DELFT UNIVERSITY OF TECHNOLOGY

REPORT 10-15

Accelerating the LSTRS algorithm

J. Lampe, M. Rojas, D.C. Sorensen, and H. Voss

July 2009. Revised July 2010.

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2010

ACCELERATING THE LSTRS ALGORITHM

J. LAMPE [*], M. ROJAS [†], D.C. SORENSEN [‡], AND H. VOSS [§]

Abstract. The LSTRS software for the efficient solution of the Large-Scale Trust-Region Subproblem was proposed in [Rojas, Santos, Sorensen: ACM ToMS 34 (2008), Article 11]. The LSTRS method is based on recasting the problem in terms of a parameter-dependent eigenvalue problem and adjusting the parameter iteratively. The essential work at each iteration is the solution of an eigenvalue problem for the smallest eigenvalue of a bordered Hessian matrix (or two smallest eigenvalues in the potential hard case) and associated eigenvector(s). Using the Nonlinear Arnoldi method to solve the eigenvalue problems makes it possible to recycle most of the information from previous iterations which can substantially accelerate LSTRS.

Key words. constrained quadratic optimization, regularization, trust-region, ARPACK, Nonlinear Arnoldi method

AMS subject classification. 65F15, 65F22, 65F30

1. Introduction.

We consider the Trust-Region Subproblem (TRS) of minimizing a quadratic function subject to a spherical constraint:

$$\min_x \ \psi(x) := \frac{1}{2}x^T Hx + g^T x \quad \text{subject to } \|x\| \le \Delta \tag{1.1}$$

where $H = H^T \in \mathbb{R}^{n\times n}$, $g \in \mathbb{R}^n$ and $\Delta > 0$ are given. In (1.1) and throughout the paper, $\|\cdot\|$ denotes the Euclidean norm. We assume that H is large and possibly not explicitly available, so that factoring H is either not possible or too expensive. We also assume that matrix-vector products with H can be efficiently computed.

Problem (1.1) arises in connection with the trust-region globalization strategy underlying state-of-the-art trust-region methods in optimization (cf. [5, 24]). The more expensive computation in trust-region methods is the solution of a TRS per iteration. In that context, ψ is a local quadratic model of a general nonlinear objective function f. The region $D_\Delta := \{\|x\| \le \Delta\}$, centered at the current iterate, represents a region where we *trust* ψ to be a good approximation of f. The radius Δ is updated at each iteration. A special case of (1.1) is the least squares problem with a norm constraint, i.e. $\psi(x) := \|Ax - b\|^2$, which is equivalent to Tikhonov regularization for discrete forms of ill-posed problems. The radius Δ is fixed in regularization.

Problem (1.1) always has a solution that can lie in the interior or on the boundary of the trust region. The solution is unique if H is positive definite. The problem may have multiple solutions if H is not positive definite. Necessary and sufficient conditions for optimality were derived independently in [7, 35] and are presented in Lemma 1.1 from [35].

[*]Institute of Numerical Simulation, Hamburg University of Technology, D-21071 Hamburg, Germany (`joerg.lampe@tu-harburg.de`). The work was supported by Philips Research Germany and the German Federal Ministry of Education and Research (BMBF) under grant number 13N9079

[†]Delft Institute of Applied Mathematics, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands (`marielba.rojas@tudelft.nl`)

[‡]Dept. of Computational & Applied Mathematics, Rice University, Houston, TX 77005-1892, USA (`sorensen@caam.rice.edu`). The work was supported in part by NSF grants CCR-9988393 and ACI-0082645, and by AFOSR grant FA9550-09-1-0225.

[§]Institute of Numerical Simulation, Hamburg University of Technology, D-21071 Hamburg, Germany (`voss@tu-harburg.de`)

LEMMA 1.1. ([35]) *A feasible vector x^* is a solution to* (1.1) *with corresponding Lagrange multiplier λ^* if and only if x^* and λ^* satisfy $(H-\lambda^* I)x^* = -g$ with $H-\lambda^* I$ positive semidefinite, $\lambda^* \leq 0$, and $\lambda^*(\Delta - \|x^*\|) = 0$.*

Lemma 1.1 implies that all solutions of the TRS are of the form $x^* = -(H - \lambda^* I)^\dagger g + z$ for some $z \in \mathcal{N}(H - \lambda^* I)$, where † denotes the pseudo inverse and $\mathcal{N}$ the null space. If the Hessian matrix H is positive definite and if $\|H^{-1}g\| < \Delta$, problem (1.1) has a unique interior solution $x^* = -H^{-1}g$ with Lagrange multiplier $\lambda^* = 0$. In all other cases, there exists a (possibly non-unique) boundary solution satisfying $\|x^*\| = \Delta$ and $\lambda^* \leq \delta_1$, where δ_1 denotes the smallest eigenvalue of H. In this paper, we only consider the case of boundary solutions.

Non-unique boundary solutions can occur in the so-called *hard case*. This situation arises under three conditions: H is not positive definite, $\Delta \geq \|(H - \delta_1 I)^\dagger g\|$, and $g \perp \mathcal{S}_1$, with $S_1 = \mathcal{N}(H - \delta_1 I)$. In the hard case, $\lambda^* = \delta_1$. In practice, we usually see the *near* hard case with g nearly orthogonal to $\mathcal{S}_1$. The hard case is structural, i.e. its occurrence depends on the relationship between the problem data H, g, and Δ, and not on the method used to solve it. Trust-region subproblems where the hard case or near hard case are present are usually very challenging for numerical methods. In regularization, very difficult instances of the (near) hard case are common (cf. [29, 32]). When the hard case is *not* present, this is usually called the easy or standard case.

The TRS can be efficiently solved with the method proposed by Moré and Sorensen [22] when it is affordable to compute Cholesky factorizations of matrices of the form $H - \lambda I$. The method consists of a Newton iteration for finding a solution of a reformulation of the secular equation $(\Delta - \|x\|) = 0$ for $(H - \lambda I)x = -g$, on an interval where the function is almost linear. The method computes a solution that approximately satisfies the optimality conditions in Lemma 1.1.

In many large-scale applications, however, factoring or even forming $H - \lambda I$ is prohibitive, and matrix-free methods which rely only on matrix-vector products are needed. Methods for the TRS are usually classified as *approximate* (those that do not aim to satisfy the optimality conditions in Lemma 1.1), and *exact* (those that aim to approximately satisfy the optimality conditions).

Approximate methods for the large-scale TRS include the truncated Conjugate Gradients (CG) approach proposed by Steihaug [38] and Toint [39], the GLTR method of Gould, Lucidi, Roma, and Toint [10], and the Sequential Subspace Method (SSM) of Hager [11]. The truncated CG approach computes an approximate solution in a Krylov space and is particularly efficient in the context of trust-region methods. GLTR is a Lanczos version of the CG approach that includes preconditioning and a strategy for handling the hard case. A Fortran 90 implementation of GLTR is available in the Harwell subroutine library HSL [15]. In SSM, a sequence of four-dimensional subspaces $\mathcal{V}_k$ acts as an additional constraint $x \in \mathcal{V}_k$ for (1.1) (with the inequality replaced by equality, i.e. $\|x\| = \Delta$). The ingredients of each subspace are the current iterate x_k, the gradient of $\psi(x)$ at x_k, an estimate of an eigenvector corresponding to the smallest eigenvalue of H, and the first iterate generated by the sequential quadratic programming (SQP) method applied to problem (1.1). The first three vectors are sufficient for linear global convergence, proven by Hager and Park in [12]. By inserting the SQP iterate into the search space, the convergence is locally quadratic. The main cost in SSM is the approximate solution of a sequence of linear systems. In the current implementation, these linear systems are treated independently by variants of MINRES, i.e. there is no reuse of information from previous iterations except for the current iterate x_k itself.

Exact methods for the large-scale TRS include the method of Golub and von Matt [9], the semidefinite programming approach (SDP) of Rendl and Wolkowicz [27] and the extension by Fortin and Wolkowicz [6], the method of Sorensen [37], and LSTRS of Rojas, Santos, and Sorensen [30, 31]. The method of Golub and von Matt is based on Lanczos Bidiagonalization, matrix moments, and Gauss quadrature and derived under assumptions that exclude the occurrence of the hard case. The method is matrix-free but not limited-memory. The SDP, Sorensen, and LSTRS methods are all limited-memory techniques based on a reformulation of (1.1) as the following parameter-dependent eigenvalue problem:

$$B_\alpha y := \begin{pmatrix} \alpha & g^T \\ g & H \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} = \lambda \begin{pmatrix} 1 \\ x \end{pmatrix} =: \lambda y, \tag{1.2}$$

where the real parameter α has to be adjusted such that the solution of (1.1) can be read off an (appropriately normalized) eigenvector corresponding to the smallest eigenvalue of B_α.

The SDP approach formulates (1.1) as a semidefinite programming problem. It uses a primal-dual approach and the fact that strong duality holds to design an iteration for constructing the sequence $\{\alpha_k\}$. In this algorithm, the eigenvalue problems can become increasingly difficult if the smallest eigenvalue is not simple. An extension that can handle the hard case using an eigenvalue shift and deflation was proposed in Fortin and Wolkowicz [6]. Sorensen's method uses the eigenpair $(\lambda(\alpha_k), y(\alpha_k))$ corresponding to the smallest eigenvalue of B_{α_k} to construct a rational interpolant for a secular function. It then uses the interpolant to obtain α_{k+1}. The basic iteration is superlinearly convergent except in the hard case, when it must switch to a linearly-convergent scheme. LSTRS uses one or possibly *two* eigenpairs of B_α in a rational interpolation scheme. The method is a unified iteration that converges superlinearly in all cases. In the MATLAB implementation [31] of LSTRS the main eigensolver is the Implicitly Restarted Lanczos Method (IRLM) [36], implemented in ARPACK [21] and included in MATLAB as function `eigs`.

The acceleration strategy proposed in this paper is based on the fact that as the sequence of parameters $\{\alpha_k\}$ produced by the SDP, Sorensen or LSTRS methods converges, the matrices B_{α_k} converge as well and the eigenvalue problems do not vary significantly. The situation where a sequence of slightly-varying or convergent parameter-dependent problems must be solved arises in various areas of numerical computing and it suggests trying to improve convergence when solving the current problem by recycling as much information as possible from previous steps. Some examples of heavy reuse include homotopy methods for bifurcation problems [1] or for tracking invariant subspaces of dynamical systems [3], generalized Krylov subspaces in projection methods for linear [42, 43] and quadratic eigenvalue problems [14], recycling of Krylov spaces for sequences of linear systems [25, 41], for Newton-Krylov approaches for nonlinear systems of equations [28], and for Gauss-Newton method for nonlinear least squares problems in optical tomography [16].

In LSTRS (and SDP), information can be reused when solving the eigenvalue problems. For example, if the eigensolver is the IRLM, using the first Lanczos vector from the IRLM call in the previous LSTRS step as initial vector for the IRLM call in the current LSTRS step considerably accelerates convergence with respect to using a random initial vector for the IRLM at every LSTRS step. This seems to indicate that we may possibly obtain greater gains in performance if we could reuse more information from previous steps. Thus, eigensolvers that allow the recycling of more than one vector are attractive in this context.

Due to the simple dependence of the eigenvalue problem (1.2) on the parameter α, it is obvious that iterative projection methods that allow thick starts, i.e. starts with an initial search space of arbitrary size, are perfectly suited for reusing information when solving those problems. An iterative projection method determines an approximation to an eigenpair from the projection $V^T B_{\alpha_k} V z = \lambda z$ of the eigenproblem onto a subspace V of small dimension, expanding V if the approximation does not meet a specified accuracy. Methods of this type include the Nonlinear Arnoldi method for general nonlinear eigenvalue problems [40], and the Jacobi-Davidson method [34]. Both methods construct search spaces V which do not have a special structure (such as a basis of a Krylov space) and therefore can be started with any full-rank matrix $V \in \mathbb{R}^{n+1\times k}$.

In this paper, we discuss a modification of the LSTRS algorithm that uses the Nonlinear Arnoldi method for solving the eigenvalue problems. Numerical results demonstrate that LSTRS can be considerably accelerated by using the Nonlinear Arnoldi method as eigensolver.

Note also that the SDP algorithms [6, 27] can be accelerated in exactly the same way as presented here for LSTRS. This is illustrated with preliminary numerical results in Section 4.2. Similarly, SSM [11] could possibly be accelerated by combining it with the recycling approach for MINRES in [41] when solving the sequence of linear systems from SQP. Exploring this possibility for SSM is beyond the scope of this work.

This paper is organized as follows. Section 2 contains a brief description of the LSTRS algorithm. In Section 3, we describe the Nonlinear Arnoldi method and how it is combined with the LSTRS algorithm. Section 4 demonstrates with several examples the improvement obtained in LSTRS performance by recycling prior information. Concluding remarks are presented in Section 5.

2. The LSTRS method.

In this section, we briefly describe the LSTRS method. Its theoretical foundation and a discussion of its convergence properties is contained in [29, 30, 32]. A detailed description of the LSTRS algorithm with special emphasis on computational aspects can be found in [31].

Lemma 1.1 reveals the relationship between the TRS and the eigenvalue problem (1.2). For a real α, let $\lambda_1(\alpha)$ be the smallest eigenvalue of B_α and let y be a corresponding eigenvector.

We first consider the case when the first component of y is different from 0, and thus, can be scaled to be equal to one. For such an eigenvector $y = (1, x^T)^T$ we have:

$$\alpha - \lambda_1(\alpha) = -g^T x, \tag{2.1}$$

and

$$(H - \lambda_1(\alpha) I)x = -g. \tag{2.2}$$

Equation (2.2) demonstrates that, for any value of α, two of the conditions of Lemma 1.1 are automatically satisfied by $\lambda_1(\alpha)$ and y: $(H - \lambda_1(\alpha)I)x = -g$, and since the eigenvalues of H interlace those of B_α, $H - \lambda_1(\alpha)I$ must be positive semidefinite. Therefore, the parameter α must simply be adjusted to satisfy the two remaining conditions, $\lambda_1(\alpha) \leq 0$ and $\lambda_1(\alpha)(\Delta - \|x\|) = 0$.

Using (2.1), α can be updated according to

$$\alpha_+ = \lambda_1(\alpha) - g^T x = \lambda_1(\alpha) + g^T (H - \lambda_1(\alpha)I)^\dagger g =: \lambda_1(\alpha) + \phi(\lambda_1(\alpha)),$$

where $\phi(\lambda) = g^T (H - \lambda I)^\dagger g$ is a rational function with possible poles at the (distinct) eigenvalues of H, and therefore, too expensive to evaluate. LSTRS employs a rational interpolant $\hat{\phi}(\lambda)$ of $\phi(\lambda)$ based on the Hermitian data $\phi(\lambda) = -g^T x$ and $\phi'(\lambda) = g^T((H-\lambda I)^\dagger)^2 g = x^T x$. The new value α_+ is then computed as $\alpha_+ = \hat{\lambda} + \hat{\phi}(\hat{\lambda})$, where $\hat{\lambda}$ is determined such that $\hat{\phi}'(\hat{\lambda}) = \Delta^2$.

The previous approach assumes that there exists an eigenvector corresponding to the smallest eigenvalue of B_α whose first component can be scaled to one. The strategy breaks down if all eigenvectors associated with $\lambda_1(\alpha)$ have first component zero or very small. This can only happen when g is orthogonal (or nearly orthogonal) to $\mathcal{S}_1$, the eigenspace of H corresponding to δ_1. As discussed in Section 1, when $g \perp \mathcal{S}_1$ there is the possibility for the occurrence of the hard case (cf. [22]), and therefore this situation is called potential hard case in [30].

It was shown in [30] that in a potential hard case, for all values of α greater than a certain critical $\tilde{\alpha}$ all eigenvectors associated with $\lambda_1(\alpha)$ have first component zero. It was also shown that for any α there is a well-defined eigenvector of B_α depending continuously on α that can be safely normalized to have first component one. If either $g \perp \mathcal{S}_1$ and $\alpha \le \tilde{\alpha}$, or $g \not\perp \mathcal{S}_1$, then this eigenvector corresponds to the smallest eigenvalue $\lambda_1(\alpha)$. If $g \perp \mathcal{S}_1$ and α exceeds $\tilde{\alpha}$ by a small amount it is associated with the second smallest eigenvalue. In the potential hard case, it is this eigenpair that is used in LSTRS to construct the rational Hermitian interpolation $\hat{\phi}$ mentioned above.

We have sketched the essential ingredients of the LSTRS method showing that the main cost per iteration is the solution of the eigenvalue problem (1.2) with fixed α for the smallest eigenvalue, or the two smallest eigenvalues in the potential hard case, and corresponding eigenvectors. LSTRS uses a safeguarding strategy to ensure global convergence of $\{\alpha_k\}$ to its optimal value, and it employs the pairs $\{\lambda_1(\alpha_k), x_k\}$ and $\{\lambda_1(\alpha_{k-1}), x_{k-1}\}$ from the last two iterations when constructing the rational interpolant $\hat{\phi}$ to generate the next parameter $\alpha_{k+1} = \hat{\lambda} + \hat{\phi}(\hat{\lambda})$. It was shown in [30, Theorem 5.1] that the resulting algorithm is superlinearly convergent. The method is presented in Figure 2.1. More details can be found in [30, 31].

3. Nonlinear Arnoldi.

The IRLM is the method of choice for solving a large-scale symmetric eigenvalue problem for the smallest or two smallest eigenvalues and corresponding eigenvectors. However, in LSTRS a sequence of eigenproblems depending continuously on a convergent parameter has to be solved, and one should take advantage of information gained in previous iterations. The only freedom in Krylov subspace methods such as the IRLM is the choice of the initial vector, and consequently in the original LSTRS method the kth iteration is initialized by the first Lanczos vector of the $(k-1)$th iteration.

Iterative projection methods, on the other hand, are eigensolvers that allow for a thick start and therefore are able to use all information from previous iterations. Here the eigenvalue problem $(B_\alpha - \lambda I)y = 0$ under consideration is projected onto a subspace $\mathcal{V} = \operatorname{span}(V)$ of small dimension. If the Ritz pair (λ, Vz) of the projected problem $V^T(B_\alpha - \lambda I)Vz = 0$ corresponding to the smallest eigenvalue (or the second smallest in the hard case) does not meet a prescribed accuracy requirement then the subspace is expanded by a direction v which has a high approximation potential for the wanted eigenpair. As mentioned before, methods of this type are the Jacobi-Davidson method introduced by Sleijpen and van der Vorst [34] and the Nonlinear Arnoldi (NLArn) method, which was introduced in [40] for solving general nonlinear eigenvalue problems. In Jacobi-Davidson, $\mathcal{V}$ is expanded in such a way that the direction of inverse iteration at the current approximation is (approximately) contained in the next search space. In NLArn, $\mathcal{V}$ is expanded by an approximate Cayley transformation $v := PC \cdot (B_\alpha - \lambda I)Vz \approx (B_\alpha - \mu I)^{-1}(B_\alpha - \lambda I)Vz$ with a preconditioner PC which

```
Input: $H \in \mathbb{R}^{n\times n}$, symmetric; $g \in \mathbb{R}^n$; $\Delta > 0$; tolerances $(\epsilon_\Delta, \epsilon_{HC}, \epsilon_{Int}, \epsilon_\nu, \epsilon_\alpha)$.
Output: $x^*$, solution to TRS and Lagrange multiplier $\lambda^*$.

 1: Initialization
 2:   Compute $\delta_U \geq \delta_1$, initialize $\alpha_U$ and $\alpha_0$, set $k = 0$     % $\alpha_U \geq \alpha_k$
 3:   Compute eigenpairs $\{\lambda_1(\alpha_0), (\nu_1, u_1^T)^T\}$, and $\{\lambda_i(\alpha_0), (\nu_2, u_2^T)^T\}$ of $B_{\alpha_0}$
 4:   Initialize $\alpha_L$     % $\alpha_L \leq \alpha_k$
 5: repeat
 6:    Adjust $\alpha_k$ (might need to compute eigenpairs)
 7:    Update $\delta_U = \min\left\{\delta_U, \frac{u_1^T A u_1}{u_1^T u_1}\right\}$
 8:    if $\|g\||\nu_1| > \epsilon_\nu\sqrt{1-\nu_1^2}$ then
 9:       Set $\lambda_k = \lambda_1(\alpha_k)$, $x_k = \frac{u_1}{\nu_1}$, and update $\alpha_L$ or $\alpha_U$
10:    else
11:       Set $\lambda_k = \lambda_i(\alpha_k)$, $x_k = \frac{u_2}{\nu_2}$, and $\alpha_U = \alpha_k$
12:    end if
13:    Compute $\alpha_{k+1}$ by 1-point ($k = 0$) or 2-point interpolation scheme
14:    Safeguard $\alpha_{k+1}$ and set $k = k + 1$
15:    Compute eigenpairs $\{\lambda_1(\alpha_k), (\nu_1, u_1^T)^T\}$, and $\{\lambda_i(\alpha_k), (\nu_2, u_2^T)^T\}$ of $B_{\alpha_k}$
16: until convergence
```

FIG. 2.1. *The LSTRS Method.*

is kept fixed for several iterations, i.e. for several values of λ. In the latter case, the expanded space contains an approximation to the direction of residual inverse iteration [23]. It is obvious that due to the particularly simple dependence of B_α on the parameter α, the projected problem comes with no additional cost when changing the parameter α and reusing the search space.

Here we consider the use of NLArn to accelerate LSTRS. At the end of this section, we comment on the possibility of using Jacobi-Davidson instead.

Although NLArn is usually more expensive than the IRLM when solving a single eigenvalue problem, the heavy reuse of information gained from previous steps leads to a significant speedup in LSTRS, i.e. all necessary information for solving $B_{\alpha_{k+1}}y = \lambda y$ is already contained in the subspace that has been built for solving $B_{\alpha_k}y = \lambda y$.

A similar technique has been successfully applied in regularized total least squares (RTLS) [17, 19] for accelerating the RTLS solver [33] which is based on a sequence of quadratic eigenvalue problems. Another method for RTLS presented in [26] which is based on a sequence of linear eigenproblems has also been substantially accelerated in [18, 20].

Here, we have used the NLArn algorithm in Figure 3.1 in LSTRS for solving

$$T_k(\lambda)y = (B_0 + \alpha_k N - \lambda I)y = \left(\begin{pmatrix} 0 & g^T \\ g & H \end{pmatrix} + \alpha_k e_1 e_1^T - \lambda I\right) y = 0. \qquad (3.1)$$

The Nonlinear Arnoldi method allows thick starts in line 1, i.e. when solving $T_k(\lambda)y = 0$ in step k of the LSTRS method, algorithm 3.1 is started with the orthonormal basis V that was used in the preceding iteration when determining the

```
Input: $T_k$ as in (3.1), ($H, g, \alpha_k$ from LSTRS), $\epsilon \in (0,1)$.
Output: $\{\mu, u\}$ an eigenpair corresponding to the smallest eigenvalue of $T_k$
 1: Start with initial basis $V$, $V^TV = I$
 2: For fixed $\alpha_k$ find smallest eigenvalue $\mu$ of $V^TT_k(\mu)Vz = 0$
    and corresponding eigenvector $z$
 3: Determine positive definite preconditioner $PC \approx T_k^{-1}(\mu)$
 4: Set $u = Vz$, $r = T_k(\mu)u$
 5: while $\|r\|/\|u\| > \epsilon$
 6:     $v = PCr$
 7:     $v = v - VV^Tv$
 8:     $\tilde{v} = v/\|v\|$, $V = [V, \tilde{v}]$
 9:     Find smallest eigenvalue $\mu$ of $V^TT_k(\mu)Vz$
        and corresponding eigenvector $z$
10:     Set $u = Vz$, $r = T_k(\mu)u$
11: end while
```

FIG. 3.1. *The Nonlinear Arnoldi Method.*

solution $y_{k-1} = Vz$ of $V^TT_{k-1}(\lambda)Vz = 0$. So all search spaces of previous problems are kept.

The projected problem in the kth iteration

$$V^TT_k(\mu)Vz = \left(V^TB_0V + \alpha_k(e_1^TV)^T(e_1^TV) - \mu I\right) z = 0 \tag{3.2}$$

can be constructed directly from the previous step since the matrices V, V^TB_0V and $v_1 = V(1,:)$ are known. Within the iteration, these matrices are obtained on-the-fly by appending one column to V and one column and row to V^TB_0V, respectively. This update relies on matrix-vector products only, and does not require the matrix B_0 explicitly.

If $v = 0$ after Step 7 is completed, the iteration must halt. However, this is a fortunate event as it would imply $r = 0$. This is because $v = 0$ would imply $Vs = PCr$ for some s. Hence, $r^TPCr = r^TVs = 0$ by virtue of Step 9 which in turn implies $r = 0$ since PC is positive definite. Therefore, the test at Step 5 would have already halted the iteration with a numerical solution.

The LSTRS method combined with NLArn can be executed with low and fixed storage requirements. For memory allocation purposes, a maximal dimension $p \ll n$ of the search space span(V) can be set in advance, and if in the course of the LSTRS method the dimension of V reaches p, then NLArn can be restarted with a subspace spanned by a small number q of eigenvectors corresponding to the smallest eigenvalues of $T_k(\lambda)$. Since the value of p is usually modest, orthogonality of the basis of V is maintained and re-orthogonalization is not required.

LSTRS requires an eigenvector associated with the smallest eigenvalue of $T_k(\lambda)$. An additional eigenvector is needed only if the first eigenvector cannot be safely scaled to have one as a first component. The implicitly restarted Lanczos method approximates eigenvectors corresponding to extreme eigenvalues simultaneously, and therefore in the original version of LSTRS, two eigenvectors are returned by `eigs` at every iteration. NLArn aims at one eigenpair at a time. In the NLArn algorithm in Figure 3.1, this is the smallest one. If a second eigenvector is needed the search space V can be further extended now aiming at the second smallest eigenvalue μ in statement 9.

A few further comments are in order:

— To solve the very first eigenproblem with the Nonlinear Arnoldi it is recommended to put some useful information in the starting basis V. An orthonormal basis of the Krylov subspace $\mathcal{K}_\ell(B_0, e_1)$ with $\ell \approx 5$ a suitable choice.

— The projected eigenproblems in line 2 and 9 can be solved by a dense solver for all eigenvalues; in the numerical experiments MATLAB's `eig` has been used.

— For general nonlinear eigenproblems, NLArn usually requires a preconditioner. In this application, PC would need to be positive definite if it is to approximate the inverse of a positive definite matrix, $PC \approx T_k^{-1}(\mu)$. For the test examples in Section 4 the method always worked fine without it, i.e. $PC = I$.

— For a least squares problem with norm constraint, the explicit form of the matrix $H = A^T A$ is not needed to determine the projected matrix $V^T B_0 V$, since this can be updated according to $V^T B_0 V = ([-b, A]V)^T([-b, A]V) - \|b\|^2 v_1^T v_1$ (recalling that v_1 is the first row of V).

The advantage of using the Nonlinear Arnoldi method in LSTRS over the implicitly restarted Lanczos method is due to the fact that thick starts are possible. This holds true also for other iterative projection approaches like the Jacobi-Davidson method where the search space span(V) is expanded by an approximate solution of the correction equation

$$\big(I - \frac{uu^T}{u^T u}\big)T_k(\mu)\big(I - \frac{uu^T}{u^T u}\big)v = T_k(\mu)u, \quad v \perp u. \tag{3.3}$$

However, solving (3.3) will usually be much more expensive than the Nonlinear Arnoldi expansion $v = PC \cdot T_k(\mu)u$ where only two (or even one in case $PC = I$) matrix-vector products are needed at every iteration. For really huge problems where storage is critical and only coarse preconditioners are available, using Jacobi-Davidson could be beneficial. In that case, the dimension of the search space built by Nonlinear Arnoldi can become quite large, but it can be kept much smaller in Jacobi-Davidson if the correction equation (3.3) is solved very accurately, which can be done by a Krylov solver with short recurrence.

4. Numerical Results.

In order to evaluate the performance of LSTRS using Nonlinear Arnoldi as eigensolver, we used a modified version of LSTRS that allows the computation of one eigenpair at a time. If a second eigenpair is needed, the eigensolver is called again to compute it.

Numerical experiments were performed on three different problem classes, namely: regularization problems, shifted Laplacian problems, and problems with H of the form $H = UDU^T$, with D a diagonal matrix and U a Householder matrix.

In Section 4.1, we compare the performance of LSTRS on regularization problems using the IRLM and NLArn as eigensolvers. In Section 4.2, we compare the performance of several methods for the large-scale TRS on the shifted Laplacian, UDU^T, and two regularization problems with different levels of ill-posedness. In Section 4.3, we present results for LSTRS on a large-scale image restoration problem.

In all experiments, we chose the parameter Δ such that the solution of problem (1.1) is located on the boundary of the feasible set $\|x\| = \Delta$. Note that for TRS arising in trust-region methods, Δ is determined in the outer iteration. In regularization, the optimal Δ is either known from the Physics of the problem or has to be determined for example, by means of the L-curve. This could be another source of recycling of information, since a few TRS have to be solved in order to determine the value of Δ.

As in [31], all experiments were performed using the idea of meeting certain *targets*. The results shown correspond to the input settings that yielded the best performance for each method, i.e. with those settings, the methods met the target at the lowest cost in terms of storage and matrix-vector products. The target for each experiment is described in the corresponding section. The settings used are described in Appendix A. The experiments were carried out in MATLAB R2008b on a MacBookPro with a 2.66 GHz processor and 4 GB of RAM, running the Mac OS X version 10.6.4 (Snow Leopard) operating system. The floating-point arithmetic was IEEE standard double precision with machine precision $2^{-52} \approx 2.2204 \times 10^{-16}$.

4.1. Regularization Problems. The regularization test problems were taken from the Regularization Tools package [13]. Most of the problems in that package are discretizations of Fredholm integral equations of the first kind, which are typically very ill-posed.

Regularized solutions were computed by solving the following quadratically-constrained least squares problem:

$$\min_x \frac{1}{2}\|Ax - b\|^2 \quad \text{subject to} \quad \|x\| \leq \Delta, \tag{4.1}$$

where $A \in \mathbb{R}^{m\times n}$, $m \geq n$, and $b \in \mathbb{R}^m$. The matrix A is a discretized operator from an ill-posed problem and is typically very ill-conditioned. Problem (4.1) is equivalent to a trust-region problem of type (1.1) with $H = A^TA$ and $g = -A^Tb$. Hence the matrix H is at least positive semidefinite or in the full-rank case, positive definite. No noise was added to the vector b since the absence of noise yields more difficult trust-region problems for which the potential (near) hard case is present in a multiple instance (cf. [31, 32]).

In all tests, $m = n = 1000$ and $\Delta = \|x_{true}\|$, where x_{true} was the true solution to the inverse problem provided in [13]. Note that although in practice the exact value $\|x_{true}\|$ is usually not known, good estimates for Δ are available in many important applications (cf. [2, 8]). We compared the following eigensolvers: IRLM+C (the IRLM combined with the Chebyshev spectral transformation described in [31, 32]), IRLM+H (the IRLM combined with the heuristics described in [31]), and NLArn.

As in [31], the *target* in these experiments was a relative error in the approximate solution x with respect to x_{true} of the same order as in the IRLM+H solution. Thus, the results for IRLM+H were obtained first. The settings for LSTRS combined with IRLM+H were chosen ad hoc, as they often are in practice, after several trial runs. Most of the choices were the same as in [31]. The parameters for IRLM+C and NLArn were then adjusted to meet the target. All settings can be found in Section A.1.

The results are presented in Table 4.1. For each eigensolver, we report number of matrix-vector products (MVP), number of vectors (VEC), the optimality measure $\frac{\|(H-\lambda I)x+g\|}{\|g\|}$, and the relative error $\frac{\|x-x_{true}\|}{\|x_{true}\|}$. The results are presented graphically in Figures 4.1 and 4.2 (where problem names have been abbreviated). We can observe in Figure 4.1 the tremendous improvement obtained in LSTRS performance when NLArn was used as eigensolver. When compared with IRLM+H, we observe that for problem

heat, mild, savings of approximately 35% in MVP were obtained and for **deriv2, ex.2** the savings are approximately 42%. For the rest of the problems, the savings in MVP were 60% or higher. When compared with IRLM+C, savings of at least 80% in MVP were obtained. Moreover, for all test problems except **heat, mild**, NLArn required the same or fewer number of vectors than both variants of the IRLM.

Problem	Eigensolver	MVP	VEC	$\frac{\|(H-\lambda I)x+g\|}{\|g\|}$	$\frac{\|x-x_{true}\|}{\|x_{true}\|}$
baart	IRLM+C	600	10	4.23e-15	8.63e-02
	IRLM+H	84	10	1.02e-15	8.63e-02
	NLArn	18	10	7.95e-14	8.63e-02
deriv2, ex.1	IRLM+C	2605	8	2.52e-02	5.66e-01
	IRLM+H	555	8	2.52e-02	5.67e-01
	NLArn	217	7	2.50e-02	5.84e-01
deriv2, ex.2	IRLM+C	3609	13	7.00e-03	4.91e-01
	IRLM+H	256	13	1.65e-01	6.32e-01
	NLArn	148	5	6.95e-03	3.42e-01
foxgood	IRLM+C	370	10	3.74e-15	3.71e-02
	IRLM+H	114	10	2.09e-14	3.71e-02
	NLArn	18	10	2.96e-11	3.71e-02
heat, mild	IRLM+C	313	8	1.52e-04	2.50e-03
	IRLM+H	105	8	1.48e-04	3.60e-03
	NLArn	68	10	5.42e-05	5.03e-03
heat, severe	IRLM+C	1029	8	7.29e-05	7.05e-02
	IRLM+H	551	8	7.09e-06	5.48e-02
	NLArn	112	8	5.37e-05	8.05e-02
i_laplace, ex.1	IRLM+C	1079	10	4.20e-06	1.99e-01
	IRLM+H	279	10	1.53e-05	2.33e-01
	NLArn	137	9	7.96e-06	3.28e-01
i_laplace, ex.3	IRLM+C	970	10	6.77e-07	4.26e-02
	IRLM+H	198	10	6.39e-07	4.36e-02
	NLArn	52	10	5.55e-06	6.69e-02
phillips	IRLM+C	493	10	6.85e-05	9.98e-03
	IRLM+H	333	10	4.01e-09	8.97e-03
	NLArn	92	10	5.30e-06	9.88e-03
shaw	IRLM+C	551	10	1.11e-14	5.85e-02
	IRLM+H	135	10	1.09e-13	5.85e-02
	NLArn	36	10	2.67e-10	5.86e-02

TABLE 4.1
LSTRS on regularization problems, $m = n = 1000$.

4.2. Comparison of TRS solvers. In this section, we compare LSTRS with other methods for the large-scale trust-region subproblem. The methods used for comparisons were the Sequential Subspace Method (SSM) of Hager [11], the Semidefinite Programming approach (SDP) of Fortin and Wolkowicz [6], and the Generalized Lanczos Trust Region method (GLTR) of Gould, Lucidi, Roma and Toint [10]. Note that only LSTRS, SSM and SDP are limited-memory methods. For SSM, results are re-

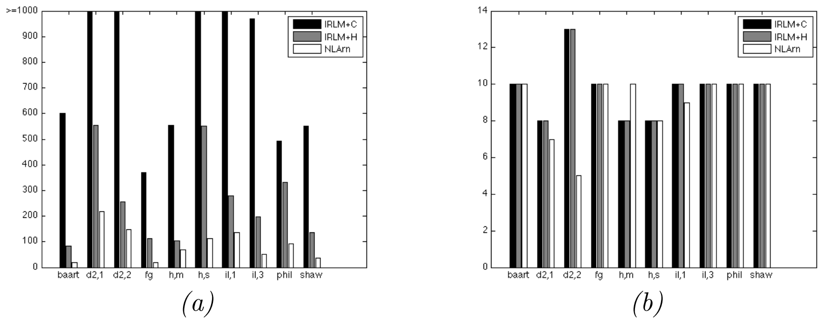

FIG. 4.1. *(a) Number of matrix-vector products (MVP); (b) number of vectors (VEC) required by IRLM+C (black), IRLM+H (gray), and NLArn (white). LSTRS on regularization problems, $m = n = 1000$.*

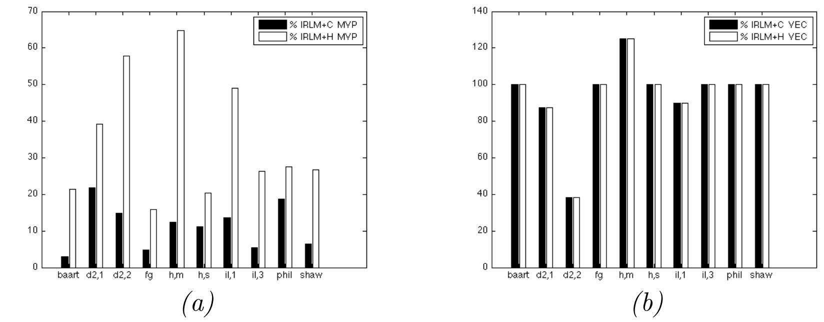

FIG. 4.2. *(a) NLArn MVP as % of IRLM+C MVP (black) and IRLM+H (white); (b) NLArn VEC as % of IRLM+C VEC (black) and IRLM+H VEC (white). LSTRS on regularization problems, $m = n = 1000$.*

ported only for the two matrix-free variants SSM and SSM$_d$. Two eigensolvers (IRLM and NLArn) were used in combination with LSTRS and SDP.

The results for all methods, except those using NLArn, were taken from [31] since there have been no changes in the GLTR, SSM, and SDP methods since those experiments were performed. In all experiments, we use the same targets as in [31].

We used MATLAB implementations of LSTRS, SSM and SDP, and a Fortran 90 implementation of GLTR. It is important to note that the four codes are at a different stage of development. The GLTR code is the routine HSL_VF05 of the HSL library [15]. The SSM and SDP codes are research implementations provided by their authors for the purpose of these comparisons. In particular, the SDP code is still at a very early stage of development. The double precision version of HSL_VF05 was used whereas the results for the MATLAB codes were obtained under MATLAB 6.0. We were able to reproduce the results reported in [31] for LSTRS and SDP under MATLAB R2008b. SSM uses mexfiles that do not run under this new version of MATLAB. The results for LSTRS+NLArn and SDP+NLArn were obtained under MATLAB R2008b.

4.2.1. Laplacian Problems. Here $H = L - 5I$, with L the standard 2-D discrete Laplacian on the unit square based upon a 5-point stencil with equally-spaced mesh points. The dimension of the problem was $n = 1024$. The trust-region radius was fixed at $\Delta = 100$. Ten related trust-region subproblems were solved differing only in the vector g which was randomly generated with entries uniformly distributed on $(0,1)$. Problems with and without hard case were studied. To generate the hard case, the vector g was orthogonalized against the eigenvector q_1 corresponding to the smallest eigenvalue of H. A noise vector of norm 10^{-8} was then added to g. Note that for $\Delta = 100$, we have $\|(H - \delta_1 I)^\dagger g\| < \Delta$, which is a necessary condition for the hard case.

The *target* in this experiment was a prescribed level for the optimality measure $\frac{\|(H-\lambda I)x+g\|}{\|g\|}$. Here, $\frac{\|(H-\lambda I)x+g\|}{\|g\|} \leq 10^{-5}$ was required. The parameters for GLTR, LSTRS+IRLM, SSM, and SDP were as in [31]. All settings are described in Section A.2.1.

Average results for the ten problems are shown in Table 4.2 where we report number of matrix-vector products (MVP), number of vectors (VEC), and the optimality measure (target) $\frac{\|(H-\lambda I)x+g\|}{\|g\|}$. We can observe that in the standard case the performance of LSTRS+NLArn is closer to SSM's (approximately 15% more MVP as opposed to 52% more MVP required by LSTRS+IRLM). In the hard case, LSTRS+NLArn required 50% more vectors than the other methods and the lowest number of MVP of all (approximately 20% fewer MVP than LSTRS+IRLM, the second best of all methods for these problems).

Preliminary experiments with SDP+NLArn indicate that NLArn can greatly improve SDP's performance. In the standard case, only 14% of the MVP required by SDP+IRLM were needed by SDP+NLArn. Although the number of vectors was larger, SDP+NLArn could reach the target, while SDP+IRLM could not. In the hard case, SDP+NLArn was not able to solve any of the ten problems. However, as we mentioned before, SDP is not a mature code at this moment. We expect that an improved SDP code can take full advantage of efficient eigenvalue calculations with NLArn.

An interesting quantity to look at in the hard case is $\rho = \frac{|\lambda_* - \delta_1|}{|\delta_1|}$ since, as discussed in Section 1, λ^* must equal δ_1 in this case (in exact arithmetic). The average value of ρ was `1.45e-03` for LSTRS+IRLM, and `6.72e-11` for LSTRS+NLArn. This shows that using NLArn yields a very accurate λ^* in the hard case for these problems.

4.2.2. UDU^T **family.** In these problems, the matrix $H \in \mathbb{R}^{1000\times 1000}$ was of the form $H = UDU^T$ with D a diagonal matrix with elements $d_1, \ldots, d_n$, and $U = I - 2uu^T$ with $u^T u = 1$. The elements of D were randomly generated with a uniform distribution on $(-5, 5)$, then sorted in nondecreasing order and d_1 was set to -5. Both vectors u and g were randomly generated with entries selected from a uniform distribution on $(-0.5, 0.5)$. The vector u was normalized to have unit length.

The eigenvectors of H are of the form $q_i = e_i - 2uu_i$, $i = 1, \ldots, n$, with e_i the ith canonical vector and u_i the ith component of the vector u. The vector g was orthogonalized against $q_1 = e_1 - 2uu_1$, and a noise vector was added to g. Finally, g was normalized to have unit norm. The noise vectors had norms 10^{-2} and 10^{-8} for the standard and hard case, respectively. To construct suitable examples for both cases we computed $x_{min} = -(H - d_1 I)^\dagger g$ and $\Delta_{min} = \|x_{min}\|$, and then set $\Delta = 0.1\Delta_{min}$ in the standard case and $\Delta = 5\Delta_{min}$ in the hard case. One fact makes this problem extremely difficult to solve: typically, x_{min} is almost orthogonal to q_1 but has huge components γ_i in the directions $q_i, i = 2, 3, 4, 5$ and only very small components in eigendirections corresponding to large eigenvalues of H. To construct an appropriate solution in the hard case, the vectors

$$\begin{pmatrix} 0 \\ q_1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 \\ x_{min} \end{pmatrix} \approx \begin{pmatrix} 1 \\ \sum_{i=2}^{5} \gamma_i q_i \end{pmatrix}$$

Method	MVP	VEC	$\frac{\|(H-\lambda I)x+g\|}{\|g\|}$
LSTRS+IRLM	127.1	10.0	2.32e-06
LSTRS+NLArn	79.9	10.0	1.78e-06
SSM	67.3	10.0	9.53e-07
SSM$_d$	67.3	10.0	9.53e-07
SDP+IRLM	595.0	10.0	3.17e-05
SDP+NLArn	89.0	17.0	2.91e-06
GLTR	81.6	41.3	8.56e-06

Standard Case

Method	MVP	VEC	$\frac{\|(H-\lambda I)x+g\|}{\|g\|}$
LSTRS+IRLM	252.6	10.0	6.91e-06
LSTRS+NLArn	201.4	15.0	7.28e-06
SSM	377.9	10.0	1.42e-06
SSM$_d$	377.9	10.0	1.42e-06
SDP+IRLM	2023.8	10.0	5.76e-02
SDP+NLArn	*	*	*
GLTR	151.8	76.4	8.37e-06

Hard Case

TABLE 4.2
Average results for the 2-D Laplacian, $n = 1024$.

have to be properly separated. This is a highly demanding task since the vectors q_i correspond to the same cluster around $\delta_1 = -5$.

The *target* in this experiment was $\frac{\|(H-\lambda I)x+g\|}{\|g\|} \leq 10^{-5}$. The parameters for GLTR, LSTRS+IRLM, SSM, and SDP were as in [31]. All settings are described in Section A.2.2.

Average results for ten related problems, differing only in the vector g, are shown in Table 4.3 where we report number of matrix-vector products (MVP), number of vectors (VEC), and the optimality measure (target) $\frac{\|(H-\lambda I)x+g\|}{\|g\|}$.

We observe that in the standard case the performance of LSTRS+NLArn is essentially the same as SSM's and closer to SSM$_d$'s than LSTRS+IRLM's (approximately 48% more MVP as opposed to almost 4 times more MVP required by LSTRS+IRLM). In the hard case, LSTRS+NLArn required 35% fewer MVP than SSM and only 30% of the total number of MVP required by LSTRS+IRLM. The preconditioned variant of SSM (SSM$_d$) outperformed all methods for these problems. LSTRS+NLArn required the lowest number of MVP of all unpreconditioned methods in all cases.

Regarding SDP+NLArn, we see again savings in the number of MVP for both cases. In the standard case, savings of 43% in MVP were obtained with 20% more vectors. In the hard case, savings of 85% in MVP were obtained with approximately 3.5 times the number of vectors. As we mentioned before, our experiments with SDP are preliminary. Other settings could possibly yield large savings with fewer vectors. A careful numerical investigation of the SDP+NLArn performance is beyond the scope of this paper.

We look again at the quantity $\rho = \frac{|\lambda_* - \delta_1|}{|\delta_1|}$ which was `2.74e-04` for LSTRS+IRLM, and `5.02e-06` for LSTRS+NLArn. This shows that NLArn computes λ_* more accurately than LSTRS+IRLM in the hard case for these problems.

4.2.3. Inverse Heat Equation. In order to compare the methods on regularization problems, we chose problem **heat** from [13]. The problem is a discretized version of the Inverse Heat Equation, which arises, for example, in the inverse heat conduction problem of determining the temperature on the surface of a body from transient measurements of the temperature at a fixed location in the interior [4]. The equation is a Volterra integral equation

$$\gamma(y) = \int_0^y \mathcal{K}(y,t)\phi(t)dt, \quad 0 \le y \le 1, \tag{4.2}$$

where $\mathcal{K}(y,t) = k(y-t)$, with $k(t) = \frac{t^{-3/2}}{2\kappa\sqrt{\pi}} \exp\left(-\frac{1}{4\kappa^2 t}\right)$. The parameter κ controls the degree of ill posedness. We performed experiments with a mildly ill-posed problem ($\kappa = 5$) and a severely ill-posed one ($\kappa = 1$). The dimension was $n = 1000$.

To compute regularized solutions, we solve the constrained least squares problem (4.1) as in Section 4.1. As before, Regularization Tools provided the matrix A, the vector b, as well as x_{true}, a discretized version of the analytical solution of the continuous problem. In this case, 20% of the singular values of the matrix A were zero to working precision. As in Section 4.1, in order to generate a more difficult TRS, no noise was added to the vector b.

The *target* in this experiment was a relative error $\frac{\|x - x_{true}\|}{\|x_{true}\|}$ of the same level as in the SSM solution. Thus, the SSM solution was computed first (using the best possible settings after several trial runs), and then the solutions with the other methods were computed. Table 4.4 shows the best results (relative error) at the lowest cost (storage and matrix-vector products) for each method. We report number of matrix-vector products (MVP), number of vectors (VEC), the optimality measure $\frac{\|(H-\lambda I)x+g\|}{\|g\|}$, and $\frac{\|x - x_{true}\|}{\|x_{true}\|}$. The settings used are described in Section A.2.3.

The results show that LSTRS+NLArn outperformed all methods in terms of MVP. For the mildly ill-posed problem, we obtained savings of approximately 25% with respect to the second best results (LSTRS+IRLM's) using 50% more vectors. In the severely ill-posed case, savings of approximately 40% were obtained with respect to the second best results (SSM$_d$'s) using 20% more vectors.

Method	MVP	VEC	$\frac{\|(H-\lambda I)x+g\|}{\|g\|}$
LSTRS+IRLM	90.2	10.0	2.95e-06
LSTRS+NLArn	35.9	9.0	9.62e-06
SSM	35.2	10.0	1.35e-06
SSM$_d$	24.1	10.0	9.90e-07
SDP+IRLM	950.4	10.0	9.65e-07
SDP+NLArn	541.2	12.0	1.05e-06
GLTR	36.8	18.9	7.37e-06

Standard Case

Method	MVP	VEC	$\frac{\|(H-\lambda I)x+g\|}{\|g\|}$
LSTRS+IRLM	954.1	24.0	9.65e-06
LSTRS+NLArn	247.1	60.0	2.84e-06
SSM	445.1	24.0	1.91e-06
SSM$_d$	130.4	24.0	9.59e-07
SDP+IRLM	1720.8	24.0	7.86e-06
SDP+NLArn	266.4	80.0	3.85e-07
GLTR	634.6	317.8	7.64e-06

Hard Case

TABLE 4.3

Average results for UDU^T, $n = 1000$.

4.3. An Image Restoration Problem. In this section, we present results on a large-scale image restoration problem where the goal is to recover an image from blurred and noisy data. The problem was constructed in the following way. A digital photograph of an art gallery in Paris was blurred with the routine **blur** from [13]. Then, a random Gaussian noise vector was added to the blurred image. The regularization problem was as in Section 4.1, where A was the blurring operator and b was the blurred and noisy image generated as above and stored columnwise as a one-dimensional array. The noise level was 1%. The original image was a digital array of 256×256 pixels which gives a TRS of dimension $n = 65536$.

Method	MVP	VEC	$\frac{\|(H-\lambda I)x+g\|}{\|g\|}$	$\frac{\|x-x_{true}\|}{\|x_{true}\|}$
LSTRS+IRLM	265	8	9.12e-07	6.13e-04
LSTRS+NLArn	194	12	6.71e-06	9.41e-04
SSM	700	8	2.99e-09	2.41e-04
SSM$_d$	649	8	2.74e-09	4.57e-04
SDP+IRLM	5700	8	2.73e-07	3.63e-04
SDP+NLArn	1208	8	7.07e-05	2.52e-03

Mildly Ill-Posed Case

Method	MVP	VEC	$\frac{\|(H-\lambda I)x+g\|}{\|g\|}$	$\frac{\|x-x_{true}\|}{\|x_{true}\|}$
LSTRS+IRLM	551	8	7.05e-06	5.49e-02
LSTRS+NLArn	126	10	2.55e-05	5.38e-02
SSM	512	8	1.81e-07	3.75e-02
SSM$_d$	215	8	2.04e-07	2.25e-02
SDP+IRLM	4600	8	2.27e-04	2.08e-01
SDP+NLArn	2415	15	3.15e-06	5.56e-02

Severely Ill-Posed Case

TABLE 4.4
Problems: **heat, mild** *and* **heat, severe**, $m = n = 1000$.

The performance of LSTRS using the IRLM and NLArn as eigensolvers is shown in Table 4.5. We observe that LSTRS+NLArn computed a restored image with similar accuracy (in 2-norm) to the one computed with LSTRS+IRLM at a much lower number of MVP (56% fewer MVP), using the same number of vectors. Interestingly enough, LSTRS was able to compute a boundary solution (BS) when using NLArn as eigensolver, while only a quasi-optimal solution (QO) could be computed when using the IRLM. A boundary solution is preferred in regularization. The original image, the blurred and noisy data, and the restored images are shown in Figure 4.3.

5. Conclusions.

LSTRS is a suitable algorithm for solving the Large-Scale Trust-Region Subproblem. The main computation in LSTRS is the solution of a sequence of eigenvalue problems. Since by construction of the algorithm this sequence is convergent, it is highly advantageous to use the information gathered while solving one eigenproblem in the solution of the next. The Nonlinear Arnoldi method can efficiently

Eigensolver	MVP	VEC	$\frac{\|(H-\lambda I)x+g\|}{\|g\|}$	$\frac{\|x-x_{true}\|}{\|x_{true}\|}$	Solution Type
IRLM+C	201	7	1.01e-03	1.06e-01	QO
NLArn	89	7	7.99e-04	1.01e-01	BS

TABLE 4.5
Image Restoration Problem: **paris**, $n = 65536$.

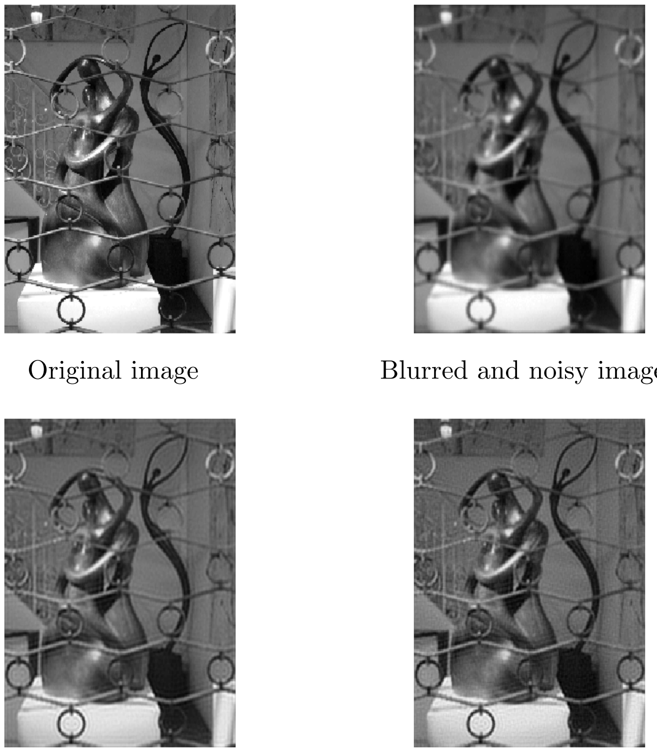

FIG. 4.3. *Image Restoration Problem:* **paris**, $n = 65536$.

use all previously-obtained information. In all our experiments, using the Nonlinear Arnoldi method instead of the Implicitly Restarted Lanczos Method, significantly reduced the computational cost. Further improvement in performance could possibly be obtained by the use of suitable preconditioners within NLArn.

Acknowledgment. We would like to thank Bill Hager and Henry Wolkowicz for providing the MATLAB codes implementing their methods for the purpose of comparisons. We also thank the anonymous referees for their constructive comments and questions which helped improve an earlier version of the manuscript. M. Rojas would like to thank Professor Kees Roos and the Optimization Group of the Software Technology Department at the Delft University of Technology in The Netherlands for their hospitality during an extended visit during which part of this work was carried

out.

REFERENCES

[1] E.L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*, volume 45 of *Classics in Applied Mathematics.* SIAM, Philadelphia, 2003.

[2] Mario Bertero and Patricia Bocacci. *Introduction to Inverse Problems in Imaging.* Institute of Physics, Bristol, 1998.

[3] W.-J. Beyn, A. Champneys, E. Doedel, W. Govaerts, Yu.A. Kuznetsov, and B. Sandstede. Numerical continuation and computation of normal forms. In B. Hasselblatt, B. Fiedler, and A.B. Katok, editors, *Handbook of Dynamical Systems*, volume 2, pages 149 – 219. North–Holland, Amsterdam, 2001.

[4] Carasso, A. Determining surface temperatures from interior observations. *SIAM J. Appl. Math.*, 42:558–574, 1982.

[5] Andrew R. Conn, Nicholas N.I.M. Gould, and Philippe L. Toint. *Trust Region Methods.* SIAM, Philadelphia, 2000.

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

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

[8] M.S. Gockenbach and W.W. Symes. Duality for inverse problems in wave propagation. In F. Santosa L. Biegler, T. Coleman and A. Conn, editors, *Large Scale Optimization*, pages 37 – 61. New York Institute for Mathematics and its Applications, 1997.

[9] G.H. Golub and U. von Matt. Quadratically constrained least squares and quadratic problems. *Numer. Math.*, 59:561 – 580, 1991.

[10] N.I.M. Gould, S. Lucidi, M. Roma, and P.L. Toint. Solving the trust-region subproblem using the lanczos method. *SIAM J. on Optimization*, 9:504–525, 1999.

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

[12] W.W. Hager and S. Park. Global convergence of SSM for minimizing a quadratic over a sphere. *Math. Comp.*, 74:1413 – 1423, 2005.

[13] P.C. Hansen. Regularization Tools version 4.0 for Matlab 7.3. *Numer. Algo.*, 46:189–194, 2007.

[14] U.B. Holz, G.H. Golub, and K.H. Law. A subspace approximation method for the quadratic eigenvalue problem. *SIAM J. Matrix Anal. Appl.*, 26:498 – 521, 2005.

[15] HSL. HSL 2004: A collection of Fortran codes for large scale scientific computation, 2004. Available from `www.cse.clrc.ac.uk/nag/hsl`.

[16] M.E. Kilmer and E. de Sturler. Recycling subspace information for diffuse optical tomography. *SIAM J. Sci. Comput.*, 27:2140 – 2166, 2006.

[17] J. Lampe and H. Voss. On a quadratic eigenproblem occuring in regularized total least squares. *Comput. Stat. Data Anal.*, 52/2:1090 – 1102, 2007.

[18] J. Lampe and H. Voss. A fast algorithm for solving regularized total least squares problems. *Electr. Trans. Numer. Anal.*, 31:12 – 24, 2008.

[19] J. Lampe and H. Voss. Global convergence of RTLSQEP: a solver of regularized total least squares problems via quadratic eigenproblems. *Math. Modelling Anal.*, 13:55 – 66, 2008.

[20] J. Lampe and H. Voss. Solving regularized total least squares problems based on eigenproblems. *Taiwanese J. Math.*, 14:885 – 909, 2010.

[21] R.B. Lehoucq, D.C. Sorensen, and C. Yang. *ARPACK Users' Guide. Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM, Philadelphia, 1998.

[22] J.J. Moré and D.C. Sorensen. Computing a trust region step. *SIAM J. Sci. Stat. Comput.*, 4:553 – 572, 1983.

[23] A. Neumaier. Residual inverse iteration for the nonlinear eigenvalue problem. *SIAM J. Numer. Anal.*, 22:914 – 923, 1985.

[24] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer, New York, 1999.

[25] M.L. Parks, E. de Sturler, G. Mackey, D.J. Johnson, and S. Maiti. Recycling Krylov spaces for sequences of linear systems. *SIAM J. Sci. Comput.*, 28:1651 – 1674, 2006.

[26] R.A. Renaut and H. Guo. Efficient algorithms for solution of regularized total least squares. *SIAM J. Matrix Anal. Appl.*, 26:457 – 476, 2005.

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

[28] F. Risler and C. Rey. Iterative accelerating algorithms with Krylov subspaces for the solution to large–scale nonlinear problems. *Numer. Algorithms*, 23:1 – 30, 2000.

[29] M. Rojas. *A large-scale trust-region approach to the regularization of discrete ill-posed prob-*

lems. PhD thesis, Rice University, Houston, Texas, 1998.

[30] M. Rojas, S.A. Santos, and D.C. Sorensen. A new matrix-free algorithm for the large-scale trust-region subproblem. *SIAM J. Optim.*, 11:611 – 646, 2000.

[31] M. Rojas, S.A. Santos, and D.C. Sorensen. Algorithm 873: LSTRS: MATLAB software for large-scale trust-region subproblems and regularization. *ACM Trans. Math. Software*, 34(2):Article 11, 28 pages, 2008.

[32] M. Rojas and D.C. Sorensen. A trust-region approach to the regularization of large-scale discrete forms of ill-posed problems. *SIAM J. Sci. Comput.*, 23:1842 – 1860, 2002.

[33] D.M. Sima, S. Van Huffel, and G.H. Golub. Regularized total least squares based on quadratic eigenvalue problem solvers. *BIT Numerical Mathematics*, 44:793 – 812, 2004.

[34] G.L. Sleijpen and H.A. van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17:401 – 425, 1996.

[35] D.C. Sorensen. Newton's method with a model trust-region modification. *SIAM J. Numer. Anal.*, 19:409 – 426, 1982.

[36] D.C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM Journal on Matrix Analysis and Applications*, 13(1):357–385, 1992.

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

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

[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, London, 1981. Academic Press.

[40] H. Voss. An Arnoldi method for nonlinear eigenvalue problems. *BIT Numerical Mathematics*, 44:387 – 401, 2004.

[41] S. Wang, E. de Sturler, and H. Paulino. Large–scale topology optimization using preconditioned Krylov subspace methods with recycling. *Internat. J. Numer. Meth. Engrg.*, 69:2441 – 2468, 2007.

[42] T. Zhang, G. H. Golub, and K. H. Law. Eigenvalue perturbation and generalized Krylov subspace methods. *Appl. Numer. Math.*, 27:185 – 202, 1998.

[43] T. Zhang, G. H. Golub, and K. H. Law. Subspace iterative methods for eigenvalue problems. *Linear Algebra Appl.*, 294(1-3):239 – 258, 1999.

Appendix A. Settings for Numerical Experiments.

A.1. Regularization Problems. LSTRS+IRLM: `epsilon.HC = 1e-16` and `epsilon.Int = 0` were used to favor a boundary solutions; `epsilon.Delta = 1e-2`; `lopts.max_eigentol = 0.7` was used for problems **deriv2, ex.1**; **heat, mild**; **i_laplace, ex.1**; and **i_laplace, ex.3**; `lopts.max_eigentol = 0.4` was used for the rest. Default values were used for the remainder of the parameters. The initial vector for the first call to the IRLM was $v_0 = (1, \ldots, 1)^T/\sqrt{n+1}$. Two vectors were kept at each implicit restart.

For NLArn, the LSTRS parameters were as before with the exception of `epsilon.Delta = 1e-4` for problem **phillips**. The tolerance used in NLArn's stopping criterion was `1e-10`. The values of (p, ℓ, q) with p the maximum dimension of the search space, ℓ the dimension of the initial search space, and q the number of eigenvectors kept at restart, were as follows: $(7, 5, 2)$ for **deriv2, ex.1**, $(5, 5, 2)$ for **deriv2, ex.2**, $(10, 9, 2)$ for **heat, mild**, $(8, 3, 2)$ for **heat, severe**, $(9, 6, 2)$ for **i_laplace, ex.1**, $(10, 3, 2)$ for **phillips**, and $(10, 5, 2)$ for the rest.

A.2. Comparison of TRS solvers.

A.2.1. Shifted Laplacian Problems. For LSTRS+IRLM, SSM, SSM$_d$ and SDP+IRLM the number of vectors was 10. In the IRLM, 8 shifts were applied in each implicit restart. The remainder parameters were as follows.

LSTRS+IRLM: `epsilon.Delta = 1e-5` and `epsilon.HC = 1e-11` in the standard case, `epsilon.Delta = epsilon.HC = 1e-11` in the hard case, and `lopts.deltaU = 'mindiag'` and `lopts.alpha0 = lopts.deltaU`. Defaults values were used for the remainder of the parameters. The initial vector for ARPACK was $v_0 = (1, \ldots, 1)^T/\sqrt{n+1}$. These choices are the same as in [30].

LSTRS+NLArn: The tolerance used in NLArn's stopping criterion was `1e-7` in all cases. In the standard case: `epsilon.Delta = 1e-6` and `epsilon.HC = 1e-11`, and (p, ℓ, q) was $(10, 2, 10)$. In the hard case: `epsilon.Delta = epsilon.HC = 1e-10`, and (p, ℓ, q) was $(15, 15, 6)$ for eight of the problems and $(15, 15, 7)$ for the remaining two problems. In both cases, default values were used for the remainder of the parameters.

SSM, SSM$_d$: $\|(H - \lambda I)x + g\|$ was required to be less than or equal to `tol = 1e-5` and one of the initial vectors in a relevant Krylov subspace was chosen as the vectors of all ones.

SDP+IRLM: the tolerance for the duality gap was set to `1e-10` in the standard case and `1e-9` in the hard case.

SDP+NLArn: in the standard case, the tolerance used in NLArn's stopping criterion was `1e-10` and (p, ℓ, q) was $(17, 12, 2)$. In the hard case, SDP+NLArn did not converge for any of the settings tried.

GLTR: the tolerance for the optimality measure was set to `1e-5` and the required fraction of the optimal value of the objective function was set to 1. The desired optimality level could not be achieved for lower fractions of the optimal objective value. This was also the case for the other experiments.

A.2.2. UDU^T **problems.** For LSTRS+IRLM, SSM, SSM$_d$ and SDP+IRLM, the number of vectors was 10 in the easy case and 24 in the hard case. Other parameters were as follows.

LSTRS+IRLM: `epsilon.Delta = 1e-4`, `epsilon.HC = 1e-10` in all cases. In the standard case, `lopts.deltaU = 'mindiag'`, `lopts.alpha0 = lopts.deltaU`, `lopts.maxeigentol = 0.2`, and 8 shifts were applied in each implicit restart. In the hard case, `lopts.deltaU = -4.5`, `lopts.alpha0 = 'min'`, `lopts.maxeigentol = 0.03`, and 12 shifts were applied in each implicit restart. The initial vector for ARPACK was $v_0 = (1, \dots, 1)^T / \sqrt{n+1}$. Default values were used for the remainder of the parameters.

LSTRS+NLArn: in all cases, the tolerance used in NLArn's stopping criterion was `1e-7`, `epsilon.Delta = 1e-4` and `epsilon.HC = 1e-10`. In the standard case: (p, ℓ, q) was $(9, 5, 2)$. In the hard case: (p, ℓ, q) was $(60, 40, 40)$ for nine of the problems and $(60, 50, 50)$ for the remaining problem.

SSM, SSM$_d$: $\|(H - \lambda I)x + g\|$ was required to be less than or equal to `1e-5` and one of the initial vectors in a relevant Krylov subspace was chosen as the vectors of all ones.

SDP+IRLM: the tolerance for the duality gap was set to `1e-11` in the standard case and `1e-12` in the hard case.

SDP+NLArn: the tolerance for the duality gap was set to `1e-11` in the standard case and `1e-12` in the hard case. The tolerance used in NLArn's stopping criterion was `1e-10` in the standard case and `1e-7` in the hard case. The values of (p, ℓ, q) were $(12, 5, 2)$ in the standard case and $(80, 50, 50)$ in the hard case.

GLTR: the tolerance for the optimality measure was set to `1e-5` in the standard case and `1e-7` in the hard case. The required fraction of the optimal value of the objective function was set to 1.

A.2.3. Inverse Heat Equation. LSTRS+IRLM: `lopts.heuristics = 1`; `epsilon.Delta = 1e-3` and `lopts.maxeigentol = 0.7` for **heat, mild** and `epsilon.Delta = 1e-2` and `lopts.maxeigentol = 0.4` for **heat, severe**. The initial vector for ARPACK was $v_0 = (1, \dots, 1)^T / \sqrt{n+1}$. Default values were used for the remainder of the parameters.

LSTRS+NLArn: the tolerance used in NLArn's stopping criterion was `1e-10` for **heat, mild** and `1e-5` for **heat, severe**; `epsilon.Delta = 1e-7` for **heat, mild** and `1e-3` for **heat, severe**; `epsilon.HC = 1e-16` was used for both problems. For **heat, mild**: (p, ℓ, q) was $(12, 9, 5)$. For **heat, severe**: (p, ℓ, q) was $(10, 3, 2)$.

SSM, SSM$_d$: $\|(H - \lambda I)x + g\|$ was required to be less than or equal to `1e-8` for **heat, mild** and `1e-7` for **heat, severe**. One of the initial vectors in a relevant Krylov subspace was chosen as the vectors of all ones.

SDP+IRLM: the tolerance for the duality gap was set to `1e-7` for the mildly ill-posed problem and `1e-8` for the severely ill-posed problem.

SDP+NLArn: the tolerance for the duality gap was set to `1e-7` for **heat, mild** and `1e-10` for **heat, severe**. The tolerance used in NLArn's stopping criterion was `1e-10` for both problems. The values of (p, ℓ, q) were $(8, 4, 2)$ for **heat, mild** and $(15, 5, 2)$ for **heat, severe**.

A.3. An Image Restoration Problem. LSTRS+IRLM: the eigensolver was the IRLM combined with a Chebyshev spectral transformation, `epsilon_Delta = 1e-2` and `epsilon_HC = 1e-4`. The initial vector was the vectors of all ones. Default values were used for the remainder of the parameters.

LSTRS+NLArn: the LSTRS parameters were as before. For NLArn, the tolerance in the stopping criterion was `1e-10` and (p, ℓ, q) was $(7, 2, 4)$.