Simultaneous iterative solutions for the trust-region and minimum eigenvalue subproblem

I.G. Akrotirianakis, M. Gratton, J.D. Griffin, S. Yektamaram, W. Zhou[a*]

[a] *SAS Institute, 100 SAS Campus Drive, Cary, NC 27513, USA.;*

Given the inability to foresee all possible scenarios, it is justified to desire an efficient trust-region subproblem solver capable of delivering any desired level of accuracy on demand; that is, the accuracy obtainable for a given trust-region subproblem should not be partially dependent on the problem itself. Current state-of-the-art iterative eigensolvers all fall into the class of restarted Lanczos methods; whereas, current iterative trust-region solvers at best reduce to unrestarted Lanczos methods; which in this context are well known to be numerically unstable with impractical memory requirements. In this paper, we present the first iterative trust region subproblem solver that at its core contains a robust and practical eigensolver. Our solver leverages the recently announced groundbreaking work of Stathopoulos and Orginos [24] which has not been noticed by the optimization community and can be utilized because, unlike other restarted Lanczos methods, its restarts do not necessarily modify the current Lanczos sequence generated by Conjugate Gradient methods (CG). This innovated strategy can be utilized in the context of TR solvers as well. Moreover, our TR subproblem solver adds negligible computational overhead compared to existing iterative TR approaches.

Keywords: nonlinear programming; nonconvex large scale problems; trust region methods; conjugate gradient method

AMS Subject Classification: 90C56, 90C30, 90C26, 65K05, 49M37, 49M30

1. Introduction

In this paper we present a restarted Lanczos algorithm for solving the trust-region subproblem (TRS)

$$\begin{array}{ll} \underset{s\in\mathbb{R}^n}{\text{minimize}} & \psi(s) = g^T s + \frac{1}{2} s^T H s, \\ \text{subject to} & \|s\|_2 \leq \Delta \end{array} \tag{1}$$

where H is a symmetric (possibly indefinite) matrix and g may be any vector (including the zero vector). We are primarily interested in the use of this algorithm in the context of large-scale nonconvex optimization, where g and H may denote the first and second derivatives of a corresponding merit function. The proposed algorithm comes equipped with tolerance parameters that permit the user to achieve fast approximate trust-region solutions reminiscent of the Steihaug-Toint algorithm [20, 25]. However, unlike the Steihaug-Toint algorithm, accuracy can be controlled even in the case where the trust-region solution lies on the boundary. This paper assumes that the reader has some

*Corresponding author. Email: wenwen.zhou@sas.com

familiarity with the conjugate gradient (CG) and Lanczos algorithms, the Steihaug-Toint exit strategy, sequential subspace methods (SSM), and basic properties of the trust-region subproblem solution [5].

Iterative Lanczos-based trust-region solvers can be divided into two categories:

(1) unrestarted Lanczos variants [11, 20, 25], and
(2) restarted Lanczos variants [7, 8, 13, 14, 21].

Here we use the term "unrestarted" to refer to algorithms that follow a single Lanczos factorization stream to the global solution. Restarted Lanczos approaches were first proposed as efficient and robust alternatives for computing extreme eigenvalues and their corresponding eigenvectors [3, 17, 26]. The motivation for these approaches was that unrestarted Lanczos suffers numerical instability and storage issues. Attempts to improve the stability of the unrestarted algorithm (such as the selective reorthogonalization approach of Parlett and Scott [19]) suffer from inefficiencies and a possibly prohibitive storage requirement.

More recently, restarted approaches have surfaced in the trust-region context under the banner of sequential subspace methods (SSM), first coined by Hager [14]. A minor contribution of this paper is the simple observation that SSMs have begun to employ strategies and ideas already fine-tuned by the developers of eigenvalue solvers; SSMs can be extended and improved simply by extending existing eigensolver algorithms to the trust-region context. We emphasize that the TRS "hard case" has an embedded eigenvalue problem. We also note that any efficient algorithm for solving (1) would necessarily be a minimum eigenvalue calculator for H, simply by setting g equal to 0 and shifting H appropriately. The case $g = 0$ can, in this context, be considered the ultimate hard case.

Because of the effectiveness of restarted Lanczos algorithms in the eigenvalue context, we advocate the use of a restarted Lanczos approach when solving the trust-region subproblem, with a single exception: the case where the solution lies strictly inside the trust region. In this case, algorithms such as CG can exploit a stable recurrence relation that provides a global minimizer of $\psi(s)$ over a nested subspace sequence. This subspace is never recomputed or stored, so the storage requirement is minimal. For this reason, our proposed algorithm, like those in [7, 8, 11, 13, 20, 25], reduces to CG when the solution is interior.

Once the boundary is encountered, unrestarted Lanczos algorithms such as GLTR [11] continue to use nested subspaces in the absence of an equivalent recurrence relation. These algorithms must either store or recompute the Lanczos vectors to determine the solution. Because of the instabilities of Lanczos and the need to either recompute (effectively doubling the number of matrix multiplies) or store large number of high-dimensional dense vectors, the authors of [11] advocate using their algorithm at most 5-20 matrix multiplies past the Steihaug-Toint exit point. Whenever the trust-region boundary is encountered, a heuristic is used to choose between two options, the first of which is to double the current number of matrix multiplies to form the current solution. The alternative is to discard the information gained from the 5-20 matrix multiplies and return to the Steihaug-Toint exit point.

Restarted algorithms such as those in [7, 8, 13, 14] do not suffer from the same computational issues. These methods are provably globally convergent at the cost of a single matrix-vector multiply per iteration if they can obtain a good estimate of the *minimum eigenvector*, which we define as

$$\arg\min_{v} \frac{v^T H v}{v^T v}. \tag{2}$$

Restarted algorithms need not distinguish between the hard case and the easy case, as both problems can be solved exactly in the reduced dimension with negligible computation. These algorithms do, however, have two primary weaknesses. The first weakness is that convergence rates for SSM can be poor without implementing complicated acceleration schemes. These acceleration schemes are costly in that they require the partial solution of a linear system at each iteration, so the number of matrix-vector multiplies per iteration varies between one and many. The second weakness stems from the need to compute the minimum eigenvector in order to guarantee global convergence. However, none of these approaches provide a viable means for computing this vector in practice.

In this paper we present a new algorithm, eigCG-TR, that resolves both of these weaknesses in an efficient and numerically stable way while requiring only a single matrix-vector multiply per iteration. We resolve the first weakness by slightly relaxing the trust-region radius in a manner similar to Moré and Sorensen [18]. This creates an annular region that enables the algorithm to correct itself without the use of costly acceleration schemes. The second weakness is resolved by incorporating the recent eigCG algorithm [24], which enables the accurate computation of eigenpairs while performing no more than simple book-keeping on top of an existing CG algorithm. Remarkably, eigCG has shown itself to be a robust and competitive eigensolver, despite following an unrestarted Lanczos stream (in the eigCG algorithm the subspaces are restarted, but the seed residual vector is not). This feature permits the trust-region algorithm to control if and when a restart should occur. In practice eigCG competes well with existing state-of-the-art matrix-free eigensolvers, but the proof of its convergence is still pending. Assuming that the eigCG algorithm is indeed globally convergent, we will show that the trust-region solver in this paper is likewise globally convergent, as discussed in [27].

2. Background and notation

A vector s^* is a *global* solution of the trust region subproblem if there exists a scalar σ^* such that

$$\begin{aligned} (H + \sigma^* I)s &= -g, \quad \sigma^*(\Delta - \|s\|) = 0, & (3)\\ (H + \sigma^* I) &\succeq 0, \quad \|s^*\| \leq \Delta, \quad \sigma^* \geq 0. & (4) \end{aligned}$$

Here σ^* denotes the corresponding Lagrange multiplier for the trust-region constraint $\|s\| \leq \Delta$. In the following section we will use an asterisk to denote the exact solution of a problem. The pair (s^*, σ^*) represents the global solution and corresponding multiplier for (1). The pair (v^*, λ^*) represents the global solution and corresponding eigenvalue for (2).

Two central components of the eigCG-TR method are the conjugate gradient algorithm (CG) and the sequential subspace method (SSM). In this section, we briefly describe how these methods are implemented in the context of the eigCG-TR algorithm.

Algorithm 1 defines a function that executes a single iteration of the CG (equivalently, Lanczos) algorithm, then returns book-keeping terms that are passed to other components of eigCG-TR. Note that in practice, Algorithm 1 computes a single matrix multiply and stores the result for later use. The input and output arguments are as follows. The term r_j denotes residual vectors from the CG algorithm. The term p_j denotes the H-conjugate vector; the term α_j denotes its corresponding weight with respect to the current residual. The term γ_j is a scalar quantity needed only in Algorithm 1. The term q denotes the current Lanczos vector, which is obtained by normalizing the residual

Algorithm 1 Conjugate Gradient Iteration (with Lanczos book-keeping)

function $[r_{j+1}, \alpha_j, p_j, \gamma_j, q, t_u, t_d] = \mathbf{cgUpdate}(H, r_j, \alpha_{j-1}, p_{j-1}, \gamma_{j-1})$
$\gamma_j = r_j^T r_j$;
if $\gamma_{j-1} = 0$ **then** $\beta = 0$ **else** $\beta = \gamma_j/\gamma_{j-1}$; **end if**
$p_j = r_j + \beta_j p_{j-1}$;
$\alpha_j = \gamma_j / p_j^T H p_j$;
$r_{j+1} = r_j - \alpha_j H p_j$;
Lanczos book-keeping steps:
$q = r_j/\sqrt{\gamma_j}$;
if $\alpha_{j-1} = 0$ **then**
    $t_d = 1/\alpha_j$ and $t_u = 0$;
else
    $t_d = 1/\alpha_j + \beta/\alpha_{j-1}$ and $t_u = -\sqrt{\beta}/\alpha_{j-1}$
end if
endfunction

vector. Finally, the terms t_u and t_d denote the corresponding diagonal and off-diagonal entries in the Lanczos tridiagonal matrix. Note that in a CG algorithm for a QP, the algorithm terminates when $\gamma_{j-1} = 0$, i.e. function $\mathbf{cgUpdate}(H, r_j, \alpha_{j-1}, p_{j-1}, \gamma_{j-1})$ will not be called.

Algorithm 2 defines the SSM update to be used in the eigCG-TR algorithm. The goal

Algorithm 2 SSM Update

function$[s, \sigma, v, \lambda] = \mathbf{ssmUpdate}(H, g, W, \Delta)$
Determine (z^*, η^*) as the minimum eigenpair of $(W^T H W)z = \eta(W^T W)z$.
Determine (u^*, ξ^*) by minimize (1) in span of W:

$$\begin{array}{ll} \underset{u}{\text{minimize}} & u^T(W^T g) + \frac{1}{2}u^T(W^T H W)u, \\ \text{subject to} & \|Wu\|_2 \leq \Delta_k \end{array} \tag{5}$$

Set $v = Wz^*$, and $\lambda = \eta^*$.
Set $s = Wu^*$, and $\sigma = \xi^*$
endfunction

of Algorithm 2 is to solve the eigenvalue and trust-region subproblems in steps 2 and 3 within the subspace spanned by the columns of W. For our purposes, the matrix W is typically of size $n \times k$ where $k \leq 6$, so the subproblems are computationally negligible to solve. Please note that the vector v calculated is used to create W for next subproblem in Step 4, and λ is calculated to track the progress of the algorithm.

3. SSM using annulus-based acceleration

In this section we describe the core SSM that will later incorporate the eigCG algorithm. The main steps we follow are shown in Algorithm 3. This algorithm keeps track of two solution estimates, one from a shifted CG iteration and one from SSM updates. We denote the shifted CG solution by $\hat{s}_k$ and the SSM solution by s_k. The algorithm monitors the proximity of these two solution pairs to determine when to restart the CG iteration.

More details on annulus-based SSM are discussed in [12]

The shifted CG estimate $\hat{s}$ begins with the naive assumption that we have a good initial guess for the trust-region multiplier $\hat{\sigma} \approx \sigma^*$. $\hat{s}$ is computed by applying CG to

$$(H + \hat{\sigma} I)\hat{s} = -g,$$

initialized at our current best approximate solution to the trust-region subproblem. If $\hat{\sigma}$ is sufficiently accurate, then CG should converge to a point satisfying

$$\|\hat{s}\| \leq (1 + \tau_1)\Delta$$

for any specified $\tau_1 > 0$. An SSM update can then generate the second solution estimate s, using vectors from the CG solve to form the basis matrix W. The columns in W form a basis for the subspace spanned by the two current solution estimates $\hat{s}_k$ and s_k, the current best estimate v_k of the minimum eigenvector, and the two most recent CG residual vectors r_k and r_{k+1}. The choice of vectors is significant, as we will show later in this section.

The algorithm continues in this manner, updating $\hat{s}$ via CG and s via SSM until either convergence is detected or $|\|s\| - \|\hat{s}\|| \geq \tau_1\Delta$. In this case, the algorithm restarts using the value of σ returned from the SSM update to define the CG shift $\hat{\sigma}$, initializing $\hat{s} = s$. We refer to the iterations where j is incremented (and CG restarts) as the *outer iterations* for Algorithm 3.

Algorithm 3 SSM with annulus-based acceleration

Require: H, g, and Δ, $\tau_1, \tau_2 \in (0, 1)$.
Require: $v_0 = 0$, $s_0 = 0$.
Initializations: $r_0 = -(g + Hs_0)$; $\gamma_{-1} = 0$; $p_{-1} = 0$;
$j = 0$, $\hat{\sigma}_j = 0$, $\hat{s}_k = 0$.
for $k = 0, \ldots,$maxitr **do**
  $[r_{k+1}, \alpha_k, \beta_k, \gamma_k, q_k, p_k] =$ **cgUpdate**$(H + \hat{\sigma}_j I, r_k, \gamma_{k-1}, p_{k-1})$;
  $\mathcal{A}_k = \{\hat{s}_{k-1}, r_k, r_{k+1}\}$
  $\mathcal{S}_k = \{s_{k-1}, v_{k-1}\} \cup \mathcal{A}_k$
  Set $W = \text{basis}(\mathcal{S}_k)$;
  $[s_k, \sigma_k, v_k, \lambda_k] =$ **ssmUpdate**(H, g, W, Δ);
  if $\alpha_k \leq 0$ **or** $\left| \|s_k\| - \|\hat{s}_k\| \right| > \tau_1\Delta$ **then** ▷ Restart CG
    $j = j + 1$;
    $\hat{\sigma}_j = \sigma_k$;
    $\hat{s}_k = s_k$;
    $r_{k+1} = -(H + \hat{\sigma}_j I)\hat{s}_k + g$.
    $\gamma_{k+1} = 0$.
  else
    $\hat{s}_{k+1} = \hat{s}_k + \alpha_k p_k$
  end if
  if $\|r_{k+1}\| \leq \tau_2\|g\|$ **then**
    break
  end if
end for

Let us consider the first outer iteration of Algorithm 3. Initially, $\hat{\sigma}_j = 0$, so $\hat{s}_k$ simply denotes the CG solution sequence from the system $Hs = -g$. As CG iterations continue,

either the algorithm converges to a point where $\|s_k\| \leq (1+\tau_1)\Delta$ and $\|Hs_k+g\| \leq \tau_2\|g\|$, or the algorithm enters the inner loop at Step 9.

If the algorithm enters the inner loop at Step 9, then j is set equal to 1 and the next CG shift $\hat{\sigma}_1$ is set equal to the most recent multiplier σ_k generated by the SSM update. The initial guess $\hat{s}_k$ for the CG iteration is set equal to the current best estimate s_k generated by the SSM update. By setting $\gamma_{k+1} = 0$ and $r_{k+1} = -(H + \hat{\sigma}_j I)\hat{s}_k + g$, the CG algorithm is effectively restarted.

An important element of Algorithm 3 is the choice of basis vectors in the subspace matrix W. The remainder of this section concerns the theory guiding our choice.

The first necessary theorem is the fundamental convergence theorem for SSM algorithms, stated below.

THEOREM 3.1 (Hager and Park [15]) *Suppose the sequence $\{s_k\}$ is generated recursively as*

$$[s_{k+1}, \sigma, v_k, \lambda_k] = \mathbf{ssmUpdate}(H, g, W_k, \Delta)$$

where the columns of W_k form a basis for $\text{span}(s_k, \nabla\psi(s_k), v^*)$*, and v^* is the corresponding eignvector of the smallest eignvalue λ^* of H. Then s_k converges to a global minimizer of the trust-region subproblem (1).*

We next prove a lemma that demonstrates that Algorithm 3 has several desirable properties.

LEMMA 3.2 *Let s^* be the optimal solution to the trust-region subproblem (1) and let λ^* be the minimum eigenvalue of H. For any subspace sequence $\mathcal{S}_k$ we have*

(1) $\psi(s^*) \leq \psi(s_k) \leq \psi(s_{k-1})$,
(2) $\psi(s_k) \leq \psi(\hat{s}_k)$,
(3) $\lambda^* \leq \lambda_k \leq \lambda_{k-1}$,
(4) $s_k^T g \leq 0$.

While $j = 0$, $s = \hat{s}_k$ can be computed directly by applying the CG algorithm to $Hs = -g$. For $k \geq 1$, $\psi(s_k) \leq \psi(s_c)$, where s_c denotes the corresponding Cauchy-Point solution.

Proof. The definition of Algorithm 2 requires that $\|s_j\| \leq \Delta$ and $\|v_j\| = 1$ for all iterates j. By construction, W_k in Algorithm 2 satisfies $W_k e_1 = s_{k-1}$ and $W_k e_2 = v_{k-1}$. Hence,

$$\psi(s_k) = \min_{\|W_k u\| \leq \Delta} \psi(W_k u) \leq \psi(W_k e_1) = \psi(s_{k-1}).$$

Similarly we have that

$$\lambda_k = \min_{\|W_k z\|=1} z W_k^T H W_k z \leq e_2 W_k^T H W_k e_2 = \lambda_{k-1}.$$

By definition, $\psi(s^*)$ and λ^* are the respective lower bounds on $\psi(s_k)$ and λ_k for all vectors in $\mathbb{R}^n$. Suppose $s_k^T g > 0$. Then $\psi(-s_k) < \psi(s_k)$ and $-s_k \in \mathcal{S}_k$, which contradicts Step 5 in Algorithm 2. The final assertion ($\psi(s_k) \leq \psi(s_c)$ for $k \geq 1$) follows by observing that the Cauchy point is obtained by solving the equation in Step 2 of Algorithm 2 with $W = \{g\}$. ■

It is important to note that the CG solution $\hat{s}$ globally minimizes $\psi(s)$ in the current Krylov subspace and $\psi(s_k) \leq \psi(\hat{s}_k)$ for all k. This implies that the SSM solution s_k is

equivalent to $\hat{s}_k$ for the first outer iteration ($j = 0$) of Algorithm 3, so the computation of s_k may be omitted while $j = 0$. Thus while $\|s_k\| \leq (1 + \tau_1)\Delta$ and $j = 0$, Algorithm 3 is computationally equivalent to CG as shown by following lemma.

LEMMA 3.3 *Assuming that, in Algorithm 3 the solution from CG lies on the interior of the trust region boundary, starting from the same solution s_0, CG and SSM will generate same sequence of solutions s_k and $\hat{s}_k$.*

Proof. It is well known that at every iteration, the CG solution s_k is the global minimizer of the quadratic function, restricted to the corresponding Krylov subspace $\mathcal{K} = \{r_0, Ar_0, A^2r_0, \ldots, A^{k-1}r_0\}$. Thus, it remains to show that the solution generated by SSM would be the minimizer in this subspace. At each iteration k, SSM works on a subspace created by $\mathcal{M} = \{\hat{s}_k, \hat{v}_k, r_{k-1}, r_k, s_{k-1}, v_{k-1}\}$. Given that s_k is already in the SSM subspace, it would be the minimizer on the Krylov subspace, it just remains to show that

$$\text{span}(\mathcal{M}) \subset \text{span}(\mathcal{K})$$

Since all vectors spanning $\mathcal{M}$ remain in the subspace except for the eigenvectors, we need to show that the eigenvector estimates also belong to the Krylov subspace so far created. From step 15 of Algorithm 5, the eigenvectors are computed and projected back to full space by using Q, thus lie on the subspace spanned by Q. Before $k = m$, Q stores the normalized residuals ($\frac{r_j}{\|r_j\|}$) which span is equal to that of $\mathcal{K}$ based on previously known CG results. On step 15, since we have $Q_{\text{new}} = QWZ$, thus $\text{span}(Q_{\text{new}}) \subset \text{span}(QWZ)$, therefore the newer eigenvectors calculated using Q_{new} would be in the $\text{span}(\mathcal{K})$. This means that will not grow out of the already explored conjugate direction i.e. $\text{span}(\mathcal{M}) \subset \text{span}(\mathcal{K})$, and since $s_k \in \text{span}(\mathcal{M})$ the global minimizer would be chosen when solving in the created subspace by $\mathcal{M}$.

■

4. The eigCG method

In this section we discuss the eigCG method developed recently by Stathopoulos and Orginos [24]. The algorithm was originally developed for solving a sequence of large-scale symmetric positive-definite linear systems

$$Ax = b_i, \text{ for } i = 1, \ldots, m. \tag{6}$$

Because A is positive-definite, the CG algorithm is a natural choice for solving $Ax = b_i$. It is well know that the extreme eigenvectors (if efficiently obtainable) of A may be used to either precondition A or deflate b_i [1, 4, 6, 22, 24]. The eigCG algorithm seeks to generate eigenvectors while applying unmodified CG to $Ax = b_j$ in order to reduce the cost of solving $Ax = b_{j+1}$.

For such an approach to be practical, three properties are required. First, accurate eigenvectors must be obtainable (otherwise one would not expect deflation to reduce CG solve time). Second, eigenvectors computations should be restricted to use only information that is naturally available during a CG solve. And third, the additional cost to the CG algorithm should be nominal. By design eigCG satisfies the second and third properties. Remarkably, extensive numerical results have demonstrated the eigCG

algorithm to be an accurate and efficient eigen solver as well.

It is worth noting that this is in direct contrast to algorithm like IRLM which require the Lanczos factorization to periodically be restarted. Though the Lanczos algorithm and CG are mathematically equivalent given the same initial starting point, after the first restart (which occurs quite early in the algorithm), such a correspondence is immediately lost. Thus, for example, it would be inefficient to attempt to solve a linear system using the sequence of partial Lanczos factorization generated by IRLM.

For the remainder of this section we will primarily focus on Algorithm 4, describing the computational work and algorithmic steps used in eigCG. Algorithm 4 is identical to CG save for an additional step that uses the current CG iterates to periodically improve the current best approximate solution to eigenvalue subproblem (2) in Step 6. We emphasize that Step 6 has no external effect on the CG algorithm itself, and simply uses the current corresponding Lanczos information available during the CG algorithm to update matrices Q_j and T_j. Thus if Step 6 is omitted from Algorithm 4, the resulting algorithm is identical to an ordinary CG algorithm.

Algorithm 4 eigCG Algorithm (in functional form)

Require: s_0, H, τ
Require: m, n_{ev} with $2n_{ev} < m$
1: Initializations: $r_0 = -g - Hs_0$; $\gamma_{-1} = 0$; $p_{-1} = 0$; $\alpha_{j-1} = 1$;
2: $T_0 = []$, $Q_0 = []$;
3: **for** $j = 0, \ldots,$maxitr **do**
4: $\quad [r_{j+1}, \alpha_j, p_j, \gamma_j, q, t_u, t_d] = \mathbf{cgUpdate}(H, r_j, \alpha_{j-1}, p_{j-1}, \gamma_{j-1})$
5: $\quad s_{j+1} = s_j + \alpha_j p_j$;
6: $\quad [T_{j+1}, Q_{j+1}] = \mathbf{eigCGUpdate}(T_j, Q_j, q, t_u, t_d, m, n_{ev})$
7: $\quad$ **if** $\|r_{j+1}\| \leq \tau \|r_0\|$ **then**
8: $\quad\quad$ **break;**
9: $\quad$ **end if**
10: **end for**

Algorithm 4 begins by applying the CG algorithm to positive-definite system $Hs = -g$, generating standard CG iterates in Step 4 according to Algorithm 1; the CG approximate solution s_{j+1} is then updated in Step 5 using pre-computed weight α_j. If the relative residual is sufficiently reduced, the CG algorithm exits in Step 7, having achieved the desired accuracy for the linear system $Hs = -g$. The only ingredients from CG itself, needed to compute Step 6, are the current Lanczos vector q and the corresponding Lanczos tridiagonal entries t_d and t_u. While Step 7 remains false, the eigenvalue component of the eigCG algorithm thus runs unobtrusively in the background, absorbing information from the CG algorithm and periodically improving internal eigenvalue estimates.

We next described Step 6 of Algorithm 4 presented in in Algorithm 5, whose primary goal is to re-use information produced during the CG process to determine the n_{ev} eigenvectors corresponding to the n_{ev} smallest eigenvalues of H. This is achieved by progressively building and updating two key matrices: Q and T. $T = (T_{i,j})_{m*m}$. Matrix Q is used to hold (in its columns) a small dimensional subspace of m eigenvectors (with $m \ll n$). Algorithm 5 ensures the matrices Q and T satisfy the following properties (reminiscent of the Lanczos vectors and Lanczos tridiagonal) on exit:

$$Q^T H Q = T \text{ and } Q Q^T = I. \tag{7}$$

In general we will not, however, retain the Lanczos property that HQ equals QT plus a

rank one residual term; this latter condition is compromised to avoid a necessary restart to CG. We emphasize that no actual matrix-vector products are computed in Algorithm 5 when implemented properly.

Algorithm 5 EigCG Update

1: **function**$[T, Q, v_{\text{best}}] =$ **eigCGUpdate**$(T, Q, q_{new}, t_u, t_d, m, n_{ev})$
2: $[k, k] = \text{size}(T)$;
3: **if** k = 0 **then**
4: $T_{1,1} = t_d$
5: **else**
6: $T_{k+1,k+1} = t_d$ and $T_{k,k+1} = T_{k+1,k} = t_u$
7: **end if**
8: **if** k = m **then**
9: Let T and $\hat{T}$ denote m and m-1 principal sub-matrices of T respectively;
10: Let Y store eigenvectors of n_{ev} smallest eigenvalues of T;
11: Let $\hat{Y}$ store eigenvectors of n_{ev} smallest eigenvalues of $\hat{T}$;
12: Let W denotes an orthonormal basis for $\left(Y, \begin{bmatrix}\hat{Y}\\ 0\end{bmatrix}\right) \in \mathbb{R}^{m\times 2n_{ev}}$
13: Let E denote diagonal matrix of $2n_{ev}$ eigenvalues of $W^T TW$;
14: Let Z denote corresponding $2n_{ev}$ eigenvectors of $W^T TW$;
15: Set $Q \leftarrow QWZ$; now $Q \in \mathbb{R}^{n\times 2n_{ev}}$
16: Set $T = \begin{pmatrix} E & Q^T H q_{\text{new}} \\ q_{\text{new}}^T HQ & T_{k+1,k+1}\end{pmatrix}$
17: **end if**
18: Let v_{best} denote vector V with smallest Rayleigh Quotient of $W^T TW$.
19: Set $Q \leftarrow [Q, q_{\text{new}}]$
20: **endfunction**

For the first m-1 iterations, T is identical to the corresponding Lanczos tridiagonal matrix. This can be seen in steps 3-7, where the new Lanczos tridiagonal elements t_d and t_u are simply appended to matrix T. Similarly, the matrix Q, which is updated in step 19, contains the Lanczos vectors computed during the CG update as the normalized residuals (see step 7 in Algorithm 1).

When T becomes an $m \times m$ matrix, an internal restart occurs that reduces the dimension of T to $(2n_{ev} + 1) \times (2n_{ev} + 1)$, implying Q and T will never be large than $n \times m$ and $m \times m$ respectively. To better describe this part of the algorithm, we define the first $m - 1$ columns of the matrix Q by $\hat{Q}$. Then at this point we have the properties that

$$Q^T HQ = T \text{ and } \hat{Q}H\hat{Q} = \hat{T},$$

where denotes the $m - 1$ principal submatrix of T. We next compute the n_{ev} smallest eigenvalues of T and $\hat{T}$ and store their corresponding eigenvectors in matrix Y and $\hat{Y}$ respectively. As stated in [24], we may think of QY and $\hat{Q}\hat{Y}$ as two consecutive solution estimates for estimating the n_{ev} smallest eigenvalues of H.

The next computation denotes the fundamental step of the algorithm, proof of its surprising effectiveness still pending: the algorithm simply forms the new Q matrix using columns that form an orthonormal basis for the

$$\text{span}(QY, \hat{Q}\hat{Y}),$$

with the addition property that $Q^T HQ$ denotes a diagonal matrix. We thus first need a mechanism for forming an orthonormal basis matrix for $(QY, \hat{Q}\hat{Y})$. A less cost-effective (but much simpler to explain) method for computing the new basis matrix would be to perform something like Gram-Schmidt on $(QY, \hat{Q}\hat{Y})$; however, it is much preferable to, whenever possible, work in the reduce $\mathcal{O}(m)$ dimension. Since

$$(QY, \hat{Q}\hat{Y}) = QA, \text{ where } A \triangleq \left(Y, \begin{bmatrix}\hat{Y}\\ 0\end{bmatrix}\right),$$

rather than orthogonalzing a matrix of size $n \times 2n_{ev}$, we can instead orthognalize the much smaller $m \times 2n_{ev}$ matrix A, generating the orthonormal basis matrix W in Step 12. That is W at this point satisfies the property

$$\text{span}(W) = \text{span}\left(Y, \begin{bmatrix}\hat{Y}\\ 0\end{bmatrix}\right) \text{ and } W^T W = I,$$

At this point we could replace the current matrix Q with QW, however, the product $(QW)^T H(QW)$ is dense, not diagonal. Thus we form the eigendecomposition

$$ZEZ^T = (QW)^T H(QW) = W^T TW,$$

and ultimately redefine $Q = QWZ$ in Step 15. At this point the matrix Q now satisfies

$$Q^T HQ = E \text{ and } Q^T Q = I.$$

Finally we append the current Lanczos vector to the end of Q in Step 19, which implies Q now satisfies

$$Q^T HQ = \begin{pmatrix} E & Q^T Hq_{\text{new}} \\ q_{\text{new}}^T HQ & T_{k+1,k+1}\end{pmatrix}$$

in addition to $Q^T Q$ equaling the identity matrix, justifying the corresponding update made to T in Step 17. On exit T will have the classical "chicken-foot" nonzero structure reminiscent of other restarted Lanczos algorithms.

This conclude our description of Algorithm 4. For a more in depth description we refer the interested reader to [24]. Remarkably, without modification of r_{j+1}, significant numerical results demonstrate that the first n_{ev} vectors of Q_{j+1} converge to the first n_{ev} eigenvectors corresponding to the n_{ev} smallest eigenvectors of H with convergence rates comparable in accuracy to unrestarted (stabilized and hence significantly more costly) Lanczos methods [24].

To obtain a direct analogy of the above process in the area of optimization we simply need to recall that iterative trust-region methods such as Luksan2004 [7, 8, 13, 14] apply the CG method to different variations of the shifted system

$$(H + \sigma_i I)s = -g, \text{ for } i = 1, \ldots, m. \tag{8}$$

where $\sigma_i \to \sigma^*$ respectively. Note that as in case for the specific application suggested by [24] for eigCG, the eigenvectors for (8) is invariant with respect to σ_i. We will exploit this property in section 5 to develop a numerically robust and efficient iterative trust-region subproblem solver with nominal computational overhead expended each iteration.

5. eigCGTR

Algorithm 6 combines Algorithm 2 and Algorithm 4 into a single algorithm dedicated to solving the trust-region subproblem to any level of desired accuracy. We stress that the following algorithm require minimal overhead to that of the original CG algorithm. When including Algorithm 4 in the framework of Algorithm 2, a key point is we do not

Algorithm 6 eigCG-TR

Require: H, g, and δ
Require: $\epsilon \in (0,1)$, $\tau > 1$.
Require: m, n_{ev} with $2n_{ev} < m$
Require: Initial guesses for v^* and s^*: v_0, s_0.

Initializations: $j = 0$, $\hat{\sigma}_j = 0$, $\hat{s}_0 = 0$.
$r_0 = -(g + Hs_0)$; $\gamma_0 = 0$; $p_0 = 0$;
for $k = 1, \ldots, 2n$ **do**
    $[r_k, \alpha_k, p_k, \gamma_j, q, t_u, t_d] = \mathbf{cgUpdate}(H + \hat{\sigma}_j I, r_{k-1}, \alpha_{k-1}, p_{k-1}, \gamma_{k-1})$;
    $\hat{s}_k = \hat{s}_{k-1} + \alpha_k p_k$;
    $[T_{k+1}, Q_{k+1}, \hat{v}_k] = \mathbf{eigCGUpdate}(T_k, Q_k, q, t_u, t_d, m, n_{ev})$

    $\mathcal{A}_k = \{\hat{s}_k, \hat{v}_k, r_{k-1}, r_k\}$;
    $\mathcal{S}_k = \{s_{k-1}, v_{k-1}\} \cup \mathcal{A}_k$
    Set $W = \text{basis}(\mathcal{S}_k)$;
    $[s_k, \sigma_k, v_k, \lambda_k] = \mathbf{ssmUpdate}(H, g, W, \delta)$;
    if $p_k^T(H + \hat{\sigma}_j I)p_k \leq 0$ **or** $\left| \|s_k\| - \|\hat{s}_k\| \right| \geq \epsilon\delta$ **then** ▷ Restart CG
        $j = j + 1$;
        $W = [Q_{k+1}, s_k]$;
        $[s_k, \sigma_k, v_k, \lambda_k] = \mathbf{ssmUpdate}(H, g, W, \delta)$;
        $\hat{\sigma}_j = \sigma_k$; $\hat{s}_k = s_k$;
        $r_{k+1} = -(H + \hat{\sigma}_j I)\hat{s}_k - g$, $\gamma_{k+1} = 0$, $\alpha_{k+1} = 0$.
    end if

    if $\|r_k\| \leq \epsilon\|g\|$ **then**
        break ▷ Approximate solution found
    end if
end for

wish to necessarily restart eigCG every time CG restarts. A fortunate feature of Lanczos is that it is invariant to shifting; that is, we may add any multiple of Q_k to both sides of the Lanczos equations

$$HQ_k = Q_k T_k + r_{k+1} e_{k+1}^T$$

to get an equivalent shifted relation

$$(H + \sigma I)Q_k = Q_k(T_k + \sigma I) + r_{k+1} e_{k+1}^T,$$

while the remaining Lanczos conditions $Q_k^T r_{k+1}$ and $Q_k^T Q_k = I$ continue to hold as before. However, in Algorithm 6 not only to we add a shift to H, but we also replace r_{k+1} with $\hat{r}_{k+1}$. Thus to avoid a complete restart to eigCG, we need to ensure that

the new $\hat{s}_k$ assigned in Step (16) of Algorithm 6 is chosen so that $Q_k^T \hat{r}_{k+1} = 0$. One possibility possible route to ensure thing orthogonality relation is to determine $\hat{s}_k$ as the unconstrained minimizer of the shifted quadratic model with the subspace spanned by the current Q_k and s_k, that is, set $\hat{s}_k = \begin{pmatrix} Q_k & s_k \end{pmatrix} y$ where y denotes the solution to

$$\begin{pmatrix} Q_k^T \\ s_k^T \end{pmatrix} (H + \hat{\sigma}_j I) \begin{pmatrix} Q_k & s_k \end{pmatrix} \begin{pmatrix} y_q \\ y_s \end{pmatrix} = - \begin{pmatrix} Q_k^T \\ s_k^T \end{pmatrix} g, \tag{9}$$

which expands to the $k+1$ system of the form

$$\begin{pmatrix} T_k + \hat{\sigma}_j I & Q_k^T (H s_k + \hat{\sigma}_j s_k) \\ (H s_k + \hat{\sigma}_j s_k)^T Q_k & s_k^T H s_k + \hat{\sigma}_j s_k^T s_k \end{pmatrix} \begin{pmatrix} y_q \\ y_s \end{pmatrix} = - \begin{pmatrix} Q_k^T g \\ s_k^T g \end{pmatrix}. \tag{10}$$

Thus when CG restarts, we reset the following quantities

$$\begin{aligned} \hat{s}_{k+1} &= Q_k y_q + y_s s_k & (11) \\ \hat{r}_{k+1} &= -(H + \hat{\sigma}_j I)\hat{s}_{k+1} - g & (12) \\ T_{k+1} &= T_{k+1} + \hat{\sigma}_j I; & (13) \end{aligned}$$

By constructing the subspace matrix W from current solution estimates and the current best estimate for the minimum eigenvector, we ensure that the solution quality is maintained as stated in Lemma 3.2. Including the previous two residual vectors form the concurrent CG solve ensures an improvement in solution quality and ultimately enables us to prove our next theorem.

The following theorem ensures Algorithm 6 is globally convergent as long as v_k converges to v^*.

THEOREM 5.1 *If the global trust-region solution is interior, then Algorithm 6 reduces to CG on $Hs = -g$, so the algorithm is globally convergent. Otherwise, Algorithm 6 converges to a point satisfying*

$$(1 - \tau_1)\Delta \leq \|\hat{s}\| \leq (1 + \tau_1)\Delta, \tag{14}$$

$$\|(H + \hat{\sigma}_j I)\hat{s} + g\| \leq \tau_2 \|g\|, \tag{15}$$

whenever v_k converges to v^*.

Proof. If the solution lies within the interior, H is positive definite and $\alpha_k > 0$ for every iteration k. Furthermore, because r_k and r_{k+1} (which are simply multiples of the Lanczos vectors q_k and q_{k+1}, respectively) are contained in the subspace $\mathcal{S}_k$ at each iteration,thus $\hat{\sigma}_j = 0$ for all j, and $\hat{s}_k$ simply denotes the classical CG iterates.

If the solution lies on the boundary, we can identify two cases. The first case is when σ_j is modified only a finite number of times. This implies that there exists an iteration K such that for all $k > K$, $\hat{s}_k$ satisfies (14). Furthermore, the standard CG method (without restarts) is being applied to the system $(H + \hat{\sigma}_K I)s = -g$. By the properties of CG, we are assured that $\|(H + \hat{\sigma}_K I)\hat{s} + g\|$ converges to 0, and as a result condition (15) holds for $j = K$.

The second case arises when $\hat{\sigma}_j$ is modified infinitely often. In this case, $\mathcal{S}_k$ contains the vectors s_k and $r_k = (H + \hat{\sigma}_j I)s_k + g$ infinitely often. By noting that $r_k - \sigma_j s_k = Hs_k + g$, we can observe that $\nabla\psi(s_k)$ is implicitly contained in the subspace $\mathcal{S}_k$. Because $v_k \to v^*$, we can apply Theorem 3.1 to this sub-sequence and conclude that Algorithm 6 converges. ■

Note that in the proof of this theorem, global convergence requires only that r_k and r_{k+1} be members of $\mathcal{S}_k$. The remaining vectors are merely added to accelerate convergence.

6. Numerical Results

To demonstrate that the algorithms works, a trust region algorithm is developed, and Algorithm 6 is used to solve the trust region QP subproblems. In this section we report the numerical results for the algorithm on a test set containing unconstrained problems. This set includes total 240 problems, and they are from the unconstrained CUTEr problems [2, 10], Hock and Schittkowski problems [16, 23], and FLOUDAS test problems [9]. We have done the numerical tests by using both Steihaug-Toint method [5] and Algorithm 6 developed in this paper. It has been shown that Algorithm 6 works well in this set of test cases. Comparing that the Steihaug-Toint method solves 233 test cases, Algorithm 6 solves 238 among 240 test cases. It is also found that Algorithm 6 uses fewer number of function evaluations, and fewer iterations. Note that the Steihaug-Toint method uses less CPU time on some test cases, and we think that this could be due to that the Steihaug-Toint method often exits early. How to exit early with Algorithm 6 to prevent over-solving QP subproblems will be future research work. Nevertheless, Algorithm 6 can exit with controlled acuracy for the solution of QP subproblems. Figure 1, shows that from the solution time point of view, Steihaug-Toint method uses less CPU time on some test cases, however there are some cases that eventually ECTR out-performs. From the number of iterations viewpoint, figure 6 shows that ECTR out-performs the Steihaug-Toint method in call cases by achieving fewer number of iterations. Lastly, figure 6 shows the number of function evaluations, where incorporating the estimates of eigen information helps ECTR outperform Steihaug-Toint.

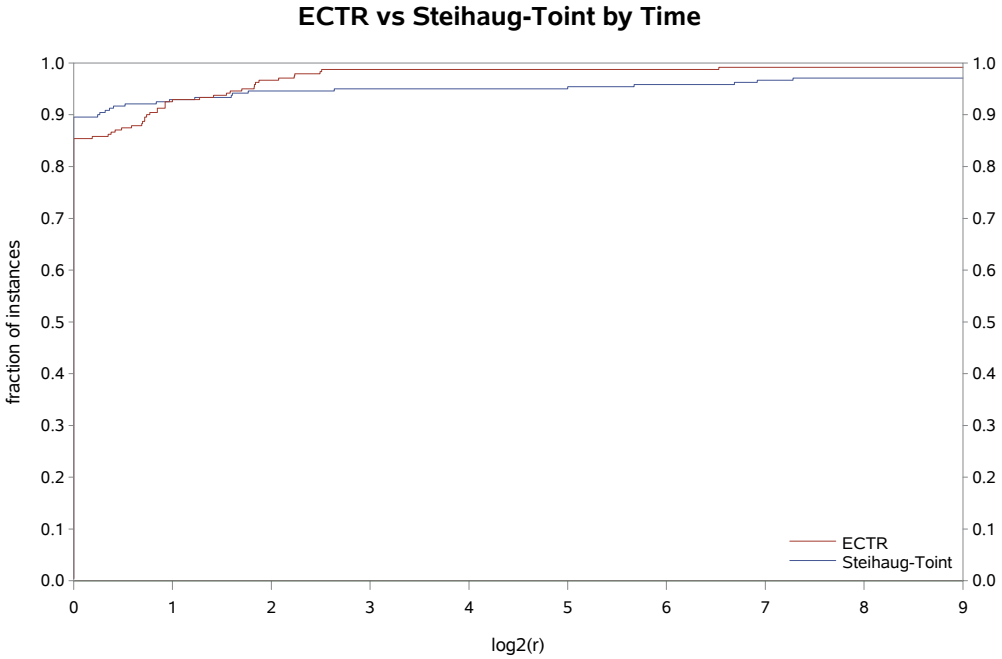

Figure 1. Performance profile comparing ECTR vs Steihaug-Toint on the solution time

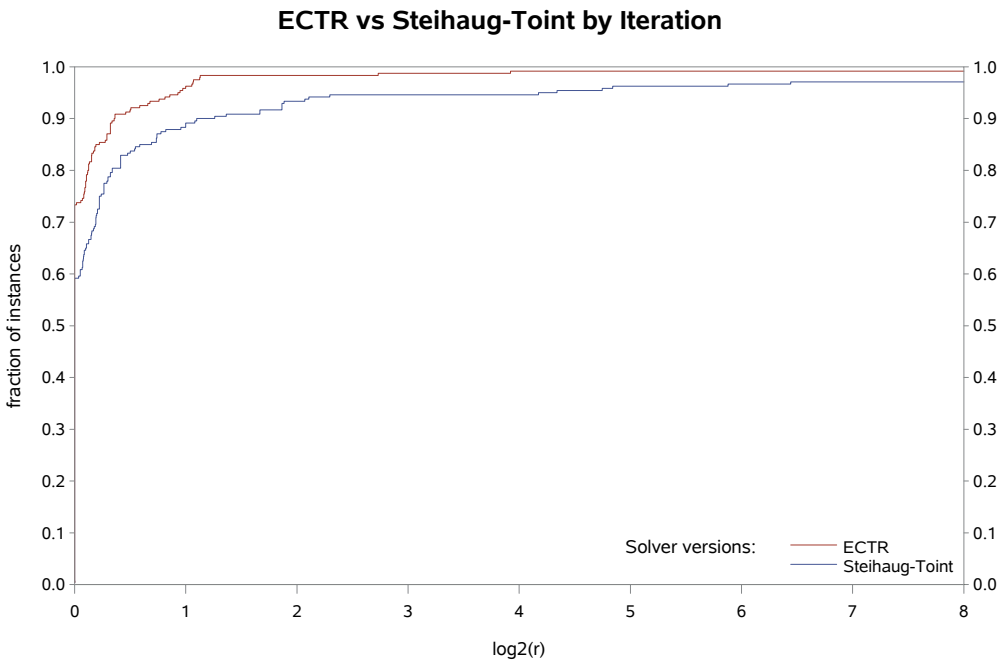

Figure 2. Performance profile comparing ECTR vs Steihaug-Toint by the number of iterates

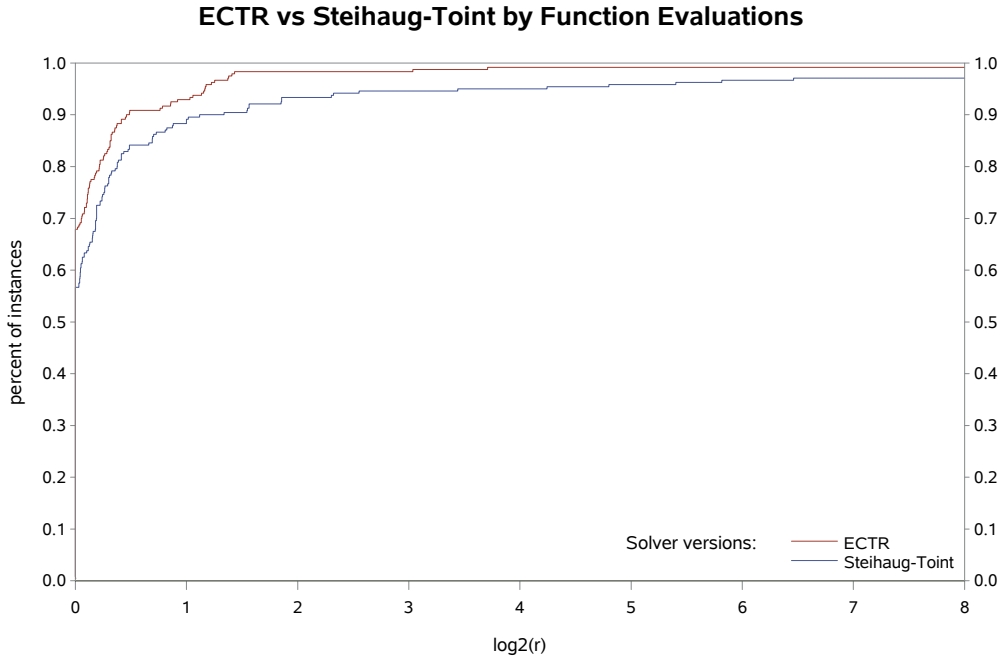

Figure 3. Performance profile comparing ECTR vs Steihaug-Toint by the number of function evaluations

7. Conclusion

In this paper, we have leveraged recent groundbreaking work by Stathopoulos and Orginos [24], and we present the first iterative trust-region subproblem solver that, with negligible computational overhead compared to existing iterative trust-region approaches, at its core contains a robust and practical eigenvalue solver. This innovated strategy em-

ployed by Stathopoulos and Orginos [24] can be utilized in this context because, unlike other restarted Lanczos methods, the restarts used do not necessarily modify the current Lanczos sequence generated by CG. Initial convergence analysis has been done, and numerical results are given to illustrate the efficiency of the algorithm.

8. Appendix

In this section we assume that $\{\lambda_1, \lambda_2, \ldots, \lambda_n)$ denotes the eigenvalues of H listed in ascending order with corresponding eigenvectors $\{v_1, \ldots, v_n\}$. The following theorem emphasizes that, regardless of whether or not the hard-case occurs, v_1 plays a key role in the solution composition of the trust-region subproblem solution s^*. Further, eigenvector corresponding to smaller eigenvalues will on average be given larger weights.

THEOREM 8.1 *Suppose (s^*, σ^*) denotes a solution to (1). Given any constant $C > \lambda_1$, we can decompose the solution as*

$$s^* = v + r, \text{ with } \|r\| \leq \frac{\|g\|}{C - \lambda_1} \text{ and } \|s^*\| = \|v\| + \|r\|,$$

where $v \in \text{span}\{v_i : \lambda_i < C\}$ and $r \in \text{span}\{v_i : \lambda_i \geq C\}$. Further, whenever H is not positive definite,

$$\lim_{\delta \to \infty} \frac{\|s^* - v\|}{\|s^*\|} = 0.$$

Proof. From equations (3) and (4), we know that if $\sigma^* > -\lambda_1$,

$$s^* = \sum_{i=1}^{n} \left(\frac{g^T v_i}{\sigma^* + \lambda_i} \right) v_i$$

And if $\sigma^* = -\lambda_1$, then

$$s^* = \tau v_1 + \sum_{\lambda_1 < \lambda_i} \left(\frac{g^T v_i}{\sigma^* + \lambda_i} \right) v_i$$

Thus in either case we can partition the solution as

$$s^* = v + \left(\sum_{\lambda_i \geq C} \frac{g^T v_i}{\sigma^* + \lambda_i} v_i \right).$$

where $v \in \text{span}\{v_i : \lambda_i < C\}$. And we may define $r = s^* - v$. Then

$$\|r\|^2 = \sum_{\lambda_i \geq C} \left(\frac{g^T v_i}{\sigma^* + \lambda_i} \right)^2 \leq \sum_{\lambda_i \geq C} \left(\frac{g^T v_i}{\lambda_i - \lambda_1} \right)^2 \leq \sum_{i=1}^{n} \left(\frac{g^T v_i}{C - \lambda_1} \right)^2,$$

which implies $\|r\| \leq \|g\|/(C - \lambda_1)$. When H is not positive definite,

$$\lim_{\delta \to \infty} \frac{\|s^* - v\|}{\|s^*\|} = 0$$

follows as

$$\lim_{\delta \to \infty} \|s^*\| = \infty$$

■

Acknowledgments

The authors profusely thank Manoj Chari, Yan Xu, Katya Scheinberg, and Jon W. Tolle for many insightful discussions, and continual support for research and development of this work.

References

[1] A.M. Abdel-Rehim, R.B. Morgan, D.A. Nicely, and W. Wilcox, *Deflated and restarted symmetric lanczos methods for eigenvalues and linear equations with multiple right-hand sides* (2008), Available at https://arxiv.org/pdf/0806.3477.pdf.

[2] I. Bongartz, A.R. Conn, N.I.M. Gould, and Ph. L. Toint, *CUTE: Constrained and unconstrained testing environment*, Report 93/10, Département de Mathématique, Facultés Universitaires de Namur, 1993.

[3] D. Calvetti, L. Reichel, and D.C. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, ETNA 2 (1994), pp. 1–21, Available at citeseer.ist.psu.edu/calvetti94implicitly.html.

[4] A. Chapman and Y. Saad, *Deflated and augmented krylov subspace techniques*, Numerical Linear Algebra with Applications 4 (1998), pp. 43–66.

[5] A.R. Conn, N.I.M. Gould, and Ph. L. Toint, *Trust-Region Methods*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.

[6] J. Erhel and F. Guyomarc'h, *An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems*, SIAM J. Matrix Anal. Appl. 21 (2000), pp. 1279–1299.

[7] J.B. Erway and P.E. Gill, *An interior method for computing a trust-region step*, Numerical Analysis Report 08-1, Department of Mathematics, University of California San Diego, La Jolla, CA, 2008.

[8] J.B. Erway, P.E. Gill, and J.D. Griffin, *Iterative methods for finding a trust-region step*, Numerical Analysis Report 07-2, Department of Mathematics, University of California San Diego, La Jolla, CA, 2007.

[9] C.A. Floudas, P.M. Pardalos, C.S. Adjiman, W.R. Esposito, Z. Gümüs, S.T. Harding, J.L. Klepeis, C.A. Meyer, and C.A. Schweiger, *Handbook of Test Problems for Local and Global Optimization*, Kluwer Academic Publishers, 1999.

[10] N.I.M. Gould, D. Orban, and Ph. L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Softw. 29 (2003), pp. 373–394.

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

[12] M. Gratton, *Algorithms for trust-region subproblems with linear inequality constraints*, Ph.D. diss., University of North Carolina at Chapel Hill, 2012.

[13] J.D. Griffin, *Interior-point methods for large-scale nonconvex optimization*, Ph.D. diss., Department of Mathematics, University of California San Diego, 2005.

[14] W.W. Hager, *Minimizing a quadratic over a sphere*, SIAM J. Optim. 12 (2001), pp. 188–208 (electronic).

[15] W.W. Hager and S. Park, *Global convergence of SSM for minimizing a quadratic over a sphere*, Math. Comp. 74 (2004), pp. 1413–1423.
[16] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems 187, Springer Verlag, Berlin, Heidelberg and New York, 1981.
[17] R.B. Lehoucq and D.C. Sorensen, *Deflation techniques for an implicitly restarted arnoldi iteration*, SIAM J. Matrix Anal. Appl. 17 (1996), pp. 789–821.
[18] J.J. Moré and D.C. Sorensen, *Computing a trust region step*, SIAM J. Sci. and Statist. Comput. 4 (1983), pp. 553–572.
[19] B.N. Parlett and D.S. Scott, *The Lanczos algorithm with selective orthogonalization*, Math. Comp. 33 (1979), pp. 217–38.
[20] Ph. L. Toint, *Towards an efficient sparsity exploiting Newton method for minimization*, in *Sparse Matrices and Their Uses*, Academic Press, London and New York, 1981, pp. 57–88.
[21] 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 (2000), pp. 611–646 (electronic).
[22] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h, *A deflated version of the conjugate gradient algorithm*, SIAM J. Sci. Comput. 21 (2000), pp. 1909–1926.
[23] K. Schittkowski, *More Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems 282, Springer Verlag, Berlin, Heidelberg and New York, 1987.
[24] A. Stathopoulos and K. Orginos, *Computing and deflating eigenvalues while solving multiple right hand side linear systems in quantum chromodynamics*, Tech. Rep., College of William and Mary, 2008, Available at http://arxiv.org/PS_cache/arxiv/pdf/0707/0707.0131v1.pdf.
[25] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal. 20 (1983), pp. 626–637.
[26] K. Wu and H.D. Simon, *Thick-Restart Lanczos Method for Symmetric Eigenvalue Problems*, in *IRREGULAR*, Lecture Notes in Computer Science, Vol. 1457, Springer, 1998, pp. 43–55.
[27] S. Yektamaram, *Optimization algorithms for machine learning designed for parallel and distributed environments*, Ph.D. diss., Lehigh University, Bethlehem PA, 2018.