficient in the non-hard case, our algorithm is the simplest to implement (largely due to the sophisticated but black-box eigensolvers available via `eig` and `eigs`), robust and reliable without being much slower, tends to scale well as $n$ grows and outperforms existing MATLAB implementations.

## 8. Discussion.

As we have seen, the real eigenvalues of $M(\lambda)$ correspond to the KKT points for the TRS, and the largest eigenvalue provides a solution that minimizes the objective function $g^\top p + \frac{1}{2}p^\top Ap$. In fact, more can be said: as shown by Forsythe and Golub [7], the objective function value is an increasing function of $\lambda$, so we can also *maximize* the objective function by finding the *smallest* real eigenvalue of $M(\lambda)$. Further analysis of the KKT points is given in [21]. The fact that we can both maximize and minimize the objective function is perhaps unsurprising as we impose no positive definiteness assumption on $A$, so the objective function is nonconvex and there is no fundamental difference between minimizing and maximizing it.

Future directions include performance benchmarking on parallel systems in the context of solving the TRS approximately in the trust-region method for nonlinear optimization problems, comparing in particular with [13, 36]. Also worth considering are extending the eigenvalue-based approach to solve other trust-region type problems [3, 29], and dealing with a general QCQP with one constraint. We note that an eigenvalue-based algorithm for QCQP with two constraints is developed in [34], which also mentions in its appendix an algorithm for one constraint. However, that algorithm involves the computation of all the eigenvalues, and thus requires $\mathcal{O}(n^3)$ flops in all cases. It remains open to develop a more efficient algorithm that exploits structure such as sparsity.

**Acknowledgement.** We are grateful to Nick Gould for providing comments on the manuscript and project, and helping us with the Galahad library. We thank the referees for their useful comments and suggestions. We thank Bill Hager and Henry Wolkowicz for discussions and testing our codes.

## REFERENCES

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, PA, USA, 2000.

[2] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] S. Burer and K. M. Anstreicher. Second-order-cone constraints for extended trust-region subproblems. *SIAM J. Optim.*, 23(1):432–451, 2013.

[4] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust Region Methods*, volume 1. SIAM, Philadelphia, PA, USA, 2000.

[5] J. B. Erway and P. E. Gill. A subspace minimization method for the trust-region step. *SIAM J. Optim.*, 20(3):1439–1461, 2009.

[6] J. B. Erway, P. E. Gill, and J. D. Griffin. Iterative methods for finding a trust-region step. *SIAM J. Optim.*, 20(2):1110–1131, 2009.

[7] G. E. Forsythe and G. H. Golub. On the stationary values of a second-degree polynomial on the unit sphere. *J. SIAM*, 13(4):1050–1068, 1965.

[8] C. Fortin and H. Wolkowicz. The trust region subproblem and semidefinite programming. *Optimization Methods and Software*, 19(1):41–67, 2004.

[9] C. Fortin and H. Wolkowicz. Trust region subroutine algorithm: Algorithm and Documentation, 2010. `http://www.math.uwaterloo.ca/~hwolkowi/henry/software/trustreg.d`.

[10] W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra Appl.*, 114:815–839, 1989.

[11] D. M. Gay. Computing optimal locally constrained steps. *SIAM J. Sci. Stat. Comp.*, 2(2):186–197, 1981.

[12] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 4th edition, 2012.

[13] N. I. M. Gould, S. Lucidi, M. Roma, and P. L. Toint. Solving the trust-region subproblem using the Lanczos method. *SIAM J. Optim.*, 9(2):504–525, 1999.

[14] N. I. M. Gould, D. Orban, and P. L. Toint. GALAHAD, a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. *ACM Trans. Math. Soft.*, 29(4):353–372, 2003.

[15] N. I. M. Gould, D. P. Robinson, and H. S. Thorne. On solving trust-region and other regularised subproblems in optimization. *Mathematical Programming Computation*, 2(1):21–57, 2010.

[16] W. W. Hager. Minimizing a quadratic over a sphere. *SIAM J. Optim.*, 12(1):188–208, 2001.

[17] N. J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, Philadelphia, PA, USA, 2008.

[18] G. Huber. Gamma function derivation of n-sphere volumes. *Amer. Math. Monthly*, 89(5):301–302, 1982.

[19] S. Iwata, Y. Nakatsukasa, and A. Takeda. Global optimization methods for extended Fisher discriminant analysis. In *Proc. Seventh AISTATS, JMLR W&CP 33*, pages 411–419, 2014.

[20] R. Lehoucq, D. Sorensen, and C. Yang. *Arpack User's Guide: Solution of Large-Scale Eigenvalue Problems With Implicitly Restorted Arnoldi Methods*, volume 6. SIAM, 1998.

[21] S. Lucidi, L. Palagi, and M. Roma. On some properties of quadratic programs with a convex quadratic constraint. *SIAM J. Optim.*, 8(1):105–122, 1998.

[22] J. M. Martínez. Local minimizers of quadratic functions on Euclidean balls and spheres. *SIAM J. Optim.*, 4(1):159–176, 1994.

[23] J. J. Moré. *Recent developments in algorithms and software for trust region methods*. Mathematical Programming: the State of the Art, Springer, 1983.

[24] J. J. Moré. Generalizations of the trust region problem. *Optimization methods and Software*, 2(3-4):189–209, 1993.

[25] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM J. Sci. Stat. Comp.*, 4(3):553–572, 1983.

[26] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer New York, 2nd edition, 1999.

[27] C. C. Paige, B. N. Parlett, and H. A. Van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numer. Lin. Alg. Appl.*, 2(2):115–133, 1995.

[28] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629, 1975.

[29] T. K. Pong and H. Wolkowicz. The generalized trust region subproblem. *Computational Optimization and Applications*, 58(2):273–322, 2014.

[30] F. Rendl and H. Wolkowicz. A semidefinite framework for trust region subproblems with applications to large scale minimization. *Math. Prog.*, 77(1):273–299, 1997.

[31] E. Rimon and S. P. Boyd. Obstacle collision detection using best ellipsoid fit. *Journal of Intelligent and Robotic Systems*, 18(2):105–126, 1997.

[32] M. Rojas, S. A. Santos, and D. C. Sorensen. A new matrix-free algorithm for the large-scale trust-region subproblem. *SIAM J. Optim.*, 11(3):611–646, 2001.

[33] M. Rojas, S. A. Santos, and D. C. Sorensen. Algorithm 873: LSTRS: MATLAB software for large-scale trust-region subproblems and regularization. *ACM Transactions on Mathematical Software*, 34(2):11:1–11:28, 2008.

[34] S. Sakaue, Y. Nakatsukasa, A. Takeda, and S. Iwata. Solving generalized CDT problems via two-parameter eigenvalues. *SIAM J. Optim.* to appear.

[35] D. C. Sorensen. Minimization of a large-scale quadratic function subject to a spherical constraint. *SIAM J. Optim.*, 7(1):141–161, 1997.

[36] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.*, 20(3):626–637, 1983.

[37] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory (Computer Science and Scientific Computing)*. Academic Press, 1990.

[38] P. D. Tao and L. T. H. An. A DC optimization algorithm for solving the trust-region subproblem. *SIAM J. Optim.*, 8(2):476–505, 1998.

[39] P. L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In I. Duff, editor, *Sparse Matrices and Their Uses*, pages 57–88. Academic Press, London, 1981.