

PARALLEL VARIABLE DISTRIBUTION FOR TOTAL LEAST SQUARES

HONGBIN GUO AND ROSEMARY A. RENAUT

ABSTRACT. A novel parallel method for determining an approximate total least squares (TLS) solution is introduced. Based on domain distribution, the global TLS problem is partitioned into several dependent TLS subproblems. A convergent algorithm using the parallel variable distribution technique (Ferris and Mangasarian, 1994) is presented. Numerical results support the development and analysis of the algorithms.

Keywords: Total least squares, Variable distribution, Errors-in-variables regression.

AMS(MOS) subject classification: 65F20, 90C06, 90C26, 90C30

1. INTRODUCTION

Total least squares (TLS), which has existed in statistics under the name “errors-in-variables regression” for a long time as a natural generalization of least squares (LS), was introduced to numerical specialists in 1973 by Golub([7]). In the last two decades, the TLS problem has been researched from many different numerical perspectives, [14, 2, 8, 5, 16, 13, 20], excepting the development of a parallel algorithm. Among these references, [14] includes a complete introduction and analysis of basic algorithms. Here, we consider large-scale well-posed problems; for ill-posed TLS problems refer to [8, 5, 13, 20].

In contrast to the standard LS model for the TLS model both the matrix A and the right hand side b in the overdetermined linear system $Ax \approx b$, $A \in R^{m \times n}$, $m > n$, are assumed to contain errors. The TLS solution x_{TLS} solves the following optimization problem:

$$(1.1) \quad \min_{E, f} \|[E, f]\|_F^2 \quad \text{subject to } (A + E)x = b + f,$$

Date: September 14, 2004.

This work was partially supported by the Arizona Center for Alzheimer’s Disease Research, by NIH grant EB 2553301 and for the second author by NSF CMG-02223.

Email:hb_guo@asu.edu.

Email:renaut@asu.edu.

where $\|\cdot\|_F$ denotes the Frobenius norm. If $U\Sigma V^T$ is a singular value decomposition of the augmented matrix $[A, b]$, the smallest singular value σ_{n+1} is simple and $v(n+1, n+1) \neq 0$ (*generic*), then $x_{\text{TLS}} = -v(1:n, n+1)/v(n+1, n+1)$ and $[E, f] = -\sigma_{n+1}u_{n+1}v_{n+1}^T$, where u_i, v_i are the columns of matrices U and V , respectively. Moreover, x_{TLS} satisfies,

$$(1.2) \quad f = \frac{Ax_{\text{TLS}} - b}{1 + \|x_{\text{TLS}}\|^2},$$

$$(1.3) \quad E = -f \cdot x_{\text{TLS}}^T,$$

and minimizes the sum of squared normalized residuals,

$$(1.4) \quad x_{\text{TLS}} = \operatorname{argmin}_x \phi(x) = \operatorname{argmin}_x \frac{\|Ax - b\|^2}{1 + \|x\|^2},$$

[10, 14]. Here ϕ is the Rayleigh quotient (RQ) of matrix $[A, b]^T[A, b]$. For convenience we denote the TLS problem for the augmented matrix $[A, b]$ by $\text{TLS}(A, b)$ and say that x_{TLS} solves the problem $\text{TLS}(A, b)$.

For a large-scale problem, the singular value decomposition (SVD) calculation is expensive. Because we only need one singular vector, iterative methods can be used to reduce the computational cost. For example, inverse power, inverse Chebyshev iteration [14] and Rayleigh quotient iteration (RQI) [2] could be used. Related methods for computing smallest singular value(s) (and vector(s)) are Jacobi-Davidson [22], trace minimization [21] and inverse Rayleigh Ritz iteration [12]. We also note that block Lanczos and block Davidson methods, based on parallel sparse matrix-vector multiplication, were discussed for computing the singular subspace associated with the smallest singular values [18]. Here, rather than computing the singular vectors, we use the TLS minimization formulation (1.4), and solve the optimization problem by application of the parallel variable distribution (PVD) approach of Ferris and Mangasarian, [4]. This approach is suitable for both sparse and dense data structures. The specialization of PVD to the linear least squares problem was presented in [19], [3] and a more general approach was presented in [6].

To apply the PVD approach we introduce the decomposition of the space R^n as a Cartesian product of lower dimensional subspaces $R^{n_j}, j = 1, \dots, p$, where $\sum_{j=1}^p n_j = n$. Accordingly, any vector $x \in R^n$ is decomposed as $(Px)^T = (x_1^T, x_2^T, \dots, x_p^T)$, where P is a permutation matrix. Matrices A and E are partitioned

consistently

$$\begin{aligned} AP &= [A_1, A_2, \dots, A_p], \\ EP &= [E_1, E_2, \dots, E_p]. \end{aligned}$$

Without loss of generality, we ignore the permutation matrix P in the analysis and theoretical development of the algorithms. For ease we introduce the notation \bar{x}_i to be the complement of x_i , namely the vector x with zeros in block i .

Consistent with application of the PVD approach we assume for the iterative algorithm that there are p processors, each of which can update a different block-component, x_j . Each of these processors may be designated a *slave* processor, which is coordinated by a single *master* processor. The slaves solve the local problems and are coordinated for solution of the global problem by the master processor. If there are insufficient available processors, the algorithms are modified appropriately to consider the separate processes, rather than processors. If $\sum_{j=1}^p R^{n_j} = R^n$ and $\sum_{j=1}^p n_j > n$, the spatial decomposition is designated as an *overlapped* decomposition, otherwise it is *without overlap*. In the theoretical analysis we assume that the subproblems are not overlapped, although our numerical experiments consider both situations.

In the presentation of the basic algorithms we do not give all details of the parallel implementation for the linear algebra operations. For example, calculation of matrix vector products, where the matrix is distributed over several processors with local memory, requires local computation, global communication and global update. Such operations are by now well documented, see for example [10], and dependent on the local architecture employed for problem solution.

We reemphasize that the focus of this work is the development of a domain decomposition approach and the study of its feasibility. The remaining sections of this paper are organized as follows. We provide the development of the algorithms and analyze their properties in Section 2. Convergence analysis is presented in Section 3, computational considerations in Section 4, numerical experiments in Section 5 and conclusions in Section 6.

2. ALGORITHMS

2.1. Development. Domain decomposition applied to the Rayleigh quotient formulation for the TLS problem, (1.4), suggests the local problems are given by

$$(2.5) \quad z_i = \operatorname{argmin}_{z \in R^{n_i}} \frac{\|A_i z - b(\bar{x}_i)\|_2^2}{1 + \|z\|_2^2 + \sum_{j \neq i} \|x_j\|_2^2},$$

where

$$(2.6) \quad b(\bar{x}_i) = b - \sum_{j \neq i} A_j x_j = r(x) + A_i x_i, \quad r(x) = b - Ax.$$

Setting

$$(2.7) \quad \beta_i^2 = 1 + \sum_{j \neq i} \|x_j\|_2^2, \quad w_i = z_i / \beta_i \quad \text{and} \quad \tilde{b}_i = b(\bar{x}_i) / \beta_i,$$

(2.5) is replaced by

$$(2.8) \quad w_i = \operatorname{argmin}_{w \in R^{n_i}} \frac{\|A_i w - \tilde{b}_i\|_2^2}{1 + \|w\|_2^2},$$

which now looks like (1.4) for $\text{TLS}(A_i, \tilde{b}_i)$. In other words, w_i solves

$$\min_{\tilde{E}_i, \tilde{f}_i} \|[\tilde{E}_i, \tilde{f}_i]\| \quad \text{subject to} \quad (A_i + \tilde{E}_i)w_i = \tilde{b}_i + \tilde{f}_i.$$

Equivalently z_i is the solution of the weighted TLS subproblem :

$$(2.9) \quad \min_{E_i, f_i} \| [E_i, \beta_i^2 f_i] \|_F^2 \quad \text{subject to} \quad (A_i + E_i)z_i = b(\bar{x}_i) + \beta_i^2 f_i.$$

2.2. Global Update. The parallel algorithm requires both a mechanism to find local solutions $Y_i, = 1 \dots p$, where Y_i is vector x with component x_i replaced by z_i , for each local problem, the *parallelization phase*, and the process by which a global update is obtained, the *synchronization phase*. Of course, if parallelism is not required, then a Gauss-Seidel update can be formed, in which the local problems are solved using the most up-to-date information from each subproblem. On the other hand, for true parallelism, there are various approaches for the global update at the synchronization step. These mimic the alternatives presented in [19] for the multisplitting solution of the least squares problem.

Before listing all synchronization approaches, we introduce a mechanism for an optimal global update at synchronization.

Theorem 2.1. *Let $D \in R^{n \times p}$ and $\gamma \in R^p$, and define*

$$\psi(\gamma) = \frac{\|A(x + D\gamma) - b\|_2^2}{1 + \|x + D\gamma\|_2^2}.$$

Then $\gamma_{\min} = \operatorname{argmin}_{\gamma \in \mathbb{R}^p} \psi(\gamma)$ satisfies

$$(2.10) \quad (D^T A^T A D - \psi \cdot D^T D) \gamma = D^T A^T r + \psi \cdot D^T x,$$

where $r = b - Ax$.

Proof. We introduce the notation $\psi = \psi_1/\psi_2$, where

$$(2.11) \quad \psi_1(\gamma) = \|A(x + D\gamma) - b\|_2^2 \geq 0,$$

$$(2.12) \quad \psi_2(\gamma) = 1 + \|x + D\gamma\|_2^2 \geq 1.$$

Then at a stationary point

$$\psi'_1 - \psi \cdot \psi'_2 = 0.$$

Replacing ψ'_1, ψ'_2 by the corresponding gradient vectors, we immediately obtain condition (2.10). \square

This result shows how to find an optimal update using the updated local solutions, specifically it leads to the updates S_1 and S_p given below.

Synchronization approaches: At iteration k , solution $x^{(k)}$ is updated by one of the following four methods.

- **Block Jacobi (BJ)** Given the local solutions $z_i^{(k)}$, $i = 1 \dots p$, form the block update $x^{(k)} = z^{(k)}$.
- **Convex Update:** Form the global update using the convex combination

$$(2.13) \quad x_i^{(k)} = (1 - \alpha_i) x_i^{(k-1)} + \alpha_i z_i^{(k)}$$

for each block i , where the weights α_i satisfy $0 < \alpha_i < 1$, with $\sum_{i=1}^p \alpha_i = 1$, and without any optimality imposed it is practical to choose $\alpha_i = 1/p$.

- **Line Search (S_1):** Define search direction $d^{(k)} = z^{(k)} - x^{(k-1)}$, and find scalar α such that solution $x^{(k)} = x^{(k-1)} + \alpha d^{(k)}$ solves the global minimization:

PVDTLS- S_1

$$(2.14) \quad \min_{\alpha \in \mathbb{R}} \psi(\alpha), \quad \psi(\alpha) = \frac{\|A(x^{(k-1)} + \alpha d^{(k)}) - b\|_2^2}{1 + \|x^{(k-1)} + \alpha d^{(k)}\|_2^2}.$$

The solution α is a root of the quadratic equation $\varrho_1 \alpha^2 + \varrho_2 \alpha + \varrho_3 = 0$, where

$$\begin{aligned} \varrho_1 &= (d^{(k)})^T x^{(k-1)} \cdot \|Ad^{(k)}\|^2 + (Ad^{(k)})^T r^{(k-1)} \cdot \|d^{(k)}\|^2, \\ \varrho_2 &= \|Ad^{(k)}\|^2 \cdot (1 + \|x^{(k-1)}\|^2) - \|r^{(k-1)}\|^2 \cdot \|d^{(k)}\|^2, \\ \varrho_3 &= -(Ad^{(k)})^T r^{(k-1)} \cdot (1 + \|x^{(k-1)}\|^2) - \|r^{(k-1)}\|^2 \cdot (d^{(k)})^T x^{(k-1)}, \end{aligned}$$

and is chosen such that $\psi(\alpha)$ is minimal. Here we introduce the use of the residual $r^{(k)} = b - Ax^{(k)}$.

- **p -dimensional update (S_p):** Search in a p -dimensional subspace \mathcal{S}_p as follows: Consider the update $x^{(k)} = x^{(k-1)} + \sum_{i=1}^p \gamma_i d_i^{(k)}$, where the $d_i^{(k)} = [0, \dots, 0, (z_i^{(k)})^T - (x_i^{(k-1)})^T, 0, \dots, 0]^T$ are the components of the global search direction, and the parameters γ_i determine the weight in each subdirection. We use $D^{(k)}$ to denote the matrix with columns $d_i^{(k)}$, γ the vector with components γ_i , and solve the global minimization:

$$(2.15) \quad \text{PVDTLS-}S_p \quad \min_{\gamma \in \mathbb{R}^p} \frac{\|A(x^{(k-1)} + D^{(k)}\gamma) - b\|_2^2}{1 + \|x^{(k-1)} + D^{(k)}\gamma\|_2^2}.$$

By Theorem 2.1, the conditions for optimality yield

$$(2.16) \quad \begin{aligned} & ((D^{(k)})^T A^T A D^{(k)} - \psi \cdot (D^{(k)})^T D^{(k)})\gamma \\ & = (D^{(k)})^T A^T r^{(k-1)} + \psi \cdot (D^{(k)})^T x^{(k-1)}, \end{aligned}$$

where

$$(2.17) \quad \psi = \frac{\|AD^{(k)}\gamma - r^{(k-1)}\|_2^2}{1 + \|x^{(k-1)} + D^{(k)}\gamma\|_2^2}.$$

Because p is assumed small relative to n we can assume that problems (2.16) and (2.17) are small. Again γ is chosen such that $\psi(\gamma)$ is minimal.

Remark 2.1. *A secular equation with ψ as variable can be derived from equations (2.16) and (2.17). Because we want to minimize the function $\psi(x)$, the smallest root of the secular equation is of interest. Thus the iteration may be initialized with $\psi_0 = 0$.*

Remark 2.2. *Observe that the two search techniques, S_1 and S_p , are more general forms of the update (2.13). In particular S_1 corresponds to (2.13) with $\alpha_i = \alpha$, but without the constraint on each α_i , and S_p finds an optimal set of α_i in (2.13) each step to minimize the global objective. Clearly, S_1 is the specification of S_p for the case $p = 1$. As presented the optimal updates S_1 and S_p do not impose convexity which is important for convergence in the LS case, but we will see is irrelevant for TLS because of its lack of convexity.*

Remark 2.3. *We do not expect that use of the BJ update at each step would generate a convergent algorithm. However, as in [19], it may speed up the iteration if at a given step the BJ update is adopted when it has generated a smaller objective*

value than that given by (2.13), and the sequence of objective function values is decreasing.

2.3. Algorithm.

Algorithm 1. (PVDTLS) Given a tolerance τ and an initial vector $x^{(0)}$, set $k = 0$, and calculate $r^{(0)} = b - Ax^{(0)}$. Compute solutions $x^{(k)}$ iteratively until $|\phi(x^{(k)}) - \phi(x^{(k-1)})|/\phi(x^{(k)}) < \tau$ as follows:

- (1) While not converged **Do**
 - (a) **Parallelization (slave processor i):**
 - (i) $k = k + 1$.
 - (ii) Calculate $b(\bar{x}_i^{(k)}) = r^{(k-1)} + A_i x_i^{(k-1)}$.
 - (iii) Find solution $w_i^{(k)}$ of TLS(A_i, \tilde{b}_i), and calculate z_i using (2.7).
 - (b) **Synchronization (master processor):**
 - (i) Use a global update algorithm to find y_{opt} and update $x^{(k)} = x^{(k-1)} + y_{\text{opt}}$, where
- (2.18)
$$\phi(x^{(k)}) < \phi(x^{(k-1)}).$$
- (ii) Test for convergence. **Break** if converged.
 - (iii) **Else** Update necessary data values over all processors. Continue.
- End Do**

3. CONVERGENCE ANALYSIS

Discussion of convergence for the PVDTLS algorithms is far more complex than that for the least squares PVD for which the objective function is convex. Consider the following low-dimensional example, for which $\phi(x)$ is illustrated in Figure 1.

(3.19)

$$[A, b] = \begin{bmatrix} \frac{-2}{\sqrt{6}} & 0 & 0 & \frac{\sqrt{3}}{3} \\ \frac{1}{\sqrt{6}} & \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{3}}{3} \\ \frac{1}{\sqrt{6}} & \frac{-\sqrt{2}}{2} & 0 & \frac{\sqrt{3}}{3} \\ 0 & 0 & -1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & & & \\ & 0.5 & & \\ & & 0.1 & \\ 0 & 0 & 0 & \end{bmatrix} * \begin{bmatrix} \frac{-2}{\sqrt{6}} & 0 & \frac{\sqrt{3}}{3} \\ \frac{1}{\sqrt{6}} & \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \\ \frac{1}{\sqrt{6}} & \frac{-\sqrt{2}}{2} & \frac{\sqrt{3}}{3} \end{bmatrix}^T.$$

We see that not only is $\phi(x)$ not convex but it also possesses saddle points. Figure 2 illustrates two curves which cross the saddle point $(0, 1)$ of $\phi(x)$ for (3.19). Obviously, if the iteration starts from $x^{(0)} = (0, 1)$, the update will not move from the saddle point, neither in the parallelization nor the synchronization step, (in the latter $D^{(1)} = 0$). Thus, even if the given algorithms converge to some point, this

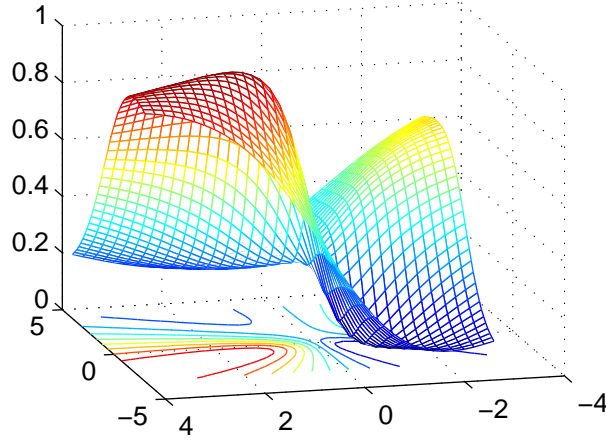


FIGURE 1. Function $\phi(x)$ for (3.19) with maximum point $(2, -1, 1)$, minimum point $(-1, -1, 0.1)$ and saddle point $(0, 1, 0.5)$.

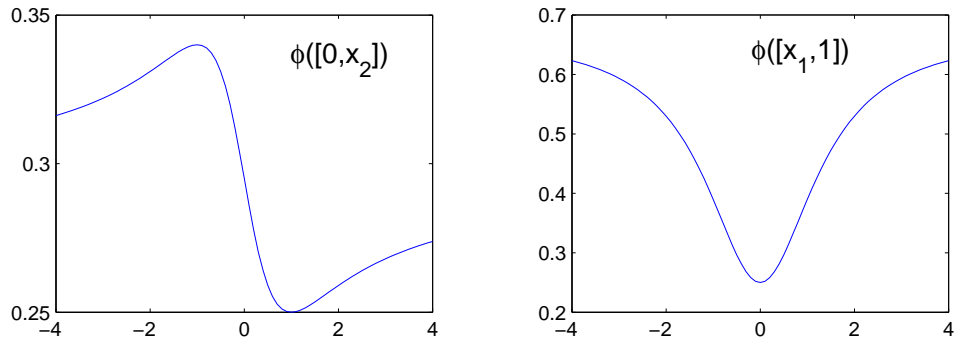


FIGURE 2. Curves crossing saddle point $(0, 1)$ in coordinate directions, $\phi([0, x_2])$ and $\phi([x_1, 1])$.

point may not be at the global minimum, and may be a saddle point which traps the subsequent iterative updates.

The difficulty exhibited by the example illustrated in Figure 1 is general, as demonstrated by the following result for the stationary points of the non-concave function $\phi(x)$.

Lemma 3.1. (FACT 1-8 in [17]) *The Rayleigh quotient of a symmetric matrix is stationary at, and only at, the eigenvectors of the matrix.*

Lemma 3.2. ([20]) *If the extreme singular values of the matrix $[A, b]$ are simple then $\phi(x)$ has one unique maximum point, one unique minimum point and $n-1$ saddle points.*

In particular, this result shows that if the iteration defined by the preceding algorithm starts at a stationary point, the iteration need not converge to the global minimum of the objective.

3.1. Convergence Proof. First, note that when we relate a right singular vector $v_i(1 : n + 1)$ of matrix $[A, b]$ to $(x^T, -1)^T$ we assume $v_i(n + 1) \neq 0$.

We use the notation $\nabla_i^2 \phi(x)$ for the n_i -dimensional *Hessian* matrix with respect to $x_i \in R^{n_i}$. Also, for a bounded sequence $\{x^{(k)}\}$ with accumulation point x^* , we use the indices $k_j, j = 1, 2, \dots$ for the subsequence $\{x^{(k_j)}\} \rightarrow x^*$, when $j \rightarrow \infty$.

Theorem 3.3. *Suppose $\{x^{(k)}\}$ is the sequence generated by Algorithm 1. Then*

- (1) $\{\phi(x^{(k)})\}$ converges.
- (2) If $\{x^{(k)}\}$ is bounded, any accumulation point, x^* , is a stationary point of $\phi(x)$ and $\nabla_i^2 \phi(x^*) \geq 0$ on the subspace R^{n_i} for $i = 1, \dots, p$.
- (3) If $\{x^{(k)}\}$ is bounded and the singular values of matrix $[A, b]$ are distinct $\{x^{(k)}\}$ converges.

Proof. (1) By the synchronization step it is guaranteed that the objective function decreases with $\{x^{(k)}\}$. Moreover, the objective function is bounded below by 0 and thus converges.

- (2) Because $\{x^{(k)}\}$ is bounded there exists at least one accumulation point x^* . Denote $\phi(x^*)$ by ϕ^* . Suppose that this accumulation point is not at a minimum of ϕ with respect to its first block component. Then there exists $x_1^{\text{new}} \in R^{n_1}$ such that

$$(3.20) \quad \phi(x_1^{\text{new}}, x_{\bar{1}}^*) < \phi^*.$$

Here $\bar{1}$ denotes the complement of element 1 in $\{1, \dots, p\}$, i.e., $\bar{1} = \{2, \dots, p\}$.

By the requirement of objective function decrease, however, we must have

$$\begin{aligned} \phi(x^{(k_j)}) &\leq \phi(x^{(k_{j-1}+1)}) \\ &\leq \phi(z^{(k_{j-1}+1)}, x_{\bar{1}}^{(k_{j-1})}) \\ &\leq \phi(x_1^{\text{new}}, x_{\bar{1}}^{(k_{j-1})}). \end{aligned}$$

In the limit $j \rightarrow \infty$ this yields

$$\phi^* \leq \phi(x_1^{\text{new}}, x_1^*),$$

which contradicts (3.20). The same argument applies with respect to the other block components.

- (3) We first prove that $\{x^{(k)}\}$ has a unique accumulation point provided that the singular values of matrix $[A, b]$ are distinct. Suppose the contrary, then there exist two accumulation points x^* and x^{**} , which by the previous statement are stationary points of $\phi(x)$. Then, by Lemma 3.1, $((x^*)^T, -1)^T$ and $((x^{**})^T, -1)^T$ are two eigenvectors of matrix $[A, b]^T[A, b]$ corresponding to two distinct eigenvalues $\sigma_k^2 < \sigma_l^2$. Moreover, $\phi(x^*) = \sigma_k^2 < \sigma_l^2 = \phi(x^{**})$. But, by statement (1), $\{\phi(x^{(k)})\}$ converges, $\phi(x^*) = \phi(x^{**})$. Thus $\{x^{(k)}\}$ must converge to a unique accumulation point. Otherwise, because of the boundedness of the sequence we could construct another accumulation point, which by the above is not possible.

□

4. COMPUTATIONAL CONSIDERATIONS

We assess the maximal theoretical efficiency of PVDTLS for solution of (1.4) through analysis of its computational cost in comparison to the traditional serial direct method and its parallelization using PARPACK [23]. Comparison with indirect techniques, [14], is not so immediate because the cost of each indirect algorithm depends on both its specific implementation, and the condition of the underlying problem. See, for example, [2] for a thorough discussion of approaches for implementation of the RQI to solve (1.4). Also, the use of an indirect algorithm for (2.5) confounds the discussion in this paper, for which the major intent is to assess the overall viability and stability of PVDTLS for large scale applications. Thus, while we do not specifically exclude that indirect techniques can be useful for solving each of the local problems, here we propose the use of a very efficient SVD update algorithm.

Measuring computational cost only in terms of number of *flops*, without consideration of any hardware related issues for measure passing, or memory or cache usage, the cost of the direct SVD solution of (1.4) is $C_s = 2mn^2 + 12n^3$, see Algorithm 12.3.1 in [10]. It is efficient to use this same direct SVD algorithm for the solution of each (2.5), because the first n_i columns of $\tilde{A}_i = [A_i, \tilde{b}_i]$ are fixed over

all outer iterations. Hence, having calculated the initial SVD, the SVD update algorithm, [11], can be utilized, yielding a total local cost $C_l(K)$, where K is the number of outer iterations to convergence,

$$(4.21) \quad C_l(K) \approx 2n_i^2(m + 6n_i) + K(2n_i(m + 5n_i) + 4n_im),$$

Here the last term is the cost of updating $b(\bar{x}_i)$ and generating A_id_i as needed for the global update. The cost of the global update, other than by S_p , is negligible. The major costs associated with the solution of (2.15) are $Lp^3/3$, where L is the number of iterations for the minimization, and $2m(p+1)$ which accounts for the initialization of the products $(AD^{(k)})^T AD^{(k)}$ and $(r^{(k-1)})^T AD^{(k)}$. In the implementation, $L \leq 5$ is limited under the assumption, verified numerically, that exact global solution each outer iteration is unnecessary. Hence,

$$(4.22) \quad C_{S_p}(K) \approx K(2m(p+1) + \frac{L}{3}p^3),$$

and, using $L = 5$, the total parallel cost used in our theoretical estimates is

$$(4.23) \quad C_p(K) \approx C_l(K) + C_{S_p}(K)$$

$$(4.24) \quad \approx 2n_i^2(m + 6n_i) + 2K(5n_i^2 + 3n_im + (p+1)m + \frac{5}{6}p^3).$$

Not surprisingly, there must be a balance between the size of the local problems, in order to reduce local cost, and maintaining p small enough that the total number of outer iterations is not too large. This is illustrated in Figure 3, in which the maximal efficiency,

$$(4.25) \quad \text{Eff}(p, K) = \frac{C_s}{pC_p(K)},$$

is calculated for problems of increasing size and different numbers of domain blocks. Note that ideal efficiency $\text{Eff}(p) = 1$ is potentially exceeded for problems with size $n = \mathcal{O}(10^3)$, with $p \leq 32$, and $500 \preceq K \preceq 1000$.

A standard alternative to the design of a new algorithm for large scale problems is the parallelization of the most computationally demanding steps of a serial algorithm. Here we consider the use of PARPACK for the parallel implicit restart Lanczos method (IRLM), [23], to calculate the smallest eigenpair of matrix $[A, b]^T[A, b]$. We use the default number, 20, of Lanczos basis vectors and request only one eigenvalue needed for the explicit representation of the TLS solution. The parallelizable cost of the IRLM includes the initial cost of generating a 20-dimensional Krylov subspace, $20(n+1)(4m+9+2*20)$, and the per iteration cost of $20(n+1)(4m+(6+9)+4*20)$, [1]. The serial cost in each iteration for 19

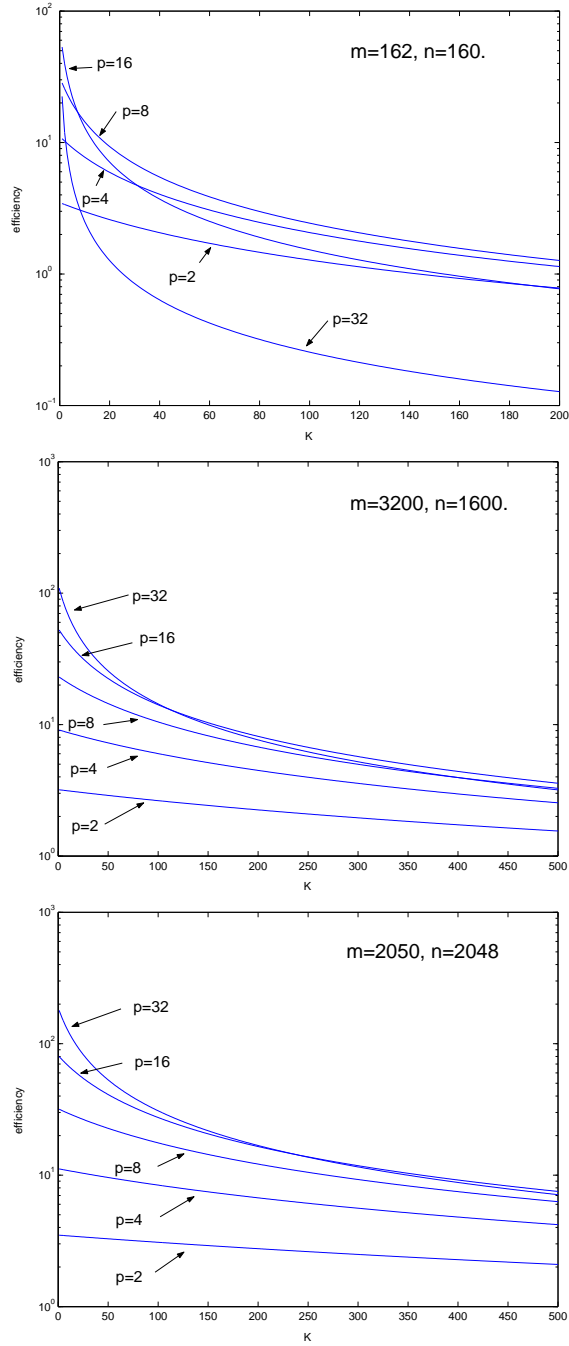


FIGURE 3. Theoretical efficiency of PVDTLS- S_p against number of iterations to convergence for different problem sizes, ignoring overheads of parallel communication and memory access.

implicitly shifted QR steps is $6 * 20^2 * 19$, [10]. Hence the total parallel cost for K iterations, again ignoring all parallel overheads, is

$$(4.26) \quad C_p^{IRLM}(K) = \frac{80(K+1)mn + (960 + 1900K)n}{p} + 45600K,$$

where, unlike PVDTLS, K is independent of the number of processors. This estimate is used to compare the efficiencies for the numerical experiments reported in Section 5.

5. NUMERICAL EXPERIMENTS

5.1. Evaluation of the PVDTLS Algorithms. To evaluate the algorithms we use both a standard test problem, **staar**, [24] and three different versions of a test problem which can be modified to change the underlying condition of the problem. In each case the problem size and number of subdomains are chosen so that each subdomain is of equal size. All tests are run with Matlab 7.0 and $x^{(0)} = 0$. We design the test data in such a way that the exact TLS solution, x_{TLS} , and optimal objective function value, $\phi_{\min} = \sigma_{n+1}^2$, are known. For stopping criterion we use tolerance $\tau = .00001$. In each case we report the number of outer iterations K to the given convergence, the relative errors of the converged $x^{(K)}$ to x_{TLS} and of the converged $\phi(x^{(K)})$ to ϕ_{\min} . Each case is tested for solution by Gauss-Seidel and PVD techniques PVDTLS- S_1 and PVDTLS- S_p . A selection strategy, denoted by ‘‘Sel’’ in the results indicating **Selection**, is also tested in which at each step the update chosen is that which gives the greatest reduction in the objective function when chosen from the local solutions Y_i , the BJ update $z^{(k)}$ and the convex update given by (2.13). This is consistent with the approach presented for the convex cases in [6], [19], and permits evaluation of whether the optimal update is necessary for obtaining a good solution. For the second test problem, the third situation leads to a case in which one block converges to a poor solution. In this case we evaluate also the use of truncated TLS for regularization of the local solution on this block to demonstrate the flexibility inherent in PVDTLS.

Test 1.

We consider problem **staar** [24] for $m = 162$. The Rayleigh quotient is $\phi(x_{\text{TLS}}) = m$, [14] (Chapter 2.4). For all cases the iteration converges to the correct solution and minimal ϕ in just two steps.

Test 2.

Based on the sensitivity analysis presented in [9], Björck *et al.* [2] suggest that the ratio $\sigma'_1/(\sigma'_n - \sigma_{n+1})$ be used as an approximate measure of the condition of a TLS problem. Here σ_i and σ'_i represent the i th singular values of matrices $[A, b]$ and A respectively. The approximate condition numbers partially reveal the local structure of the surface of $\phi(x)$ near the minimum point. A large condition number predicts that locally the surface is *flat* which makes the problem difficult to solve, while a small condition number indicates that the surface is locally *steep*. The TLS condition number for Test 1 is just 30.7, suggesting that **staar** is rather well-conditioned. In contrast, the following construction leads to examples with different condition numbers such as to evaluate PVDTLS with respect to the condition of the problem.

We construct $[A, b] \in R^{m \times (n+1)}$

$$\begin{aligned} [A, b] &= U\Sigma V^T, \\ U &= I_m - 2\chi\chi^T, \\ V &= I_{n+1} - 2\varsigma\varsigma^T, \\ \Sigma &= \begin{bmatrix} \Sigma_1 \\ 0 \end{bmatrix}, \end{aligned}$$

where

$$\begin{aligned} \chi(i) &= \sin(4\pi i/m), \quad i = 0, 1, \dots, m-1, \\ \varsigma(j) &= \cos(4\pi j/(n+1)), \quad j = 0, 1, \dots, n, \end{aligned}$$

and are normalized such that $\|\chi\|_2 = \|\varsigma\|_2 = 1$. In the tests we set $m = 162$ and $n = 160$, and Σ_1 is one of the following three diagonal matrices:

- a:** $\Sigma_1 = \text{diag}(v_1, v_2, v_3, v_4, 0.001)$, where v_1, v_2, v_3 and v_4 are constant row vectors with length $n/4$ and values $4/n, 2/n, 4/(3n)$ and $1/n$ respectively. This test has condition 4.76 and $\phi = \sigma_{n+1}^2 = 1.0e - 6$.
- b:** $\Sigma_1 = \text{diag}(1, 1/2, \dots, 1/n, 0.001)$. The condition is 190.5 and $\phi = \sigma_{n+1}^2 = 1.0e - 6$.
- c:** $\Sigma_1 = \text{diag}(1, 1/2, \dots, 1/(n+1))$. The condition is 2576 and $\phi = \sigma_{n+1}^2 = (1/161)^2$.

In these examples, the stationary points are all the same, the i th, $i \leq n$, of which is close to $[0, \dots, 0, h_i, 0, \dots, 0]$, where h_i is a positive number, see Figure 4. In each example, however, the function values at these stationary points are different.

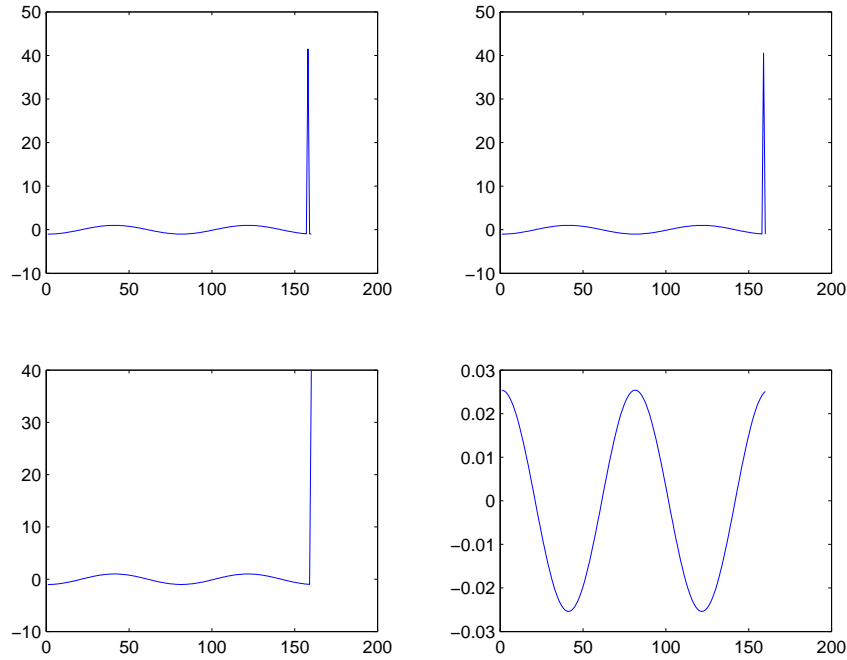


FIGURE 4. The four lowest stationary points of $\phi(x)$ for Test 2.
The lower right is the solution at the lowest stationary point, x_{TLS} .

In turn the matrix spectra are different and the condition increases from Test 2 (a) to (c). Results of the numerical experiments are detailed in Tables 1-3, in which $x.y(-t)$ indicates $x.y \times 10^{-t}$ and N indicates that convergence has not occurred by the 500th iteration.

Also illustrated in Figure 5 is the estimate of the q -factor in each case, which demonstrates the q -linear convergence behavior, [15]. Indeed, in most cases the convergence rate is nearly linear, which is consistent with the result for strongly convex problems, Theorem 2.3 of [4]. Theoretically, the converged point may be a saddle point, see Figure 1, but we did not encounter this in any of our tests. Instead, we did find some cases that converged to a point which is neither a saddle nor the true solution. Illustrated in Figure 6 is one such case, Test 2 (c), using S_1 for the global update, with $p = 2$ and overlap of 5. Here, it is apparent that the converged solution is close to the true solution, and even iterating to 50000 iterations, the solution is still not improved. This does not contradict Theorem 3.3 because the structure of objective function is so flat near to the true solution that (2.18) can not be satisfied to double float accuracy. In this case, it is possible to

TABLE 1. Test 2 (a): Outer iteration steps / relative error to ϕ_{\min}
/relative error to TLS solution.

PVD	p	overlap 0	overlap 1	overlap 2	overlap 5
S_1	2	10/ 6.5 (-6)/7.5(-4)	10/ 4.4 (-6)/6.6(-4)	10/ 2.5 (-6)/5.2(-4)	8/ 3.6 (-6)/6.3(-4)
	4	26/ 3.4 (-5)/3.8(-3)	25/ 2.0 (-5)/3.2(-3)	25/ 3.4 (-5)/4.2(-3)	24/ 2.5 (-5)/3.5(-3)
	8	54/ 9.8 (-5)/6.8(-3)	50/ 8.8 (-5)/6.4(-3)	49/ 7.9 (-5)/6.1(-3)	46/ 8.2 (-5)/6.2(-3)
S_p	2	8/ 7.3 (-7)/2.9(-4)	8/ 3.9 (-6)/8.8(-4)	8/ 5.7 (-7)/3.5(-4)	7/ 1.7 (-7)/1.9(-4)
	4	2/ 4.2 (-16)/3.7(-12)	2/ 4.2 (-16)/3.7(-12)	2/ 4.2 (-16)/3.7(-12)	2/ 0.0 (+0)/3.7(-12)
	8	2/ 0.0 (+0)/3.7(-12)	2/ 0.0 (+0)/3.7(-12)	2/ 4.2 (-16)/3.6(-12)	2/ 4.2 (-16)/3.7(-12)
Sel	2	7/ 7.3 (-6)/1.0(-3)	7/ 3.5 (-6)/7.3(-4)	7/ 1.6 (-6)/5.3(-4)	7/ 1.7 (-7)/2.6(-4)
	4	32/ 6.4 (-5)/5.8(-3)	31/ 2.8 (-5)/3.7(-3)	25/ 5.6 (-5)/5.3(-3)	22/ 3.3 (-5)/3.8(-3)
	8	45/ 1.7 (-4)/7.5(-3)	46/ 4.7 (-5)/4.8(-3)	41/ 6.3 (-5)/5.1(-3)	39/ 9.9 (-5)/6.4(-3)
GS S_1	2	6/ 9.1 (-8)/1.1(-4)	6/ 3.1 (-7)/2.0(-4)	6/ 5.2 (-8)/7.6(-5)	5/ 2.5 (-8)/5.8(-5)
	4	8/ 2.1 (-7)/3.1(-4)	8/ 5.8 (-7)/4.1(-4)	6/ 8.1 (-7)/5.8(-4)	7/ 1.8 (-8)/7.5(-5)
	8	12/ 1.2 (-6)/7.6(-4)	11/ 6.4 (-6)/1.4(-3)	11/ 3.3 (-6)/1.0(-3)	11/ 3.7 (-6)/1.1(-3)

TABLE 2. Test 2 (b): Outer iteration steps / relative error to ϕ_{\min}
/relative error to TLS solution.

PVD	p	overlap 0	overlap 1	overlap 2	overlap 5
S_1	2	N/ 7.5 (-3)/2.5(-2)	54/ 3.7 (-5)/1.8(-3)	24/ 9.0 (-6)/8.8(-4)	17/ 1.3 (-5)/1.0(-3)
	4	N/ 2.9 (-1)/3.0(-1)	80/ 5.6 (-5)/3.6(-3)	60/ 2.3 (-4)/8.6(-3)	37/ 1.3 (-3)/2.0(-2)
	8	N/ 8.2 (-1)/5.1(-1)	442/ 1.8 (-2)/8.1(-2)	338/ 2.2 (-3)/2.9(-2)	225/ 1.3 (-2)/6.7(-2)
S_p	2	364/ 1.4 (-3)/1.1(-2)	17/ 6.9 (-7)/2.6(-4)	13/ 5.1 (-7)/2.4(-4)	8/ 5.0 (-6)/1.4(-3)
	4	N/ 1.3 (-2)/7.2(-2)	57/ 2.5 (-4)/9.2(-3)	28/ 5.2 (-5)/4.6(-3)	24/ 2.4 (-5)/2.9(-3)
	8	79/ 3.6 (-4)/1.0(-2)	202/ 1.0 (-3)/1.9(-2)	219/ 1.4 (-3)/2.2(-2)	196/ 1.2 (-3)/2.0(-2)
Sel	2	10/ 1.6 (-3)/1.2(-2)	22/ 1.0 (-4)/2.4(-3)	16/ 4.7 (-5)/1.1(-3)	9/ 9.2 (-7)/4.5(-4)
	4	N/ 6.8 (-1)/5.3(-1)	195/ 1.1 (-3)/1.9(-2)	108/ 1.2 (-2)/5.7(-2)	150/ 3.1 (-4)/9.3(-3)
	8	N/ 1.3 (+0)/7.7(-1)	N/ 6.8 (-1)/5.4(-1)	465/ 2.2 (-1)/3.1(-1)	N/ 3.9 (-1)/3.9(-1)
GS S_1	2	33/ 1.2 (-3)/1.0(-2)	14/ 4.0 (-12)/6.4(-7)	8/ 8.8 (-6)/7.2(-4)	5/ 2.3 (-8)/3.3(-5)
	4	91/ 1.3 (-3)/1.8(-2)	32/ 2.8 (-4)/8.3(-3)	26/ 2.3 (-6)/7.0(-4)	12/ 7.1 (-6)/1.5(-3)
	8	119/ 2.5 (-3)/2.7(-2)	73/ 1.7 (-4)/6.5(-3)	67/ 6.8 (-5)/4.6(-3)	69/ 1.6 (-5)/1.9(-3)

obtain a solution which does converge with relative error less than 1.0×10^{-11} if we pick the initial guess as the true TLS solution with 10% noise added.

5.1.1. *Discussion.* Our main observations from the numerical results follow.

TABLE 3. Test 2 (c): Outer iteration steps / relative error to ϕ_{\min}
/relative error to TLS solution.

PVD	p	overlap 0	overlap 1	overlap 2	overlap 5
S_1	2	329/ 1.9(-3) /9.1(-2)	29/ 9.2(-5) /2.0(-2)	21/ 3.2(-5) /1.1(-2)	5/ 9.1(-4) /1.3(-1)
	4	454/ 7.7(-3) /5.3(-1)	78/ 8.4(-5) /7.6(-2)	35/ 4.5(-3) /4.2(-1)	41/ 4.3(-5) /4.6(-2)
	8	410/ 1.2(-2) /6.4(-1)	54/ 8.2(-3) /7.2(-1)	47/ 8.4(-3) /8.0(-1)	33/ 8.7(-3) /9.6(-1)
S_p	2	6/ 1.6(-3) /8.4(-2)	14/ 3.5(-9) /1.3(-4)	10/ 3.2(-6) /3.7(-3)	6/ 5.7(-6) /8.8(-3)
	4	16/ 6.3(-3) /6.7(-1)	38/ 3.2(-4) /1.4(-1)	28/ 2.8(-4) /1.2(-1)	18/ 6.8(-5) /6.1(-2)
	8	20/ 2.5(-3) /8.0(-1)	8/ 2.7(-3) /8.3(-1)	9/ 2.6(-3) /8.0(-1)	10/ 2.9(-3) /8.1(-1)
Sel	2	8/ 4.4(-5) /2.1(-2)	8/ 2.0(-5) /1.3(-2)	8/ 7.9(-6) /1.0(-2)	2/ 3.4(-2) /7.7(-1)
	4	3/ 1.5(-2) /1.1(+0)	10/ 1.1(-2) /8.6(-1)	48/ 2.9(-3) /3.7(-1)	157/ 1.3(-4) /6.9(-2)
	8	2/ 7.7(-3) /1.5(+0)	2/ 7.7(-3) /1.5(+0)	3/ 7.7(-3) /1.5(+0)	3/ 7.7(-3) /1.5(+0)
GS S_1	2	24/ 1.2(-3) /7.4(-2)	10/ 1.3(-5) /7.3(-3)	5/ 2.0(-6) /2.6(-3)	4/ 5.6(-8) /5.8(-4)
	4	8/ 2.2(-1) /2.2(+0)	14/ 2.6(-2) /1.5(+0)	10/ 4.0(-2) /1.9(+0)	14/ 1.7(-4) /9.2(-2)
	8	6/ 6.0(-2) /6.5(+0)	4/ 1.1(-1) /9.9(+1)	2/ 1.1(-1) /8.8(+1)	2/ 1.1(-1) /6.7(+1)

- (1) A simple selection strategy, “Sel”, does not provide satisfactory convergence behavior for PVDTLS. When the TLS condition number is not small, and the number of blocks increases, this approach does not succeed, even if regularization is applied to the “bad” block. In particular it is essential that the global update is optimized using an approach such as S_1 or S_p . This contrasts the LS case in which convex updates or BJ updates were often sufficient to maintain satisfactory convergence at less cost than the optimal recombination global update.
- (2) Use of a p -dimensional optimal update reduces the total number of iterations, and is more reliable at obtaining a good converged solution, than the line search S_1 , for minimal extra cost, when p is small.
- (3) When using a Gauss-Seidel approach, results show that it is essential to do some global update optimization, as indicated here by use of the line search S_1 . Moreover, as for the selection strategy, the Gauss Seidel approach is not successful when the TLS condition number increases and a given block needs some regularization. Use of S_p may be useful to improve the convergence of GS.
- (4) As p increases the number of iterations to convergence increases.

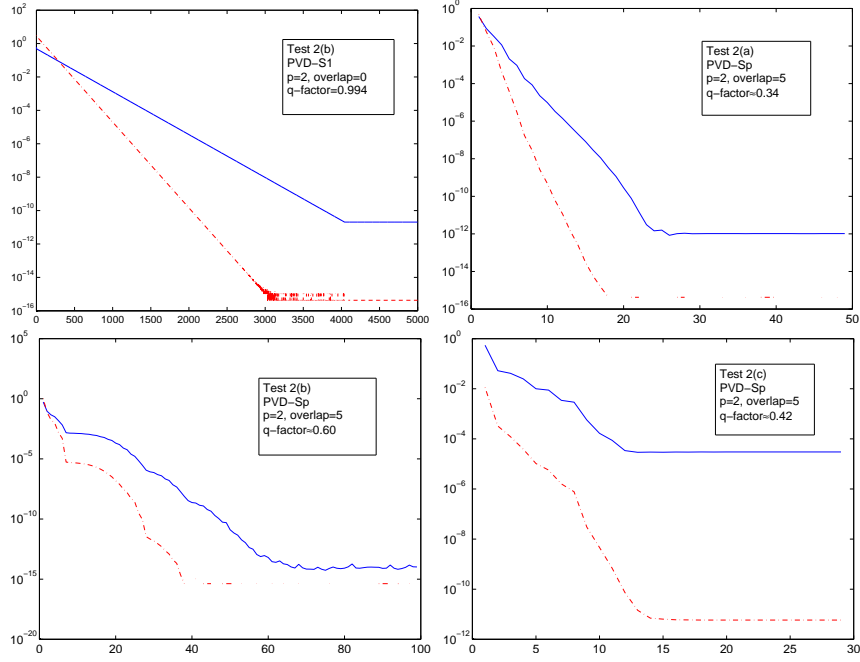


FIGURE 5. Convergence history for 4 examples. In each case the solid line indicates the relative error of $x^{(k)}$ to x_{TLS} (solid) and the dotted line the relative error of $\phi^{(k)}$ to ϕ_{\min} (dots). The approximate q -factors of linear convergence are also presented.

- (5) Increasing overlap can be used to reduce the number of iterations to convergence.

These latter two results are completely expected for any parallel approach. The former results are specific to the PVD for TLS.

Our implementation does not use the *forget-me-not* term in the global update as used in [4, 3]. While its inclusion might increase the speed of the convergence, it would also exclude the use of the efficient SVD update scheme. Moreover, the formulation as presented, also easily permits, independently of other domains, the inclusion of regularization for a specific subdomain, [13], [5].

5.2. Comparison with IRLM. To compare with parallel IRLM we also solve the second test case to achievable accuracy by both the parallel IRLM and PVD-TLS- S_p and evaluate the relative efficiency. We present the number of iterations to find in each case the *best achievable accuracy* measured in terms of the relative error

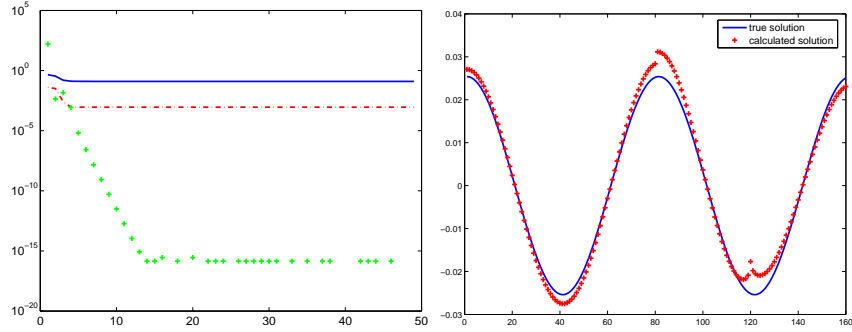


FIGURE 6. Test 2 (c), PVDTLS- S_1 , $p = 2$, overlap 5. Left: convergence history, the solid line indicates the relative error of $x^{(k)}$ to x_{TLS} , the dotted line the relative error of $\phi^{(k)}$ to ϕ_{min} and the crosses the estimated relative error of objective $|\phi(x^{(k)}) - \phi(x^{(k-1)})|/\phi(x^{(k)})$ Right: converged solution and true TLS solution.

to the true TLS solution. Representative cases for PVDTLS- S_p with overlap of 5, and 2, 4 and 8 subdomains are given in top half of Table 4. From these results we observe that PVDTLS has better achievable accuracy for the Test 2 (b), regardless of the number of subdomains. Moreover, it is apparent by the high number of iterations required to achieve a converged solution, that for this specific test the subdomain size is too small when the global problem is split into 8 local problems. The convergence history for parallel IRLM, shown in Figure 7, confirms that the IRLM does not improve with iteration after a certain number of iterations. On the other hand, in terms of accuracy parallel IRLM outperforms PVDTLS in two cases. Moreover the convergence history for PVDTLS in these cases has also stagnated, Figure 5 (a)-(d).

Table 5 shows the relative efficiency $C_p(\text{PVDTLS})/C_p(\text{IRLM})$ for achieving the equivalent accuracy with $p = 2$ and 4. A ratio greater than 1 suggests that IRLM is more efficient, while PVDTLS is potentially more efficient for a ratio less than 1. The results clearly demonstrate that parallel IRLM is less efficient than PVDTLS- S_p for the examples with higher condition number, while the converse is true for well-conditioned cases.

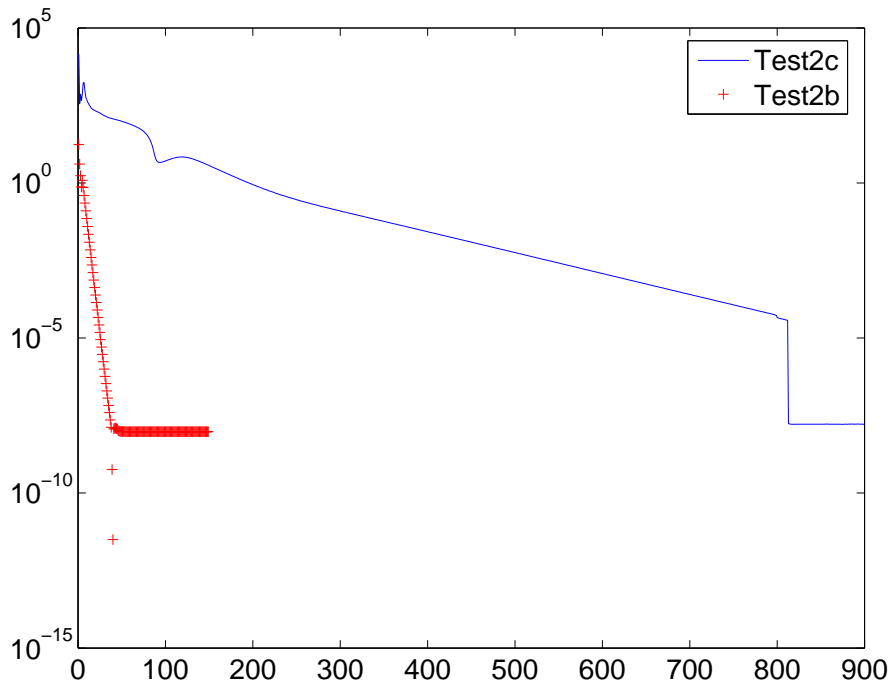


FIGURE 7. Convergence history for Test 2 (b) and Test 2 (c) with IRLM.

We also compare these two methods for larger examples, designed in the same way as Test 2 but with $m = 1602$ and $n = 800$. In this test the iteration is terminated for PVDTLS when either the relative solution error, $\|x^{(k)} - x_{\text{TLS}}\|/\|x_{\text{TLS}}\|$, is less than 1.0×10^{-6} , or if 1000 steps have been taken without achieving convergence. These results are tabulated in the lower half of Table 4. Observe that the IRLM iteration does not converge for the second two examples, even when we increase the dimension of Krylov subspace from 20 to 100. Therefore, it is clear that PVDTLS outperforms IRLM except for the well-conditioned test.

6. CONCLUSION

We have presented a domain decomposition strategy for solution of the TLS problem, using PVD techniques combined with optimal global update. The method can be implemented in a serial GS manner, or a parallel BJ approach. To obtain a satisfactory convergence history it is essential to optimize the global update at each outer iteration. The use of inexact local solutions is not needed because the local solutions use an optimal SVD update scheme. The algorithm can be adapted to handle problems in which some regularization is needed on a local subdomain

TABLE 4. Best achievable accuracy for PVDTLS- S_p with overlap 5 and for IRLM: iteration steps / relative error to TLS solution.

size	Method	Test 2 (a)	Test 2 (b)	Test 2 (c)
$m = 162$	IRLM	2/5.36(-15)	41/9.61(-12)	814/1.72(-8)
$n = 160$	PVD	$p = 2$ 25/1.03(-12)	60/9.12(-14)	12/8.71(-5)
	TLS	$p = 4$ 2/3.67(-12)	161/7.78(-14)	72/9.78(-5)
	S_p	$p = 8$ 2/3.67(-12)	4446/6.61(-13)	9450/5.48(-5)
$m = 1602$	IRLM	2/3.40(-14)	20000/3.90(+0)	20000/1.80(+3)
$n = 800$	PVD	$p = 2$ 21/5.10(-7)	41/3.80(-7)	1000/1.32(-6)
	TLS	$p = 4$ 1/1.55(-8)	300/9.77(-7)	1000/1.14(-5)
	S_p	$p = 8$ 1/1.55(-8)	1000/4.2(-2)	1000/7.28(-1)

TABLE 5. Number of iteration steps for PVDTLS / number of steps for IRLM/ **relative efficiencies**, $C_p(\text{PVDTLS})/C_p(\text{IRLM})$, ignoring parallel overheads in both cases, for reaching equivalent relative error to TLS solution, $\text{Error} = \|x^{(k)} - x_{\text{TLS}}\|/\|x_{\text{TLS}}\|$.

size	p	Test 2 (a) Error 3.67(-12)	Test 2 (b) Error 9.61(-12)	Test 2 (c) Error 9.78(-5)
$m = 162$	2	23 /2/ 3.71	51/41/ 0.341	12/683/ 0.014
$n = 160$	4	2/2/ 1.02	113/41/ 0.342	72/683/ 0.015

because of ill-conditioning of the local problem. As compared to a parallel implementation of the implicitly restarted Lanczos algorithm, PVDTLS- S_p offers a viable strategy for those large problems which are also poorly conditioned.

REFERENCES

1. Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the solution of algebraic eigenvalue problems: A practical guide*, SIAM, Philadelphia, 2000.
2. A. Björck, P. Heggernes, and P. Matstoms, *Methods for large scale total least squares problems*, SIAM Journal on Matrix Analysis and Applications **22** (2000), no. 2, 413–429.
3. J. E. Dennis and T. Steihaug, *A Ferris-Mangasarian technique applied to linear least squares problems*, Tech. Report CRPC-TR 98740, Rice University, 1998.
4. M. C. Ferris and O. L. Mangasarian, *Parallel variable distribution*, SIAM Journal on Optimization **4** (1994), no. 4, 815–832.

5. R. D. Fierro, G. H. Golub, P. C. Hansen, and D. P. O'Leary, *Regularization by truncated total least squares*, SIAM Journal on Scientific Computing **18** (1997), 1223–1241.
6. A. Frommer and R. Renaut, *A unified approach to parallel space decomposition methods*, Journal of Computational and Applied Mathematics (1999), 205–223.
7. G. Golub, *Some modified matrix eigenvalue problems*, SIAM Review **15** (1973), 318–334.
8. G. H. Golub, P. C. Hansen, and D. P. O'Leary, *Tikhonov regularization and total least squares*, Numerical Linear Algebra with Applications **21** (1999), no. 1, 185–194.
9. G. H. Golub and C. Van Loan, *An analysis of the total least squares problem*, SIAM Journal on Numerical Analysis **17** (1980), 883–893.
10. ———, *Matrix computations*, third ed., John Hopkins Press, Baltimore, 1996.
11. M. Gu and S. C. Eisenstat, *A stable and fast algorithm for updating the singular value decomposition*, Tech. Report YALEU/DCS/RR-966, Yale University, 1994.
12. H. Guo, IRR: *An algorithm for computing the smallest singular value of large-scale matrices*, International Journal of Computer Mathematics **77** (2001), no. 2, 89–104.
13. H. Guo and R. A. Renaut, *A regularized total least squares algorithm*, in *Sabine Van Huffel and Philippe Lemmerling, total least squares and errors-in-variables modeling: Analysis, algorithms and applications*, pp. 57–66, Kluwer Academic Publishers, 2002.
14. S. Van Huffel and J. Vandewalle, *The total least squares problem: computational aspects and analysis*, SIAM, Philadelphia, 1991.
15. C. T. Kelley, *Iterative methods for optimization*, Frontiers in Applied Mathematics, SIAM, 1999.
16. N. Mastronardi, P. Lemmerling, and S. Van Huffel, *Fast structured total least squares algorithm for solving the basic deconvolution problem*, SIAM Journal on Matrix Analysis and Applications **22** (2000), no. 2, 533–553.
17. B. N. Parlett, *The symmetric eigenvalue problem*, Prentice-Hall, 1980.
18. B. Philippe and M. Sadkane, *Computation of the singular subspace associated with the smallest singular values of large matrices*, Tech. Report 754, IRISA, 1993.
19. R. A. Renaut, *A parallel multisplitting solution of the least squares problem*, Numerical Linear Algebra with Applications **5** (1998), 11–31.
20. R. A. Renaut and H. Guo, *Efficient solution of regularized total least squares*, SIAM Journal on Matrix Analysis and Applications , to appear.
21. A. H. Sameh and J. A. Wisniewski, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM Journal on Numerical Analysis **19** (1982), no. 6, 1243–1259.
22. G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM Review **42** (1996), no. 2, 267–293.
23. D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl. **13** (1992), 357–385.
24. J. Staar, *Concepts for reliable modelling of linear system with application to on-line identification of multivariable state space descriptions*, Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium, June 1982.

DEPARTMENT OF MATHEMATICS AND STATISTICS, ARIZONA STATE UNIVERSITY, TEMPE, AZ
85287 1804