Int J Geomath (2015) 6:251–294
DOI 10.1007/s13137-015-0073-9

ORIGINAL PAPER

Compression approaches for the regularized solutions of linear systems from large-scale inverse problems

Sergey Voronin[1] · Dylan Mikesell[2] · Guust Nolet[3]

Received: 16 January 2015 / Accepted: 6 May 2015 / Published online: 19 May 2015

Abstract We introduce and compare new compression approaches to obtain regularized solutions of large linear systems which are commonly encountered in large scale inverse problems. We first describe how to approximate matrix vector operations with a large matrix through a sparser matrix with fewer nonzero elements, by borrowing from ideas used in wavelet image compression. Next, we describe and compare approaches based on the use of the low rank singular value decomposition (SVD), which can result in further size reductions. We describe how to obtain the approximate low rank SVD of the original matrix using the sparser wavelet compressed matrix. Some analytical results concerning the various methods are presented and the results of the proposed techniques are illustrated using both synthetic data and a very large linear system from a seismic tomography application, where we obtain significant compression gains with our methods, while still resolving the main features of the solutions.

Keywords Ill-posedness · Regularization · Singular value decomposition · Wavelets · Data compression

Mathematics Subject Classification 65F22 · 86-08 · 65T60 · 15A18

✉ Sergey Voronin
sergey.voronin@colorado.edu

[1] Department of Applied Mathematics, University of Colorado, Boulder, CO 80309, USA

[2] Department of Earth, Atmospheric and Planetary Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

[3] Géoazur, Université de Nice, 06560 Sophia Antipolis, France

1 Introduction

This paper describes practical approaches to obtain approximate but accurate regularized solutions to large linear systems arising from large scale inverse problems, without the need to load into memory the often very large original matrix used in the corresponding optimization problems. Typically, such as in the case of the seismic tomography application which we mention here for illustration (Simons et al. 2011) (involving the reconstruction of seismic wave velocities in the Earth's interior with respect to a given spherically symmetric model), the physics calls for a solution of a linear system $Ax = \bar{b}$ with matrix $A \in \mathbb{R}^{m \times n}$ (often with $m \neq n$). In practice, instead of the true right hand side $\bar{b}$, we are given the noisy right hand side $b = \bar{b} + \nu$, with ν being an unknown noise vector. The matrix A can be very large and is likely to be ill-conditioned and exhibit fast nonlinear decay of singular values (Nolet 2008). In order to obtain a solution given matrix A and right hand side b, one often uses a derivative of Tikhonov regularization involving a regularization parameter $\lambda > 0$ (Tikhonov 1963). In its classical form, this is simply the minimization problem:

$$\bar{x} = \arg\min_{x} \left\{ ||Ax - b||_2^2 + \lambda ||x||_2^2 \right\}, \tag{1.1}$$

which replaces the constrained system $Ax = b$ by the ℓ_2 minimization of the model residual norm $\|Ax - b\|_2$, with a constraint on the ℓ_2 norm of the model, controlled by the parameter λ. For large λ, $\bar{x}$ tends to be close to zero. Regularization is necessary to counter the effects of ill-conditioning: the presence of small singular values in the matrix, which if left unaccounted for, blows up the norm of the solution and makes it very sensitive to data errors (Calvetti et al. 2000). The latter part of this property is worth repeating as it is central to the ideas in this paper: small errors in the operator A and the right hand side b do not induce big changes in the regularized solution. The regularization in (1.1) is referred to as ℓ_2 regularization, because it involves the minimization of the ℓ_2 model norm. Other types of regularization are possible: for example, sparsity constrained regularization is also frequently used, including in geophysical applications (Chárlety et al. 2013). In this paper, we discuss the application of our methods to ℓ_2 regularization, as it is the most commonly used regularization. However, the techniques apply also to other types of regularization and optimization techniques. The quadratic functional in (1.1) can be differentiated to yield the linear system for the regularized solution:

$$(A^T A + \lambda I)\bar{x} = A^T b. \tag{1.2}$$

If the matrix A is not too large, then there is no problem in solving this linear system with an iterative algorithm. A conjugate gradient or the LSQR algorithm (Paige and Saunders 1982) can be efficiently used for this purpose. Typically, we may wish to incorporate additional terms into the regularization, such as Laplacian smoothing (Nolet 2008). In that case we solve instead:

$$\bar{x} = \arg\min_{x} \left\{ ||Ax - b||_2^2 + \lambda_1 ||x||_2^2 + \lambda_2 ||Lx||_2^2 \right\}, \tag{1.3}$$

which can be solved through the linear system:

$$(A^T A + \lambda_1 I + \lambda_2 L^T L)\bar{x} = A^T b, \tag{1.4}$$

or through the augmented least squares problem and its corresponding normal equations:

$$\bar{x} = \arg\min_x \left\| \begin{bmatrix} A \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix} x - \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \right\|_2^2 \Longrightarrow \begin{bmatrix} A \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix}^T \begin{bmatrix} A \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix} \bar{x} = \begin{bmatrix} A \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix}^T \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}.$$

As long as A and L can be applied to vectors, the solution can be obtained by a number of iterative algorithms. The problem occurs when A is too large to load into memory. In the seismic tomography application we refer to (Debayle and Sambridge 2004; Simons et al. 2011; van Heijst and Woodhouse 1999), the matrix is several terabytes in size, so it may not be possible to load into memory in full, even on relatively large memory computer clusters. Thus, we must find ways to condense the matrix size using acceptable approximations which do not significantly alter the final regularized solutions.

Many attempts at approximating matrices have been documented (Markovsky 2012; Wang and Zhang 2013). However, few attempts have been made to apply the approximations to regularization. One of the main papers which precedes ours is Lampe et al. (2012), where Krylov subspace approximations for Tikhonov regularization are discussed. In this paper, we discuss two different techniques: wavelet based approximations and low rank singular value decomposition (SVD). Our SVD techniques are especially effective when the matrix exhibits fast nonlinear decay of singular values. From our experiments, Krylov subspace dimensionality reduction techniques, while interesting and promising, tend to do worse when the decay of singular values of the matrix is fast. This is in contrast to the techniques we describe, which in such cases, do not significantly degrade the solution quality and lower the hardware requirements to obtain a solution. Even if A is small enough that it can be loaded into memory, there may still be interest in the techniques we describe for gains of speed or to be able to solve several problems at once on one machine.

2 Organization of the paper

We now briefly describe the organization of this paper. We assume that the reader is interested in obtaining regularized solutions to a system $Ax = b$, where $A \in \mathbb{R}^{m \times n}$ is as previously described: very large (perhaps more than a TB), with rapidly decaying singular values, and stored on the disk. In Sect. 3, we describe notation and preliminary concepts including the various norms we use, the SVD, and a few lemmas that we use for our later derivations. In Sect. 4, we describe how to do approximate matrix–vector operations with the matrix A, using a smaller matrix M derived from A, via a wavelet thresholding based algorithm. The matrix M is obtained from A entirely on the disk. The big A matrix is never required to be loaded into RAM. We assume that on output of this procedure, the matrix M, which is still large, but significantly smaller than A (in memory size), can be loaded into RAM at least for a limited number of operations. After M is obtained, two options are available to the user: the regularization can be performed directly via M, or greater compression may be sought. In many cases, we assume that the latter will be true: the user would like to obtain a matrix small enough to use on their local machine. In Sect. 5, we describe how to compute and use the low rank SVD, which is known to provide an optimal (in terms of error in the Frobenius and spectral norms) rank k approximation of the matrix. We mention how to compute such an approximation with a randomized algorithm, which uses a limited number of matrix vector operations with M (or with A, if that is feasible). We introduce several different strategies which can be used. We show that several strategies are mathematically equivalent, but one may be preferred over others depending on the setup of the problem. Both in Sects. 4 and 5, we mention block matrix techniques, which are very useful for very large problems, where operating with the full matrices A or M is not possible. The outlined strategies make feasible to compute approximate regularized solutions to the original $Ax = b$ system, using matrices many times smaller than A, either with fewer nonzeros, in the case of the wavelet compressed M, or with much smaller dimensions, in the case of the low rank SVD. For some approaches, the matrices may be small enough to load on modern laptop computers, even if the original A was more than a TB in size. In Sect. 6, we present numerical experiments to illustrate the techniques for the compression approaches outlined in Sects. 4 and 5. We present results for both synthetic data, exhibiting different rates of decay of singular values and different wavelet compressibility characteristics, and for real data from a large scale seismic tomography application.

3 Notation and preliminaries

We refer to $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$, as respectively, a real valued vector of n elements and a real valued matrix of m rows and n columns. Most of the techniques we describe apply to complex valued matrices also. For vectors, we define the vector norm as the usual Euclidean norm:

$$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}},$$

and we use the notation $\|x\|$ to mean $\|x\|_2$. For matrices, we define the spectral norm as:

$$\|A\|_2 = \sigma_{\max}(A)$$

where $\sigma_{\max}(A)$ denotes the largest singular value of matrix A. The Frobenius norm is defined as:

$$\|A\|_F = \left(\sum_{i,j} A_{i,j}^2 \right)^{\frac{1}{2}}.$$

By A^{-1} we denote the inverse matrix, which is applicable only for square dimensions (i.e. $m = n$). The following result, which can be directly verified by means of block matrix inversion, is known as the Woodbury (1950) inverse formula and will be useful in our analysis in Sect. 5:

Lemma 3.1 *Take $D \in \mathbb{R}^{n \times n}$, $P \in \mathbb{R}^{n \times k}$, $T \in \mathbb{R}^{k \times k}$, and $R \in \mathbb{R}^{k \times n}$. Assume that D and T are invertible. Then $D + PTR$ is invertible if and only if $T^{-1} + RD^{-1}P$ is, and the following identity holds:*

$$(D + PTR)^{-1} = D^{-1} - D^{-1}P\left(T^{-1} + RD^{-1}P\right)^{-1} RD^{-1}. \tag{3.1}$$

Every matrix A admits a *SVD* (Trefethen and Bau 1997) of the form

$$\underset{m \times n}{A} = \underset{m \times p}{U} \; \underset{p \times p}{\Sigma} \; \underset{p \times n}{V^T}, \tag{3.2}$$

where $p = \min(m, n)$ and U and V are orthonormal matrices and Σ is a diagonal matrix. The columns $(u_j)_{j=1}^p$ and $(v_j)_{j=1}^p$ of U and V are called the left and right singular vectors of A, respectively, and the diagonal entries $(\sigma_j)_{j=1}^p$ of Σ are the singular values of A. The singular values of A are ordered so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$. U and V have orthonormal columns ($U^T U = V^T V = I_p$).

$$U = \left[u_1\, u_2\, \ldots\, u_p\right], \quad V = \left[v_1\, v_2\, \ldots\, v_p\right], \quad \text{and} \quad \Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots \\ 0 & \sigma_2 & 0 & \cdots \\ 0 & 0 & \sigma_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

so that

$$A = \sum_{j=1}^{p} \sigma_j\, u_j\, v_j^T.$$

In finite precision, the numerical rank of the matrix will be r and it is possible (in fact, likely for a large matrix) that $r < p$. That is, σ_j appears as 0 to the machine for $j \geq r$. Thus, in such scenario we write:

$$A = \sum_{j=1}^{r} \sigma_j\, u_j\, v_j^T.$$

where the precise value of r is typically unknown. It is always the case that $r \leq p$.

For a matrix which is not well conditioned and has fast decay of singular values, many nonzero singular values σ_j for $j < r$ will be very small relative to the largest singular value σ_1 and the drop off in value starting from σ_1 will be rapid and nonlinear. In these cases, the low rank SVD approximation A_k provides a good approximation to the matrix for relatively small k relative to p. We define A_k by taking into account only the first $k < p$ singular values and vectors: that is, with $U_k \in \mathbb{R}^{m \times k}$ consisting of the first k columns of U, $\Sigma_k = \text{Diag}(\sigma_1, \ldots, \sigma_k) \in \mathbb{R}^{k \times k}$ consisting of k rows and columns of Σ, and $V_k \in \mathbb{R}^{n \times k}$ consisting of the first k columns of V:

$$A_k = \sum_{j=1}^{k} \sigma_j\, u_j\, v_j^T = U_k\, \Sigma_k\, V_k^T, \tag{3.3}$$

$$U_k = \big[u_1\, u_2\, \ldots\, u_k\big], \quad V_k = \big[v_1\, v_2\, \ldots\, v_k\big], \quad \text{and} \quad \Sigma_k = \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & & 0 \\ 0 & 0 & 0 & \cdots & \sigma_k \end{bmatrix}.$$

By the Eckart–Young theorem (Trefethen and Bau 1997, Theorem 5.8), it is known that A_k is the optimal rank k approximation to A in both the spectral and Frobenius norms and that:

$$\|A - A_k\|_2 = \sigma_{k+1},$$

when the error is measured in the ℓ^2 operator norm, and

$$||A - A_k||_F = \left(\sum_{j=k+1}^{p} \sigma_j^2 \right)^{1/2}$$

in the Frobenius norm. When $k \ll p$, the matrices U_k, Σ_k, and V_k are significantly smaller than the corresponding full SVD matrices U, Σ, and V. The choice of k is up to the user, but greater k requires greater computation time and storage requirements. Notice that A and A_k are related via the expansion:

$$A = \sum_{i=1}^{k} \sigma_i u_i v_i^T + \sum_{i=k+1}^{r} \sigma_i u_i v_i^T$$

where the first sum on the right corresponds to A_k and the second sum corresponds to $\hat{A}_k$, consisting of the remaining singular vectors (in matrices $\hat{U}_k$, $\hat{V}_k$) which are not used in the truncated SVD expansion. These remaining singular vectors are orthogonal to the vectors in matrices U_k and V_k which go into the construction of A_k. We have the following relations for $k < r$:

$$\begin{aligned} U &= [U_k, \hat{U}_k]; \quad V = [V_k, \hat{V}_k]; \\ A &= \sum_{i=1}^{k} \sigma_i u_i v_i^T + \sum_{i=k+1}^{r} \sigma_i u_i v_i^T = U_k \Sigma_k V_k^T + \hat{U}_k \hat{\Sigma}_k \hat{V}_k^T \\ &= A_k + \hat{U}_k \hat{\Sigma}_k \hat{V}_k^T = A_k + \hat{A}_k, \\ A^T &= \sum_{i=1}^{k} \sigma_i v_i u_i^T + \sum_{i=k+1}^{r} \sigma_i v_i u_i^T = V_k \Sigma_k U_k^T + \hat{V}_k \hat{\Sigma}_k \hat{U}_k^T \\ &= A_k^T + \hat{V}_k \hat{\Sigma}_k \hat{U}_k^T = A_k^T + \hat{A}_k^{\,T}, \end{aligned}$$

$$A^T A = \sum_{i=1}^{k} \sigma_i^2 v_i v_i^T + \sum_{i=k+1}^{r} \sigma_i^2 v_i v_i^T = V_k \Sigma_k^2 V_k^T + \hat{V}_k \hat{\Sigma}_k^2 \hat{V}_k^T$$
$$= A_k^T A_k + \hat{V}_k \hat{\Sigma}_k^2 \hat{V}_k^T = A_k^T A_k + \hat{A_k}^T \hat{A}_k,$$

where

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T = U_k \Sigma_k V_k^T \quad \text{and} \quad A_k^T A_k = \sum_{i=1}^{k} \sigma_i^2 v_i v_i^T = V_k \Sigma_k^2 V_k^T,$$

and $U_k^T \hat{U}_k = V_k^T \hat{V}_k = 0$ and $U_k^T U_k = \hat{U}_k^T \hat{U}_k = V_k^T V_k = \hat{V}_k^T \hat{V}_k = I$. Additionally, we have the following properties which we will exploit in Sect. 5:

Lemma 3.2 *For vectors $v \in \mathbb{R}^k$ and $w \in \mathbb{R}^m$, $||U_k v||_2 = ||v||_2$ and $||U_k^T w||_2 \leq ||w||_2$. The same also holds for vectors $\bar{v} \in \mathbb{R}^k$ and $\bar{w} \in \mathbb{R}^n$ and matrices V_k and V_k^T.*

Proof Note that

$$UU^T = I = [U_k, \hat{U}_k] \begin{bmatrix} U_k^T \\ \hat{U}_k^T \end{bmatrix} = U_k U_k^T + \hat{U}_k \hat{U}_k^T \implies U_k U_k^T = I - \hat{U}_k \hat{U}_k^T.$$

Thus:

$$||U_k v||_2^2 = \langle U_k v, U_k v \rangle = \langle v, U_k^T U_k v \rangle = \langle v, v \rangle = ||v||_2^2,$$
$$||U_k^T w||_2^2 = \langle U_k^T w, U_k^T w \rangle = \langle w, U_k U_k^T w \rangle = \langle w, (I - \hat{U}_k \hat{U}_k^T) w \rangle$$
$$= \langle w, w \rangle - \langle w, \hat{U}_k \hat{U}_k^T w \rangle \leq ||w||_2^2.$$

The computations with V_k and V_k^T take similar form. □

4 Approximate matrix–vector operations with wavelet compression

Most iterative algorithms applicable to our discussion can be successfully implemented if we can perform the two key operations with the matrix A:

$$Ax \quad \text{and} \quad A^T y, \tag{4.1}$$

where $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$. We now discuss a technique to perform these operations approximately, using a smaller matrix derived from A by means of wavelet compression (Daubechies 1988; Härdle et al. 1998). Wavelets provide a multi-resolution approach to signal analysis, capturing the fine and coarse scale parts of a signal, and wavelet transforms can be performed efficiently (Akansu and Haddad 1992; Sweldens 1995). In our application, the matrix rows have features which are well represented by wavelets. To motivate this approach, consider wavelet compression applied to a geophysical model (or any typical vectorized image). We compare the original model x (in row vector form) to the inverse transform of the thresholded wavelet transformed model based on the relation:

$$x \approx (W^{-1}(\mathbb{T}(Wx^T)))^T, \tag{4.2}$$

where W and W^{-1} represent the forward and inverse wavelet transforms (Meyer 1993) and the thresholding operation $\mathbb{T}(\cdot)$ retains a certain percentage of the largest coefficients (by absolute value) of its input vector. The transpose operations assure that we are applying the transforms to column vectors, in view of their representation as matrices W and W^{-1}. Relation (4.2) holds when the row vector x is wavelet compressible. This is not necessarily the case for arbitrary x, yet does hold in many situations. For example, in the case of the application we allude to in this paper, the vectors are geophysical kernels representing a sensitivity of the observable (usually a phase or a delay) with respect to the intrinsic velocity as a function of space (Marquering et al. 1998). These kernels arise from integral equations and are generally smooth, and have been observed by us to be compressible by imposing a threshold on the wavelet coefficients. Many different kinds of thresholding functions exist. For our purposes, we simply use the hard thresholding function:

$$H_\alpha(x) = \begin{cases} x & \text{if } |x| > \alpha, \\ 0 & \text{if } |x| \le \alpha. \end{cases} \tag{4.3}$$

With the right choice of wavelet transform, only a small fraction of the coefficients in the wavelet transformed representation Wx need to be retained for a good reconstruction. That is, the threshold α can be taken to be quite large relative to the magnitudes of the elements of the vector Wx. In Fig. 1, below, a smooth CDF 9–7 transform was used (Cohen et al. 1992). We compare the original row vectorized image x to the reconstructed image $(W^{-1}\mathbb{T}(Wx^T))^T$ using a $2D$ CDF 9–7 transform over the image. We observe that as the amount of retained nonzero wavelet coefficients decreases, the reconstruction quality worsens, but the main features of the image are still retained. In the rightmost plot of Fig. 1, we define $E = 100\frac{\|x-(W^{-1}\mathbb{T}(Wx^T))^T\|}{\|x\|}$ as the percent error and $N = 100\frac{nnz(\mathbb{T}(Wx^T))}{nnz(Wx^T)}$ as the percent coefficients retained. Clearly, the reconstruction error can be controlled by keeping a certain (typically small) number of nonzero coefficients. Notice also that at about 7 % coefficients retained, we have a substantial 30 % error E. Yet, the image looks quite recognizable to the eye, with a bit of smoothing compared to the original.

Assuming the rows of our matrix A are wavelet compressible [that is for some relatively small threshold, satisfy approximately the relation (4.2)], we would like to apply the same principle to approximate matrix vector operations (4.1) with the big original matrix A through a smaller matrix M so that only the smaller matrix M needs to be loaded into memory. The matrix M will have the same dimensions as A but fewer nonzeros, so it takes less space on disk and in memory. One forms this matrix by transforming and thresholding the individual rows of A, an operation which can be done entirely on the disk, without loading any parts of A into RAM. The transform W used for each row can vary from application to application, depending on the structure

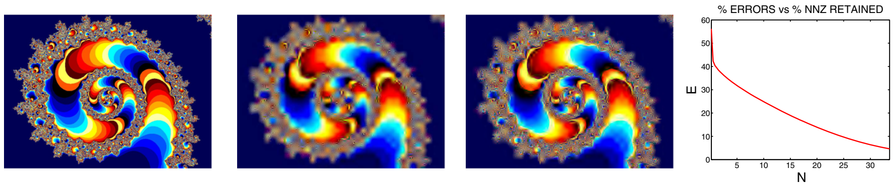

Fig. 1 A fractal image x (*left*) and reconstructions $(W^{-1}(\mathbb{T}(Wx^T)))^T$ with 1.4 and 6.8 % of retained wavelet coefficients. Plot of percent error norm vs percent nonzeros retained

of the rows of A. In our seismic tomography application for which we give examples in Sect. 6, we simply used the $1D$ CDF 9–7 transform for each row disregarding their inherent multi-dimensional structure. We believe that even better results can be obtained by tailoring W to the structure of the matrix data.

Each row of M is obtained by applying the wavelet transform and thresholding to the corresponding row of A:

$$A = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} \rightarrow M = \begin{bmatrix} \mathbb{T}(Wr_1^T)^T \\ \mathbb{T}(Wr_2^T)^T \\ \vdots \\ \mathbb{T}(Wr_m^T)^T \end{bmatrix} = \mathbb{T}(AW^T) \approx AW^T.$$

We can then approximate the operations (4.1). Using the relations:

$$Mx \approx AW^T x \quad \text{and} \quad M^T y \approx (AW^T)^T y = WA^T y,$$

we obtain the approximation formulas:

$$Ax \approx MW^{-T} x \quad \text{and} \quad A^T y \approx W^{-1} M^T y. \tag{4.4}$$

This means that the operations (4.1) can be performed approximately via (4.4), using the smaller matrix M and the inverse and inverse-transpose wavelet transforms. In practice, only M needs to be loaded in memory as the wavelet transforms would be implemented as routines. The inverse-transpose transform is equivalent to the forward transform when W is orthogonal and $W^{-1} = W^T$. For the non-orthogonal case, such as for example the CDF 9–7 transform, the inverse-transpose transform can be approximated by applying the forward transform with the inverse filters. The success of this approximation method depends on the size ratio between M and A and the percent error in the approximate operations. This depends on the data, the transform that is used, and the threshold used in the thresholding function. Typically, we identify the threshold α in (4.3) as follows. The input is sorted by putting the entries with largest absolute magnitude in front. Then a threshold is identified by putting the marker at some point of the nonzero entries (for example at the largest 15 % mark of the total nonzeros). Then all the entries with absolute magnitude less than the identified threshold are zeroed out. The percent error in the approximate operations then depends on the percent error in the reconstruction of each row. That is, if for an arbitrary row

r, $(W^{-1}\mathbb{T}(Wr^T))^T$ is not close to r, then the approximate operations using M formed with this threshold will probably not be accurate. A less aggressive threshold then needs to be used. Later we give examples for synthetic data and our seismic tomography application. For our application, we have observed that one can expect M to be at least 3 times smaller in memory requirements than A without incurring significant errors in the operations Ax, $A^T y$, and $A^T Ax$.

If A is very large, the matrix M may still be too big to load directly into memory. In that case, we may consider splitting the matrix in parts along its rows, with the matrix vector operations applied blockwise:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{bmatrix} \implies Ax = \begin{bmatrix} A_1 x \\ A_2 x \\ \vdots \\ A_p x \end{bmatrix} \quad \text{and} \quad A^T y = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{bmatrix}^T \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} = \sum_{j=1}^{p} A_j^T y_j.$$

Next, we can apply the wavelet compressed technique to the block matrices. We can proceed to form the matrices $M_1 = \mathbb{T}(A_1 W_1^T), \ldots, M_p = \mathbb{T}(A_p W_p^T)$, which are smaller wavelet thresholded versions of the original blocks $A_1, \ldots, A_p$. We can then perform approximate operations using these new sparser blocks:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_p \end{bmatrix} \to M = \begin{bmatrix} \mathbb{T}(A_1 W_1^T) \\ \mathbb{T}(A_2 W_2^T) \\ \vdots \\ \mathbb{T}(A_p W_p^T) \end{bmatrix} \implies Ax \approx \begin{bmatrix} M_1 W_1^{-T} x \\ M_2 W_2^{-T} x \\ \vdots \\ M_p W_p^{-T} x \end{bmatrix}$$

$$\text{and} \quad A^T y \approx \sum_{j=1}^{p} W_j^{-1} M_j^T y_j. \tag{4.5}$$

In the above formulas, we have used different transform matrices $W_1, \ldots, W_p$ for the different blocks. This may provide an advantage when the data in the matrix can be grouped. For example, some groups may have mostly smooth and others may have mostly sharp features. In such a case, it may be advantageous to use different transforms (ex, smooth CDF wavelet or sharper Haar wavelet) on the different blocks. If this is not the case, the same transform can be used for each block so that $W_1 = \cdots = W_p = W$.

Let us now discuss the application of these ideas to (1.2). Plugging in the approximated matrix–vector operations we obtain:

$$(W^{-1} M^T M W^{-T} + \lambda I)\tilde{x}_w = W^{-1} M^T b.$$

where $\tilde{x}_w$ will be the approximation to $\bar{x}$ in (1.2). If A is so large that after forming M we still cannot load M into memory, then M would be split into blocks $M_1, \ldots, M_p$. No matter how large A is, we can always choose p large enough so that the individual blocks M_j are manageable in size and can be loaded into RAM. In that case, we can still do operations in blocked form via (4.5) by loading as many parts of M as we can into memory, performing part of the operation and then replacing the in-memory blocks with the remaining blocks of M to perform the rest. As long as fast disks (such as SSDs) are available, this is viable in practice, but may be very slow if many operations are needed. In the case that M is too large to be loaded in full, the techniques discussed in the following section can be used to obtain further size reductions.

5 Low rank SVD approximation

The wavelet approximation techniques for matrix–vector operations discussed in the previous section enable us to approximate the operations (4.1) through a matrix several times smaller than A. However, in practice, the matrix M can still be quite big if A is particularly large. It is plausible that we can do some operations with A through M but only for a relatively short amount of time [perhaps through the blocked form (4.5)]. Assuming that we can indeed do a limited number of matrix vector multiplications with A through M, we now discuss other techniques for compression based on the low rank SVD. Once such a decomposition is obtained through a limited amount of matrix vector multiplications with A (approximated through M), we can obtain approximate forms of regularization algorithms which require the use of significantly smaller matrices.

5.1 Computation with randomized algorithm

We now discuss how the low rank SVD of rank k may be computed. One direct way is to compute it from the full SVD of the matrix. Given the full SVD $A = U\Sigma V^T$ one can take the first k columns of U and V to be the matrices U_k and V_k and the first k diagonal elements of Σ to form Σ_k. For large matrices, this is not practical since the computation of the full SVD is prohibitively expensive [the cost for an $m \times n$ matrix is on the order of $\mathcal{O}(mn \min(m, n))$ operations (Trefethen and Bau 1997)]. The algorithm which we use is an adaptation of the method proposed in Halko et al. (2011). The cost of the proposed randomized algorithm for the rank k SVD approximation is substantially lower [the cost is $\mathcal{O}(mnk)$ operations].

The randomized algorithm finding a rank k approximation of $A \in \mathbb{R}^{m\times n}$ proposed in Halko et al. (2011) consists of several simple steps. The main idea is to obtain a good estimate for the range of A by forming products of A with a sample of random vectors, then using the orthogonal basis of this sample matrix to project the original matrix into a smaller, lower dimensional one, of which we extract the full SVD and use these components to construct the low rank SVD of the original big matrix A. The steps are as follows:

- Take k samples of the range of matrix A by multiplying A with random Gaussian vectors to form sample matrix Y of size $m \times k$. We then have range $Y \approx$ range A.
- Obtain an orthogonal matrix Q from Y (by e.g. performing QR factorization on Y to get $Y = QR$, where $Q^T Q = I$ and R is upper triangular). Then range $Q \approx$ range $A \implies QQ^T A \approx A$.

- Project the original matrix into a lower dimensional one: $B = Q^T A$ where B is $k \times n$, substantially smaller than A which is $m \times n$.
- Take the SVD of the smaller matrix $B = \tilde{U}_k \Sigma_k V_k^T$.
- Take as low rank SVD of A the product $U_k \Sigma_k V_k^T$ with $U_k = Q\tilde{U}_k$ (since $QQ^T A \approx A$).

Various interpretations of these steps from Halko et al. (2011), including description of developed open source software can be found in Voronin and Martinsson (2015). We describe here the details of one particular approach mentioned in Voronin and Martinsson (2015), and formulate it in a way which can be used for very large matrices. In the approach we use, we construct a smaller matrix BB^T and work with this matrix instead of B, because the matrix B of size $k \times n$, can still be quite large for large n. We compute the SVD components $\tilde{U}_k$ and Σ_k of B using the eigendecomposition of the small $k \times k$ symmetric matrix BB^T and obtain V_k by applying B^T. This way, we avoid building B or taking the SVD of it directly. We use the following relations:

$$B = \tilde{U}_k \Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i \tilde{u}_i v_i^T; \quad B^T = V_k \Sigma_k \tilde{U}_k^T; \quad Bv_i = \sigma_i \tilde{u}_i;$$

$$\begin{aligned} BB^T &= \left(\sum_{i=1}^{k} \sigma_i \tilde{u}_i v_i^T\right)\left(\sum_{j=1}^{k} \sigma_j \tilde{u}_j v_j^T\right)^T = \sum_{i,j=1}^{k} \sigma_i \sigma_j \tilde{u}_i v_i^T v_j \tilde{u}_j^T \\ &= \sum_{i=1}^{k} \sigma_i^2 \tilde{u}_i \tilde{u}_i^T = \tilde{U}_k D_k \tilde{U}_k^T. \end{aligned}$$

This means the eigendecomposition of the $k \times k$ matrix BB^T gives us the low rank SVD components $U_k = Q\tilde{U}_k$ and $\Sigma_k = \sqrt{D_k}$ element-wise. To compute the right eigenvectors v_i, we can use the following relations:

$$B^T \tilde{U}_k = V_k \Sigma_k \tilde{U}_k^T \tilde{U}_k = V_k \Sigma_k \implies B^T \tilde{U}_k \Sigma_k^{-1} = V_k,$$

which implies:

$$v_i = V_k e_i = (B^T \tilde{U}_k \Sigma_k^{-1}) e_i = \tfrac{1}{\sigma_i} B^T \tilde{u}_i = \tfrac{1}{\sigma_i} A^T Q \tilde{u}_i, \tag{5.1}$$

assuming all the singular values in Σ_k are above zero (which is the case for k smaller than the numerical rank r).

Notice that all matrix–vector operations involving A and A^T can be approximated via the wavelet compressed matrices M and M^T. To build up BB^T column by column we can use matrix–vector products with standard basis vectors e_j:

$$BB^T e_j = Q^T A A^T Q e_j \approx Q^T M W^{-T} W^{-1} M^T Q e_j, \tag{5.2}$$

and for the right eigenvectors, we have from (5.1) that:

$$v_i = \tfrac{1}{\sigma_i} A^T Q u_i \approx \tfrac{1}{\sigma_i} W^{-1} M^T Q u_i.$$

We now illustrate the main steps of the random algorithm to compute the low rank SVD, which we use in our computations for the numerical experiments. Below, we use Matlab like pseudocode.

- Take k samples of matrix A with random Gaussian vectors and perform Gram–Schmidt orthogonalization to calculate the projection matrix Q.

```
for j=1:k
    rj = randn(n,1);
    yj = A*rj;
    Y(:,j) = yj;
end

Q = Y;
for ind=1:2
    for j=1:k
        vj = Q(:,j);
        for i=1:(j-1)
            vi = Q(:,i);
            vj = vj - project_vec(vj,vi);
        end
        vj = vj/norm(vj);
        Q(:,j) = vj;
    end
end
```

  where the projection of v in direction of u is defined as $\frac{(v \cdot u)}{||u||_2^2} u$. For best results, the Gram–Schmidt orthogonalization should be performed twice to account for loss of orthogonality. Note that for matrix–vector multiplications with A we use $Ar_k \approx MW^{-T} r_k$.
- Build the $k \times k$ matrix $BB^T = Q^T AA^T Q$ by computing k matrix–vector products with standard basis vectors.
  Once we have built Q and its transpose, we can form the matrix BB^T column by column:

```
BBt = zeros(k,k);
for j=1:k
    ej = zeros(k,1);
    ej(j) = 1;
    colj = Qt*(A*(At*(Q*ej)));
    BBt(:,j) = colj;
end
```

  Here, we would make use of (5.2) for approximating $Q^T AA^T Qe_j$.
- Compute the eigendecomposition of BB^T.
  This simply is the eigendecomposition of a small $k \times k$ matrix:

```
[Uhat,D] = eig(BBt);
```

- Compute the low rank SVD components of A by using the eigendecomposition derived in the previous step and applying $B^T = A^T Q$ to eigenvectors.

Here we use the fact that the eigenvalues of BB^T are the squares of the singular values of B and the computation (5.1) for the eigenvectors V.

```
Sigma = zeros(k,k);
for i=1:k
    Sigma(i,i) = sqrt(D(i,i));
end

U = Q * Uhat;

V = zeros(n,k);
for j=1:k
    vj = 1/Sigma(j,j) * (At * U(:,j));
    V(:,j) = vj;
end
```

Here, we can use $A^T u_j \approx W^{-1} M^T u_j$.

We note that the implementation of the low rank SVD algorithm above is simple, as long as we can perform matrix–vector operations using the wavelet compressed matrix M and compute the eigendecomposition of a small $k \times k$ matrix, which can be done with a large number of available numerical packages. The disadvantage of this version is that working with the matrix BB^T essentially squares the condition number of A, such that small singular values near machine precision may not be properly resolved. This is an issue if A is expected to have very small singular values amongst $\sigma_1, \ldots, \sigma_k$. However, if we take k to be small relative to $\min(m, n)$ as we do in our application, σ_k is significantly larger in magnitude than machine precision. The implementation of the algorithm in the pseudocode above is not very efficient for the randomized algorithm proposed in Halko et al. (2011), but one that is practical to use for very large A when the corresponding wavelet compressed matrix $M = \mathbb{T}(AW^T)$ is available. In particular, for a more efficient implementation, one may want to block as many operations as possible, replacing matrix–vector by matrix–matrix multiplications. If possible, one may want to explicitly compute the matrix B and then use it to form BB^T. Likewise, V_k can be calculated directly from the matrix product $B^T \tilde{U}_k \Sigma_k^{-1}$. A power iteration strategy can also be implemented to improve accuracy in cases where the tail singular values decay more slowly. We refer the reader to Voronin and Martinsson (2015) for more details.

5.2 Application to regularization schemes

For purposes of iterative regularization algorithms, we can make use of the low rank SVD in several ways. If we obtain the low rank SVD of the whole matrix, we can directly use it to approximate matrix vector operations:

$$Ax \approx U_k \left(\Sigma_k (V_k^T x) \right) \quad \text{and} \quad A^T y \approx V_k \left(\Sigma_k (U_k^T y) \right), \tag{5.3}$$

and in some situations this is the most convenient and straightforward approach. The disadvantage of this approach is that one must keep the matrices U_k, U_k^T, V_k, V_k^T in memory. Here and below we do not pay attention to storing the matrix Σ_k which is a very small diagonal matrix in comparison to the former matrices. If the matrix A is large it may be difficult to compute the low rank SVD of the whole matrix A. Instead, if we block A as previously discussed, we can compute the low rank SVD of certain blocks or of each block. In some applications, it may be possible to arrange the blocks of A in a way that the first block of A contains many linearly dependent rows. If that is the case, then it is worthwhile to use the low rank SVD for the first block since it could be approximated well with small k. We can then write down mixed relations as follows:

$$Ax \approx \begin{bmatrix} U_{k_1}\Sigma_{k_1}V_{k_1}^T x \\ M_2 W_2^{-T} x \\ \vdots \\ M_p W_p^{-T} x \end{bmatrix} \quad \text{and} \quad A^T y \approx V_{k_1}\Sigma_{k_1}U_{k_1}^T y_1 + \sum_{j=2}^{p} W_j^{-1} M_j^T y_j, \tag{5.4}$$

where in this example we have used the low rank SVD approximation for the first part of the matrix and the wavelet based approximation for the other parts.

Additional information can be learned by plugging in the low rank SVD directly into the regularization system. Our general model problem and its corresponding linear system are:

$$\begin{aligned} \bar{x} &= \arg\min_x \left(||Ax - b||_2^2 + \lambda_1 ||x||_2^2 + \lambda_2 ||Lx||_2^2 \right) \\ &\implies (A^T A + \lambda_1 I + \lambda_2 L^T L)\bar{x} = A^T b. \end{aligned} \tag{5.5}$$

Replacing all instances of A by the low rank SVD results in:

$$\left(A_k^T A_k + \lambda_1 I + \lambda_2 L^T L \right) \tilde{x_1} = A_k^T b,$$

which when expanded gives:

$$\left(V_k \Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L \right) \tilde{x_1} = V_k \Sigma_k U_k^T b. \tag{5.6}$$

The advantage of (5.6) is that if the right hand side $V_k\Sigma_k U_k^T b$ is computed at the start of the iteration, only the matrices V_k and V_k^T must be kept in memory during the iteration. We may think of precomputing the right hand side $A^T b$ and approximating only the operator $A^T A$. Note that $A^T b$ can always be precomputed before the iteration as long as we can split up A into blocks. In this case we get:

$$\left(V_k \Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L \right) \hat{x_1} = A^T b. \tag{5.7}$$

As we will show later, this can result in slightly better error upper bound when the singular value σ_{k+1} is sufficiently small, though the norm of the solution for the same choice of λ_1 would be higher in this case. Another approach is to work with the lower dimensional projected system:

$$\left(U_k^T A\right) x = U_k^T b, \tag{5.8}$$

where $U_k^T A$ is $k \times n$ if $A \in \mathbb{R}^{m\times n}$. Note that we have the following simple result:

Lemma 5.1 *Given the low rank SVD $A_k = U_k \Sigma_k V_k^T$ of A, we have that $U_k^T A = U_k^T A_k = \Sigma_k V_k^T$.*

Proof First, $U_k^T A_k = U_k^T(U_k \Sigma_k V_k^T) = \Sigma_k V_k^T$. Also:

$$U_k^T A = U_k^T \left(U_k \Sigma_k V_k^T + \hat{U}_k \hat{\Sigma}_k \hat{V}_k^T\right) = \Sigma_k V_k^T + 0 = \Sigma_k V_k^T.$$

□

If we solve (5.8) by means of Tikhonov regularization:

$$\tilde{x_2} = \arg\min_x \left\{ ||(U_k^T A)x - U_k^T b||_2^2 + \lambda_1 ||x||_2^2 + \lambda_2 ||Lx||_2^2 \right\} \tag{5.9}$$

$$\implies \left((U_k^T A)^T (U_k^T A) + \lambda_1 I + \lambda_2 L^T L\right) \tilde{x_2} = \left(U_k^T A\right)^T U_k^T b, \tag{5.10}$$

we will obtain the same solution as (5.6):

Lemma 5.2 *The approximation scheme $(V_k \Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L)\tilde{x} = V_k \Sigma_k U_k^T b$ has the same solution as the Tikhonov regularized solution (5.9) of the projected system $(U_k^T A)x = U_k^T b$.*

Proof Since

$$\begin{aligned} A = U_k \Sigma_k V_k^T + \hat{U}_k \hat{\Sigma}_k \hat{V}_k^T &\implies U_k^T A = \Sigma_k V_k^T + 0 = \Sigma_k V_k^T \\ &\implies (U_k^T A)^T (U_k^T A) = (U_k^T A_k)^T (U_k^T A_k) = (\Sigma_k V_k^T)^T (\Sigma_k V_k^T) = V_k \Sigma_k^2 V_k^T, \end{aligned}$$

the linear system from (5.9) is equivalent to:

$$\left(V_k \Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L\right) \tilde{x_2} = (U_k^T A)^T U_k^T b = A^T U_k U_k^T b.$$

Next, for the right hand side we have:

$$\begin{aligned} A^T U_k &= V_k \Sigma_k U_k^T U_k + \hat{V}_k \hat{\Sigma}_k \hat{U}_k^T U_k = V_k \Sigma_k I + 0 \\ &= V_k \Sigma_k \implies A^T U_k U_k^T b = V_k \Sigma_k U_k^T b. \end{aligned}$$

Hence the solution of (5.9) is equivalent to that of (5.6):

$$\left(V_k \Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L\right) \tilde{x_2} = V_k \Sigma_k U_k^T b.$$

□

The advantage of (5.8) is that it may be convenient for larger systems where we can only perform the low rank SVD of its blocks. In that case, we may form the blocked system:

$$\begin{bmatrix} U_{k_1}^T A_1 \\ U_{k_2}^T A_2 \\ \vdots \\ U_{k_p}^T A_p \end{bmatrix} x = \begin{bmatrix} U_{k_1}^T b_1 \\ U_{k_2}^T b_2 \\ \vdots \\ U_{k_p}^T b_p \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \Sigma_{k_1} V_{k_1}^T \\ \Sigma_{k_2} V_{k_2}^T \\ \vdots \\ \Sigma_{k_p} V_{k_p}^T \end{bmatrix} x = \begin{bmatrix} U_{k_1}^T b_1 \\ U_{k_2}^T b_2 \\ \vdots \\ U_{k_p}^T b_p \end{bmatrix}, \tag{5.11}$$

and solve the optimization problem via the augmented normal equations:

$$\begin{bmatrix} U_{k_1}^T A_1 \\ U_{k_2}^T A_2 \\ \vdots \\ U_{k_p}^T A_p \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix}^T \begin{bmatrix} U_{k_1}^T A_1 \\ U_{k_2}^T A_2 \\ \vdots \\ U_{k_p}^T A_p \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix} \tilde{x_2} = \begin{bmatrix} U_{k_1}^T A_1 \\ U_{k_2}^T A_2 \\ \vdots \\ U_{k_p}^T A_p \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix}^T \begin{bmatrix} U_{k_1}^T b_1 \\ U_{k_2}^T b_2 \\ \vdots \\ U_{k_p}^T b_p \\ 0 \\ 0 \end{bmatrix} \quad \text{or}$$

$$\begin{bmatrix} \Sigma_{k_1} V_{k_1}^T \\ \Sigma_{k_2} V_{k_2}^T \\ \vdots \\ \Sigma_{k_p} V_{k_p}^T \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix}^T \begin{bmatrix} \Sigma_{k_1} V_{k_1}^T \\ \Sigma_{k_2} V_{k_2}^T \\ \vdots \\ \Sigma_{k_p} V_{k_p}^T \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix} \tilde{x_2} = \begin{bmatrix} \Sigma_{k_1} V_{k_1}^T \\ \Sigma_{k_2} V_{k_2}^T \\ \vdots \\ \Sigma_{k_p} V_{k_p}^T \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix}^T \begin{bmatrix} U_{k_1}^T b_1 \\ U_{k_2}^T b_2 \\ \vdots \\ U_{k_p}^T b_p \\ 0 \\ 0 \end{bmatrix}.$$

The number of eigenvectors for each block can be adjusted based on their conditioning. If the same k is used for all the blocks then some are bound to be projected less accurately than others. If the right hand side is precomputed, only the matrices $V_{k_j}^T$ and Σ_{k_j} must be in memory for each block. If it is easier to compute the eigenvector matrix U_k, then the default system with $U_k^T A$ may be useful.

A more aggressive approach is to use the right eigenvectors V_k to project the system from both sides to form a matrix of size $k \times k$. Instead of solving the full system:

$$(A^T A + \lambda_1 I + \lambda_2 L^T L)\bar{x} = A^T b,$$

we project the matrix used to a smaller space by multiplying on left by V_k^T and preconditioning on the right by V_k:

$$V_k^T (A^T A + \lambda_1 I + \lambda_2 L^T L)(V_k \tilde{y_3}) = V_k^T A^T b; \quad \tilde{x_3} = V_k \tilde{y_3}.$$

Expanding this and noting that $V_k^T V_k = I$, we have:

$$\left(V_k^T A^T A V_k + \lambda_1 I + \lambda_2 V_k^T L^T L V_k\right) \tilde{y_3} = V_k^T A^T b; \quad \tilde{x_3} = V_k \tilde{y_3}. \tag{5.12}$$

The key observation is that the matrix used in the linear system is $V_k^T A^T A V_k$, which is just of size $k \times k$, much smaller than the $m \times n$ matrix A. We can further simplify (5.12) using the following calculations:

Lemma 5.3 *Given the low rank SVD $A_k = U_k \Sigma_k V_k^T$ of A, we have that $V_k^T A^T A V_k = V_k^T A_k^T A_k V_k = \Sigma_k^2$ and $V_k A^T b = V_k A_k^T b = \Sigma_k U_k^T b$.*

Proof

$$\begin{aligned} V_k^T A^T &= V_k^T (V_k \Sigma_k U_k^T + \hat{V}_k \hat{\Sigma}_k \hat{U}_k^T) = V_k^T A_k^T = \Sigma_k U_k^T \\ &\implies A V_k = A_k V_k = (\Sigma_k U_k^T)^T = U_k \Sigma_k \\ &\implies V_k^T A^T A V_k = \Sigma_k U_k^T U_k \Sigma_k = \Sigma_k^2 \\ &\implies V_k A_k^T b = \Sigma_k U_k^T b. \end{aligned}$$

□

Thus, we can rewrite (5.12) as:

$$\left(\Sigma_k^2 + \lambda_1 I + \lambda_2 V_k^T L^T L V_k\right) \tilde{y_3} = \Sigma_k U_k^T b; \quad \tilde{x_3} = V_k \tilde{y_3}. \tag{5.13}$$

We will show later that when $\lambda_2 = 0$, $\tilde{x_3} = \tilde{x_1}$, an important result, since the system for $\tilde{y_3}$ can be solved on a small machine, as it involves just a $k \times k$ matrix. When $\lambda_2 \neq 0$, this is only an approximation. We can obtain the k columns of $V_k^T L^T L V_k$ by evaluating matrix vector products:

$$V_k^T L^T L V_k e_j \quad \text{for } j = 1, \ldots, k.$$

This is feasible to do in practice, since k is not very large. This method is useful when many solutions with different values of λ_1 and λ_2 are required, or when a rough guess to warm start a more accurate method is desired.

Let us now summarize the different techniques we have described for approximate ℓ_2 regularization using the low rank SVD and their computational requirements.

(1) We can implement $(A^T A + \lambda_1 I + \lambda_2 L^T L)\bar{x} = A^T b$ as usual and replace the operations Ax and $A^T y$ with $U_k \Sigma_k V_k^T x$ and $V_k \Sigma_k U_k^T y$. This requires one to have the matrices U_k, U_k^T, V_k, V_k^T in memory, which may not be very efficient. However, this direct approach may be useful for larger matrices split into blocks using relations such as (5.4), where the low rank SVD is applied only to certain blocks and not to the whole matrix. In that case, only the SVD components for the relevant blocks need to be loaded.

(2) We can plug in the low rank SVD into the regularization problem to get the system:

$$(V_k \Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L)\tilde{x_1} = V_k \Sigma_k U_k^T b.$$

Note that the right hand side $V_k \Sigma_k U_k^T b$ can be precomputed before the iteration so that only the matrices V_k and V_k^T need to be in memory during iteration. The result should be equivalent to the first case but this approach is more efficient. Additionally, we can precompute accurately the right hand side $A^T b$ and use the system:

$$(V_k \Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L)\hat{x_1} = A^T b.$$

Here the only difference is in the right hand side. As we will see later this can sometimes lead to solutions with a lower upper error bound, but should be used with a larger threshold for λ_1.

(3) We can utilize the lower dimensional projected system $U_k^T Ax = U_k^T b$. The corresponding system for the regularized problem:

$$\left((U_k^T A)^T (U_k^T A) + \lambda_1 I + \lambda_2 L^T L\right) \tilde{x_2} = (U_k^T A) U_k^T b$$

is equivalent to the system for $\tilde{x_1}$. However, in certain cases, the matrix U_k may be easier to compute than V_k (depending on the dimensions of $A^T A$ and AA^T) in which case one may then compute $U_k^T A$ by means of matrix–vector products $A^T U_k e_j$ for $j = 1, \dots, k$. The method may also be useful for large systems since we can make use of (5.11).

(4) We can use the $k \times k$ system:

$$\left(\Sigma_k^2 + \lambda_1 I + \lambda_2 V_k^T L^T L V_k\right) \tilde{y_3} = \Sigma_k U_k^T b; \quad \tilde{x}_3 = V_k \tilde{y}_3.$$

The solution of the linear system can be done on small memory computers since it involves the use of $k \times k$ matrices only and one multiplication with V_k at the end. The last step can be performed on a larger machine loading only V_k into memory; or on smaller machines in blocks. This scheme is useful when many runs with the system with different values of λ_1 and λ_2 are desired. The solution is equivalent to $\tilde{x_1}$ when $\lambda_2 = 0$ as shown later in this section.

Note that up to now we have discussed the application of the compression techniques to ℓ_2 norm minimization problems. However, the techniques are applicable to other types of regularization also. For example, for ℓ_1 regularization, where we minimize $||x||_1$ instead of $||x||_2$, one typically uses a scheme similar to the iterative soft thresholding algorithm (Daubechies et al. 2004):

$$x^{n+1} = \mathbb{S}_\tau(x^n + A^T b - A^T A x^n),$$

where $(\mathbb{S}_\tau(x))_k = \text{sgn}(x_k) \max\{0, |x_k| - \tau\}$ is the componentwise soft thresholding function. The main computational requirement here is in the operation $A^T A x^n$, just as for ℓ_2 regularization. Hence, many of the techniques we have described can be used for different types of regularization problems.

5.3 Further analysis and error bounds

In this section, we give more analysis for the SVD based schemes we have discussed. To make the analysis easier, we assume that $\lambda_1 = \lambda$ and $\lambda_2 = 0$ so we can do our analysis without the smoothing operator L, which is not approximated. Consider now the true solution:

$$\bar{x} = (A^T A + \lambda I)^{-1} A^T b \quad \text{(True Solution)}. \tag{5.14}$$

Notice that we can easily understand the significance of (5.14) by plugging in the (full rank) SVD $A = U \Sigma V^T$ into (5.14). One then obtains the solution:

$$\bar{x} = V D U^T b \quad \text{with } D = \text{Diag}\left(\frac{\sigma_1}{\sigma_1^2 + \lambda}, \frac{\sigma_2}{\sigma_2^2 + \lambda}, \ldots, \frac{\sigma_{\mathrm{r}}}{\sigma_{\mathrm{r}}^2 + \lambda}, 0, \ldots, 0\right).$$

We see that the regularization alleviates the effects of the singular vectors corresponding to small singular values σ_i, by replacing each σ_i by $\frac{\sigma_i}{\sigma_i^2+\lambda}$, which prevents the singular vectors corresponding to singular values smaller than λ from dominating the solution (Tikhonov 1963). Notice that while the application of Tikhonov minimization acts to filter the small singular values of A on the solution, the use of the low rank SVD A_k in place of A removes many of the small values entirely: the filtering is now done on those singular values which are retained.

We now restate the approximate solutions $\tilde{x_1}, \hat{x_1}, \tilde{x_2}, \tilde{x_3}$ that have been described in detail in the last section, but now with $\lambda_1 = \lambda$ and $\lambda_2 = 0$:

$$\tilde{x_1} = (A_k^T A_k + \lambda I)^{-1} A_k^T b, \tag{5.15}$$

$$\hat{x_1} = (A_k^T A_k + \lambda I)^{-1} A^T b, \tag{5.16}$$

$$\tilde{x_2} = \left((U_k^T A)^T (U_k^T A) + \lambda I\right)^{-1} (U_k^T A) U_k^T b, \tag{5.17}$$

$$\tilde{x_3} = V_k (\Sigma_k^2 + \lambda I)^{-1} \Sigma_k U_k^T b. \tag{5.18}$$

Recall here that $\tilde{x_1}$ and $\hat{x_1}$ correspond respectively, to (5.6) and (5.7), $\tilde{x_2}$ corresponds to (5.9), and $\tilde{x_3}$ corresponds to (5.12). We have previously shown that $\tilde{x_2}$ and $\tilde{x_1}$ have the same solution. We will show in this section that $\tilde{x_3}$ also has the same solution as $\tilde{x_1}$.

Using the Woodbury inverse formula (3.1), we can derive expressions relating the terms $(A_k^T A_k + \lambda I)^{-1}$ and $(A^T A + \lambda I)^{-1}$ which appear in the solutions $\tilde{x_1}, \hat{x_1}, \tilde{x_2}, \tilde{x_3}$ and in the true solution $\bar{x}$.

Lemma 5.4 *Let k be in the range $1 \le k \le r - 1$ and $\lambda > 0$. Then:*

$$(A_k^T A_k + \lambda I)^{-1} = \lambda^{-1} I - V_k S_k V_k^T \quad \textit{with}$$
$$S_k = \text{Diag}\left(\frac{\sigma_s^2}{\lambda^2 + \lambda \sigma_s^2}\right) \quad \textit{for } s = 1, \ldots, k, \tag{5.19}$$

and:

$$(A^T A + \lambda I)^{-1} = (A_k^T A_k + \lambda I)^{-1} - \hat{V}_k \hat{S}_k \hat{V}_k^T$$
$$\textit{with } \hat{S}_k = \text{Diag}\left(\frac{\sigma_s^2}{\lambda^2 + \lambda \sigma_s^2}\right) \quad \textit{for } s = k+1, \ldots, r. \tag{5.20}$$

These imply that:

$$\bar{x} = \left((A_k^T A_k + \lambda I)^{-1} - \hat{V}_k \hat{S}_k \hat{V}_k^T\right) A^T b, \tag{5.21}$$

$$\tilde{x_1} = (\lambda^{-1} I - V_k S_k V_k^T) A_k^T b, \tag{5.22}$$

$$\hat{x_1} = (\lambda^{-1} I - V_k S_k V_k^T) A^T b. \tag{5.23}$$

Proof The proof follows by the use of the Woodbury inverse formula (3.1):

$$(PTR + D)^{-1} = D^{-1} - D^{-1} P (R D^{-1} P + T^{-1})^{-1} R D^{-1}.$$

We match this with $(A_k^T A_k + \lambda I)^{-1} = (V_k \Sigma_k^2 V_k^T + \lambda I)^{-1}$ to get $P = V_k$, $R = V_k^T$, $T = \Sigma_k^2$, and $D = \lambda I$:

$$\begin{aligned}
(A_k^T A_k + \lambda I)^{-1} &= \lambda^{-1} I - \lambda^{-1} V_k (V_k^T \lambda^{-1} V_k + \Sigma_k^{-2})^{-1} V_k^T \lambda^{-1} \\
&= \lambda^{-1} I - \lambda^{-2} V_k (\Sigma_k^{-2} + \lambda^{-1} I)^{-1} V_k^T \\
&= \lambda^{-1} I - \lambda^{-2} V_k \left(\mathrm{Diag}(\sigma_1^{-2}, \ldots, \sigma_k^{-2}) + \lambda^{-1} I\right)^{-1} V_k^T \\
&= \lambda^{-1} I - \lambda^{-2} V_k \,\mathrm{Diag}(\sigma_1^{-2} + \lambda^{-1}, \ldots, \sigma_k^{-2} + \lambda^{-1})^{-1} V_k^T \\
&= \lambda^{-1} I - \lambda^{-2} V_k \,\mathrm{Diag}\left((\sigma_1^{-2} + \lambda^{-1})^{-1}, \ldots, (\sigma_k^{-2} + \lambda^{-1})^{-1}\right) V_k^T \\
&= \lambda^{-1} I - \lambda^{-2} V_k \,\mathrm{Diag}\left(\frac{\lambda \sigma_1^2}{\lambda + \sigma_1^2}, \ldots, \frac{\lambda \sigma_k^2}{\lambda + \sigma_k^2}\right) V_k^T \\
&= \lambda^{-1} I - V_k \,\mathrm{Diag}\left(\frac{\sigma_1^2}{\lambda^2 + \lambda \sigma_1^2}, \ldots, \frac{\sigma_k^2}{\lambda^2 + \lambda \sigma_k^2}\right) V_k^T \\
&= \lambda^{-1} I - V_k S_k V_k^T,
\end{aligned}$$

which proves (5.19).

For (5.20), we have:

$$(A^T A + \lambda I)^{-1} = (A_k^T A_k + \hat{V}_k \hat{\Sigma}_k^2 \hat{V}_k^T + \lambda I)^{-1} = (\hat{V}_k \hat{\Sigma}_k^2 \hat{V}_k^T + Y)^{-1},$$

with $Y = A_k^T A_k + \lambda I$. Using Woodbury matrix formula:

$$(\hat{V}_k \hat{\Sigma}_k^2 \hat{V}_k^T + Y)^{-1} = Y^{-1} - Y^{-1} \hat{V}_k (\hat{\Sigma}_k^{-2} + \hat{V}_k^T Y^{-1} \hat{V}_k)^{-1} \hat{V}_k^T Y^{-1}.$$

Now, by (5.19) we have $Y^{-1} = \lambda^{-1} I - V_k S_k V_k^T$ and by orthogonality we have $\hat{V}_k^T V_k = 0$:

$$\begin{gathered}
\hat{V}_k^T Y^{-1} = \hat{V}_k^T (\lambda^{-1} I - V_k S_k V_k^T) = \lambda^{-1} \hat{V}_k^T \\
Y^{-1} \hat{V}_k = (\lambda^{-1} I - V_k S_k V_k^T) \hat{V}_k = \lambda^{-1} \hat{V}_k.
\end{gathered}$$

Thus:

$$
\begin{aligned}
(A^T A + \lambda I)^{-1} &= Y^{-1} - Y^{-1}\hat{V}_k(\hat{\Sigma}_k^{-2} + \hat{V}_k^T Y^{-1}\hat{V}_k)^{-1}\hat{V}_k^T Y^{-1} \\
&= Y^{-1} - \lambda^{-1}\hat{V}_k(\hat{\Sigma}_k^{-2} + \hat{V}_k^T \lambda^{-1}\hat{V}_k)^{-1}\lambda^{-1}\hat{V}_k^T \\
&= Y^{-1} - \lambda^{-2}\hat{V}_k(\hat{\Sigma}_k^{-2} + \lambda^{-1} I)^{-1}\hat{V}_k^T \\
&= Y^{-1} - \lambda^{-2}\hat{V}_k \operatorname{Diag}\left(\frac{\lambda + \sigma_{k+1}^2}{\lambda\sigma_{k+1}^2}, \ldots, \frac{\lambda + \sigma_r^2}{\lambda\sigma_r^2}\right)^{-1} \hat{V}_k^T \\
&= Y^{-1} - \lambda^{-2}\hat{V}_k \operatorname{Diag}\left(\frac{\lambda\sigma_{k+1}^2}{\lambda + \sigma_{k+1}^2}, \ldots, \frac{\lambda\sigma_r^2}{\lambda + \sigma_r^2}\right)\hat{V}_k^T \\
&= (A_k^T A_k + \lambda I)^{-1} - \hat{V}_k \operatorname{Diag}\left(\frac{\sigma_1^2}{\lambda^2 + \lambda\sigma_{k+1}^2}, \ldots, \frac{\sigma_r^2}{\lambda^2 + \lambda\sigma_r^2}\right)\hat{V}_k^T \\
&= (A_k^T A_k + \lambda I)^{-1} - \hat{V}_k \hat{S}_k \hat{V}_k^T,
\end{aligned}
$$

which proves (5.20).
Equations (5.19) and (5.20) imply that:

$$
\begin{aligned}
\bar{x} &= (A^T A + \lambda I)^{-1} A^T b = \left((A_k^T A_k + \lambda I)^{-1} - \hat{V}_k \hat{S}_k \hat{V}_k^T\right) A^T b, \\
\tilde{x_1} &= (A_k^T A_k + \lambda I)^{-1} A_k^T b = (\lambda^{-1} I - V_k S_k V_k^T) A_k^T b, \\
\hat{x_1} &= (A_k^T A_k + \lambda I)^{-1} A^T b = (\lambda^{-1} I - V_k S_k V_k^T) A^T b.
\end{aligned}
$$

□

Now we show that $\tilde{x_3}$ (involving the inversion of a $k \times k$ matrix) has the same solution as $\tilde{x_1}$ and derive the expression for the difference between $\tilde{x_1}$ and $\hat{x_1}$.

Lemma 5.5 *Let $\bar{x}$ be the solution of (5.14), $\tilde{x_1}$ the solution of (5.15), $\hat{x_1}$ the solution of (5.16) and $\tilde{x_3}$ the solution of (5.18). Then, we have:*

$$\tilde{x_3} = \tilde{x_1}, \tag{5.24}$$

and

$$\hat{x_1} - \tilde{x_1} = \lambda^{-1}(A^T - A_k^T)b = \lambda^{-1}\hat{A_k}^T b. \tag{5.25}$$

Proof First note that:

$$V_k V_k^T A_k^T b = V_k V_k^T V_k \Sigma_k U_k^T b = V_k \Sigma_k U_k^T b = A_k^T b.$$

Next, we expand:

$$
\begin{aligned}
\tilde{x_1} &= (\lambda^{-1} I - V_k S_k V_k^T) A_k^T b = \lambda^{-1} A_k^T b - V_k S_k V_k^T A_k^T b \\
&= \lambda^{-1} V_k V_k^T A_k^T b - V_k S_k V_k^T A_k^T b
\end{aligned}
$$

$$\begin{aligned}
&= V_k(\lambda^{-1}I - S_k)V_k^T A_k^T b = V_k\left(\lambda^{-1}I - \text{Diag}\left(\frac{\sigma_s^2}{\lambda^2 + \lambda\sigma_s^2}\right)\right)V_k^T A_k^T b\\
&= V_k \,\text{Diag}\left(\frac{1}{\lambda} - \frac{\sigma_s^2}{\lambda^2 + \lambda\sigma_s^2}\right)V_k^T A_k^T b\\
&= V_k \,\text{Diag}\left(\frac{(\sigma_s^2 + \lambda) - \sigma_s^2}{\lambda(\sigma_s^2 + \lambda)}\right)V_k^T A_k^T b\\
&= V_k \,\text{Diag}\left(\frac{1}{\sigma_s^2 + \lambda}\right)V_k^T A_k^T b = V_k(\Sigma_k^2 + \lambda I)^{-1}V_k^T A_k^T b = \tilde{x_3},
\end{aligned}$$

which proves (5.24). Next, for the difference between $\tilde{x_1}$ and $\hat{x_1}$ we have:

$$\begin{aligned}
\tilde{x_1} &= (A_k^T A_k + \lambda I)^{-1}A_k^T b = (\lambda^{-1}I - V_k S_k V_k^T)A_k^T b = \lambda^{-1}A_k^T b - V_k S_k V_k^T A_k^T b,\\
\hat{x_1} &= (A_k^T A_k + \lambda I)^{-1}A^T b = (\lambda^{-1}I - V_k S_k V_k^T)A^T b = \lambda^{-1}A^T b - V_k S_k V_k^T A^T b.
\end{aligned}$$

Note that:

$$V_k S_k V_k^T A^T b = V_k S_k V_k^T(V_k\Sigma_k U_k^T + \hat{V}_k\hat{\Sigma}_k\hat{U}_k^T)b = V_k S_k V_k^T A_k^T b.$$

Hence:

$$\hat{x_1} - \tilde{x_1} = \lambda^{-1}A^T b - \lambda^{-1}A_k^T b = \lambda^{-1}(A^T - A_k^T)b = \lambda^{-1}\hat{A_k}^T b,$$

which proves (5.25). □

By the result of Lemma 5.5, the only solutions which differ from each other are $\tilde{x_1}$ and $\hat{x_1}$. We now analyze these two solutions with respect to the true solution $\bar{x}$.

Proposition 5.6 *Let $\bar{x}$ be the solution of* (5.14) *and $\tilde{x_1}$ the solution of* (5.15). *Then*:

$$||\bar{x} - \tilde{x_1}||_2 \le \frac{\sigma_{k+1}}{\lambda + \sigma_{k+1}^2}||b||_2, \tag{5.26}$$

and

$$\tilde{x_1} = V_k V_k^T \bar{x}. \tag{5.27}$$

Proof Recall that $\tilde{x_1} = (A_k^T A_k + \lambda I)^{-1}A_k^T b$ and that $\bar{x} = (A^T A + \lambda I)^{-1}A^T b$. Next by Lemma 5.4 and using that $A_k\hat{V}_k = (U_k\Sigma_k V_k^T)\hat{V}_k = 0$ and $\hat{V}_k^T V_k = 0$:

$$\begin{aligned}
(A^T A + \lambda I)^{-1}A^T &= (A^T A + \lambda I)^{-1}(A_k^T + \hat{V}_k\hat{\Sigma}_k\hat{U}_k^T)\\
&= \left((A_k^T A_k + \lambda I)^{-1} - \hat{V}_k\hat{S}_k\hat{V}_k^T\right)(A_k^T + \hat{V}_k\hat{\Sigma}_k\hat{U}_k^T)\\
&= (A_k^T A_k + \lambda I)^{-1}A_k^T + (A_k^T A_k + \lambda I)^{-1}\hat{V}_k\hat{\Sigma}_k\hat{U}_k^T - \hat{V}_k\hat{S}_k\hat{\Sigma}_k\hat{U}_k^T\\
&= (A_k^T A_k + \lambda I)^{-1}A_k^T + (\lambda^{-1}I - V_k S_k V_k^T)\hat{V}_k\hat{\Sigma}_k\hat{U}_k^T - \hat{V}_k\hat{S}_k\hat{\Sigma}_k\hat{U}_k^T\\
&= (A_k^T A_k + \lambda I)^{-1}A_k^T + \lambda^{-1}\hat{V}_k\hat{\Sigma}_k\hat{U}_k^T - \hat{V}_k\hat{S}_k\hat{\Sigma}_k\hat{U}_k^T\\
&= (A_k^T A_k + \lambda I)^{-1}A_k^T + \hat{V}_k(\lambda^{-1}\hat{\Sigma}_k - \hat{S}_k\hat{\Sigma}_k)\hat{U}_k^T.
\end{aligned}$$

Since $\hat{S}_k = \text{Diag}\left(\frac{\sigma_s^2}{\lambda^2+\lambda\sigma_s^2}\right)$ for $s = (k+1), \ldots, r$:

$$\lambda^{-1}\hat{\Sigma}_k - \hat{S}_k\hat{\Sigma}_k = \text{Diag}\left(\frac{\sigma_s}{\lambda} - \frac{\sigma_s^3}{\lambda(\lambda+\sigma_s^2)}\right) = \text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right) \quad \text{for } s = (k+1), \ldots, r.$$

Hence:

$$(A^TA + \lambda I)^{-1}A^T = (A_k^TA_k + \lambda I)^{-1}A_k^T + \hat{V}_k\,\text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right)\hat{U}_k^T,$$

which implies:

$$\begin{aligned}\bar{x} &= (A^TA+\lambda I)^{-1}A^Tb = (A_k^TA_k+\lambda I)^{-1}A_k^Tb \\ &\quad +\hat{V}_k\,\text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right)\hat{U}_k^T \qquad (5.28)\\ &= \tilde{x}_1 + \hat{V}_k\,\text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right)\hat{U}_k^Tb \qquad (5.29)\\ \implies ||\bar{x}-\tilde{x}_1||_2 &= \left\|\hat{V}_k\,\text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right)\hat{U}_k^Tb\right\| = \left\|\text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right)\hat{U}_k^Tb\right\|\\ &\le \left\|\text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right)\right\|_2 ||b||_2\\ &\le \frac{\sigma_{k+1}}{\lambda+\sigma_{k+1}^2}||b||_2,\end{aligned}$$

which proves (5.26).

Next, to derive (5.27), we have:

$$A^Tb = (V_k\Sigma_kU_k^T)b + (\hat{V}_k\hat{\Sigma}_k\hat{U}_k^T)b,$$

so that

$$\begin{aligned}&\hat{V}_k\hat{\Sigma}_k\hat{U}_k^Tb = A^Tb - (V_k\Sigma_kU_k^T)b \implies \hat{V}_k^T\hat{V}_k\hat{\Sigma}_k\hat{U}_k^Tb = \hat{\Sigma}_k\hat{U}_k^Tb\\ &\quad = \hat{V}_k^TA^Tb - 0 \implies \hat{U}_k^Tb = \hat{\Sigma}_k^{-1}\hat{V}_k^TA^Tb\\ &\quad \implies \hat{U}_k^Tb = \hat{\Sigma}_k^{-1}\hat{V}_k^T(A^TA+\lambda I)\bar{x} = \hat{\Sigma}_k^{-1}\hat{V}_k^T(V_k\Sigma_k^2V_k^T + \hat{V}_k\hat{\Sigma}_k^2\hat{V}_k^T + \lambda I)\bar{x}\\ &\quad = \hat{\Sigma}_k^{-1}(\hat{\Sigma}_k^2\hat{V}_k^T + \lambda\hat{V}_k^T)\bar{x} = \hat{\Sigma}_k\hat{V}_k^T\bar{x} + \lambda\hat{\Sigma}_k^{-1}\hat{V}_k^T\bar{x} = (\hat{\Sigma}_k + \lambda\hat{\Sigma}_k^{-1})\hat{V}_k^T\bar{x}.\end{aligned}$$

Using (5.28), we have:

$$\begin{aligned}\bar{x} &= \tilde{x}_1 + \hat{V}_k\,\text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right)\hat{U}_k^Tb = \tilde{x}_1 + \hat{V}_k\,\text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right)(\hat{\Sigma}_k+\lambda\hat{\Sigma}_k^{-1})\hat{V}_k^T\bar{x}\\ &= \tilde{x}_1 + \hat{V}_k\,\text{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right)\text{Diag}\left(\sigma_s + \frac{\lambda}{\sigma_s}\right)\hat{V}_k^T\bar{x} = \tilde{x}_1\end{aligned}$$

$$
\begin{aligned}
&+\hat{V}_k \operatorname{Diag}\left(\frac{\sigma_s}{\lambda+\sigma_s^2}\right) \operatorname{Diag}\left(\frac{\sigma_s^2+\lambda}{\sigma_s}\right) \hat{V}_k^T \bar{x} \\
&= \tilde{x_1} + \hat{V}_k \hat{V}_k^T \bar{x} = \tilde{x_1} + (I - V_k V_k^T)\bar{x} = \tilde{x_1} + \bar{x} - V_k V_k^T \bar{x}.
\end{aligned}
$$

This proves (5.27):

$$\tilde{x_1} = V_k V_k^T \bar{x}.$$

□

Next, we look at the solution $\hat{x_1} = (A_k^T A_k + \lambda I)^{-1} A^T b$. Recall that the difference from $\tilde{x_1}$ is that in $\hat{x_1}$, $A^T b$ is not approximated by $A_k^T b$.

Proposition 5.7 *Let $\bar{x}$ be the solution of* (5.14) *and $\hat{x_1}$ the solution of* (5.16). *Then:*

$$||\bar{x} - \hat{x_1}||_2 \le \frac{\sigma_{k+1}^3}{\lambda^2 + \lambda\sigma_{k+1}^2} ||b||_2, \tag{5.30}$$

and

$$\frac{||\bar{x} - \hat{x_1}||_2}{||\bar{x}||_2} \le \frac{\sigma_{k+1}^2}{\lambda}. \tag{5.31}$$

Proof We use Lemma 5.4 to relate $\bar{x}$ to $\hat{x_1}$.

$$\bar{x} = (A^T A + \lambda I)^{-1} A^T b = \left((A_k^T A_k + \lambda I)^{-1} - \hat{V}_k \hat{S}_k \hat{V}_k^T\right) A^T b \tag{5.32}$$

$$= \hat{x_1} - \hat{V}_k \hat{S}_k \hat{V}_k^T A^T b = \hat{x_1} - \hat{V}_k \hat{S}_k \hat{V}_k^T (A_k^T + \hat{V}_k \hat{\Sigma}_k \hat{U}_k^T) b = \hat{x_1} - \hat{V}_k \hat{S}_k \hat{\Sigma}_k \hat{U}_k^T b, \tag{5.33}$$

where the last equality follows from $\hat{V}_k^T A_k^T = 0$ and $\hat{V}_k^T \hat{V} = I$. Thus, we have:

$$
\begin{aligned}
&||\bar{x} - \hat{x_1}||_2 = ||\hat{V}_k \hat{S}_k \hat{\Sigma}_k \hat{U}_k^T b||_2 = ||\hat{S}_k \hat{\Sigma}_k \hat{U}_k^T b||_2 \le ||\hat{S}_k \hat{\Sigma}_k||_2 ||\hat{U}_k^T b||_2 \\
&\le ||\hat{S}_k \hat{\Sigma}_k||_2 ||b||_2.
\end{aligned}
$$

Now from Lemma 5.4:

$$
\begin{aligned}
\hat{S}_k \hat{\Sigma}_k &= \operatorname{Diag}\left(\frac{\sigma_{k+1}^3}{\lambda^2 + \lambda\sigma_{k+1}^2}, \frac{\sigma_{k+3}^2}{\lambda^2 + \lambda\sigma_{k+2}^2}, \ldots, \frac{\sigma_r^3}{\lambda^2 + \lambda\sigma_r^2}\right) \\
&\implies ||\hat{S}_k \hat{\Sigma}_k||_2 = \max(\hat{S}_k \hat{\Sigma}_k) = \frac{\sigma_{k+1}^3}{\lambda^2 + \lambda\sigma_{k+1}^2} \\
&\implies ||\hat{S}_k \hat{\Sigma}_k||_2 ||b||_2 = \frac{\sigma_{k+1}^3}{\lambda^2 + \lambda\sigma_{k+1}^2} ||b||_2.
\end{aligned}
$$

So we obtain the bound (5.30):

$$||\bar{x} - \hat{x_1}||_2 \le ||\hat{S}_k \hat{\Sigma}_k||_2 ||b||_2 = \frac{\sigma_{k+1}^3}{\lambda^2 + \lambda\sigma_{k+1}^2} ||b||_2.$$

In order to obtain (5.31), we need to get rid of the $||b||_2$ term. We appeal back to (5.33):

$$\begin{aligned}\bar{x} &= \hat{x_1} - \hat{V}_k \hat{S}_k \hat{V}_k^T A^T b = \hat{x_1} - \hat{V}_k \hat{S}_k \hat{V}_k^T (A^T A + \lambda I)\bar{x} \\ &= \hat{x_1} - \hat{V}_k \hat{S}_k \hat{V}_k^T (V_k \Sigma_k^2 V_k^T + \hat{V}_k \hat{\Sigma}_k^2 \hat{V}_k^T + \lambda I)\bar{x} = \hat{x_1} - \hat{V}_k \hat{S}_k (\hat{\Sigma}_k^2 + \lambda I)\hat{V}_k^T \bar{x}.\end{aligned}$$

It follows that:

$$\begin{aligned}||\bar{x} - \hat{x_1}||_2 &= ||\hat{V}_k \hat{S}_k (\hat{\Sigma}_k^2 + \lambda I)\hat{V}_k^T \bar{x}||_2 \le ||\hat{V}_k \hat{S}_k (\hat{\Sigma}_k^2 + \lambda I)\hat{V}_k^T||_2 ||\bar{x}||_2 \\ &= ||\hat{S}_k (\hat{\Sigma}_k^2 + \lambda I)||_2 ||\bar{x}||_2 \\ \implies \frac{||\bar{x} - \hat{x_1}||_2}{||\bar{x}||_2} &\le ||\hat{S}_k (\hat{\Sigma}_k^2 + \lambda I)||_2 \le ||\hat{S}_k||_2 ||(\hat{\Sigma}_k^2 + \lambda I)||_2 \\ &= \frac{\sigma_{k+1}^2}{\lambda^2 + \lambda\sigma_{k+1}^2}(\sigma_{k+1}^2 + \lambda),\end{aligned}$$

which simplifies to:

$$\frac{||\bar{x} - \hat{x_1}||_2}{||\bar{x}||_2} \le \frac{\sigma_{k+1}^2}{\lambda}.$$

□

Let us now recall some results we have derived. First of all, we have shown that $\tilde{x_1}, \tilde{x_2}$ and $\tilde{x_3}$ lead to the same solution. Numerically, however, one may still observe some differences if they are not run to convergence. On the other hand, $\tilde{x_1}$ and $\hat{x_1}$ differ from each other and have the following absolute error bounds with respect to the true solution $\bar{x}$:

$$\begin{aligned}||\bar{x} - \tilde{x_1}||_2 &\le \frac{\sigma_{k+1}}{\lambda + \sigma_{k+1}^2}||b||_2; \\ ||\bar{x} - \hat{x_1}||_2 &\le \frac{\sigma_{k+1}^3}{\lambda\left(\lambda + \sigma_{k+1}^2\right)}||b||_2.\end{aligned}$$

Recall that the difference between the two is in the right hand side: $\hat{x_1}$ uses the un-approximated right hand side, or at least one computed with the wavelet transformed matrix (i.e. $A^T b \approx W^{-1} M^T b$). We mention again that one operation with a large A or M is not prohibitively expensive as it can be done by splitting the matrix into small enough blocks. The plot below in Fig. 2 gives us a sense of how the upper bounds behave. We plot the fraction:

$$\beta = \frac{\frac{\sigma_{k+1}^3}{\lambda\left(\lambda+\sigma_{k+1}^2\right)} - \frac{\sigma_{k+1}}{\lambda+\sigma_{k+1}^2}}{\left|\frac{\sigma_{k+1}}{\lambda+\sigma_{k+1}^2}\right|} \tag{5.34}$$

as a function of the value of σ_{k+1} for two different choices of λ. The fraction (5.34) is simply a relative difference between the two upper bounds for the error of the approximate solutions $\hat{x_1}$ and $\tilde{x_1}$. From Fig. 2, we may observe that the difference fraction is negative (indicating a lower upper bound error for $\hat{x_1}$) when the value of σ_{k+1} is sufficiently small. However, if k is not large enough for σ_{k+1} to be sufficiently small then the upper bound of $\hat{x_1}$ will be worse than that of $\tilde{x_1}$. Another observation about the solution $\hat{x_1}$ compared to $\tilde{x_1}$ (and the other solutions equivalent to it) is that $\hat{x_1}$ for the same choice of λ is expected to have a larger norm:

Lemma 5.8 *Let $\tilde{x_1}$ be the solution of* (5.15) *and $\hat{x_1}$ the solution of* (5.16) *for a fixed value of λ. Then, we have that $||\tilde{x_1}||_2 \leq ||\hat{x_1}||_2$.*

Proof Recall that $A = A_k + \hat{A_k}$ and

$$\tilde{x_1} = (A_k^T A_k + \lambda I)^{-1} A_k^T b; \quad \hat{x_1} = (A_k^T A_k + \lambda I)^{-1} A^T b.$$

Now by Lemma 5.5:

$$\hat{x_1} = \tilde{x_1} + \lambda^{-1} \hat{A}_k^T b.$$

Thus, the norms are related as:

$$||\hat{x_1}||_2^2 = ||\tilde{x_1}||_2^2 + 2\lambda^{-1} \tilde{x_1}^T \hat{A}_k^T b + ||\hat{A}_k^T b||_2^2,$$

where the middle term is zero as we now show. Note that $\hat{A}_k A_k^T = \hat{A}_k V_k = 0$ and:

$$\begin{aligned}\left(\tilde{x_1}^T \hat{A}_k^T\right)^T &= \hat{A}_k \tilde{x_1} = \hat{A}_k (A_k^T A_k + \lambda I)^{-1} A_k^T b = \hat{A}_k (\lambda^{-1} I - V_k S_k V_k^T) A_k^T b \\ &= \lambda^{-1} \hat{A}_k A_k^T b + \hat{A}_k V_k S_k V_k^T A_k^T b = 0.\end{aligned}$$

Thus:

$$||\hat{x_1}||_2^2 = ||\tilde{x_1}||_2^2 + ||\hat{A}_k^T b||_2^2 \implies ||\tilde{x_1}||_2 \leq ||\hat{x_1}||_2.$$

□

Thus, when using $\hat{x_1}$ as an estimate for $\bar{x}$ we typically would like to take a larger value of λ to obtain a solution with similar norm to that of $\tilde{x_1}$. If we use the same λ for $\hat{x_1}$ and $\bar{x}$, we will find that the components of the solution of $\bar{x}$ have larger amplitudes.

6 Numerical experiments

In this section, we give some numerical examples to discuss and illustrate the approximation techniques we have discussed. We will use both synthetic data and matrices from the seismic tomography application which we have previously referred to in order to illustrate the effect of wavelet thresholding and low rank SVD based compression.

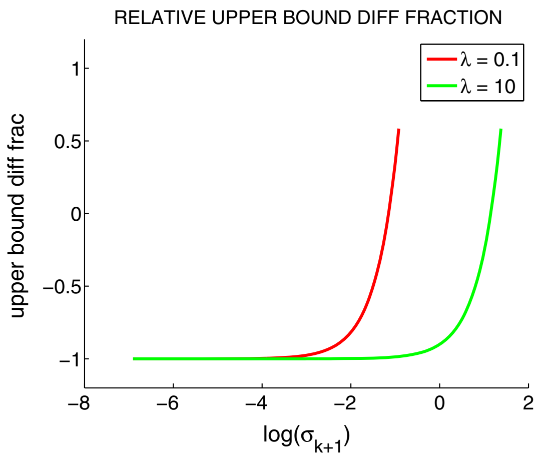

Fig. 2 Relative difference between *upper bounds* for the errors for approximate solutions $\tilde{x_1}$ and $\hat{x_1}$ [fraction (5.34)] as a function of different values of σ_{k+1}

6.1 Examples with synthetic data

We use three different synthetic matrix types, which we denote by $A_{(1)}$, $A_{(2)}$, and $A_{(3)}$. The matrices are of size 1000×1500, small enough to be easily handled in full, but large enough for randomization techniques to work. Matrix $A_{(1)}$ is constructed via the reverse SVD construction $A_{(1)} = U \Sigma V^T$ where U and V are taken to be orthonormal Gaussian random matrices and the singular values in Σ are logspaced between 10^0 and 10^{-4}. That is, the decay of singular values of $A_{(1)}$ is relatively fast. Matrix $A_{(2)}$ is a different kind of matrix, whose rows are permuted vectorized images. It is constructed by choosing at random, one of five images for each row, vectorizing the image and then using a randomized permutation of its vector form as a row of the matrix. Matrix $A_{(3)}$ is also constructed from the same vectorized images, but its rows are not randomly permuted vectors but rather vectors rearranged in a continuous way with overlooping boundaries, where we choose at random a starting index within the image vector and then go to the end of the array, looping back to the beginning and proceeding in order until we have n elements.

We now comment on the wavelet compressibility of each matrix. By "wavelet compressible" we mean that the rows of the matrices satisfy the relation (4.2). In our case, we apply the one dimensional CDF 9–7 wavelet transform to each row vector and threshold out all but $1/3$ of the largest coefficients by absolute magnitude. It should be apparent that the rows of $A_{(1)}$ are not readily wavelet compressible (as they are vectors picked at random having no apparent structure), some but not all of the rows of $A_{(2)}$ are wavelet compressible (as they are image vectors re-arranged in random order so that only rows arranged by chance in such a way as to have some structure are expected to be compressible), and virtually all rows of $A_{(3)}$ are readily wavelet compressible (they are vectorized images with a random starting index, but the pixel structure of the original image is preserved).

We start by constructing the compressed wavelet matrices $M_{(1)}$, $M_{(2)}$, and $M_{(3)}$, keeping a third of the nonzero wavelet coefficients in the thresholding. We then compare the errors induced in approximating matrix vector operations with the full

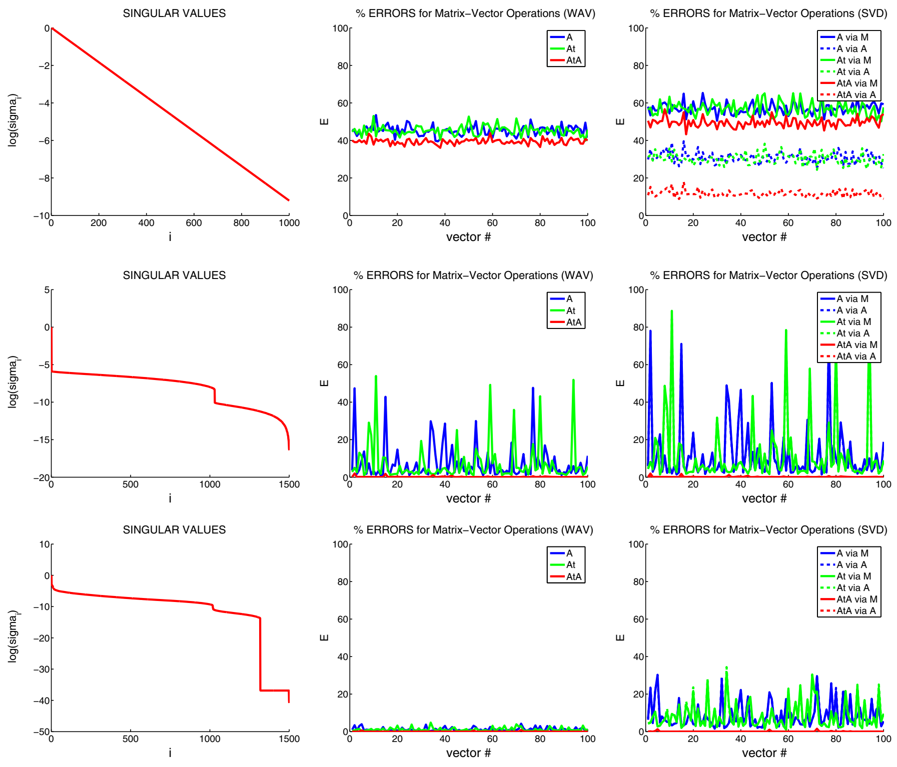

Fig. 3 Percent errors for approximating matrix–vector operations with $A_{(1)}$ (*row 1*), $A_{(2)}$ (*row 2*), and $A_{(3)}$ (*row 3*) via the wavelet compressed matrices $M_{(i)}$ and via the low rank SVD. Singular values (*column 1*), percent errors in matrix vector operations for 100 Gaussian random vectors using wavelet compression (*column 2*) and low rank SVD (*column 3*). In *column 3* we plot errors obtained via the low rank SVD approximation obtained using the full $A_{(i)}$ matrices and using the corresponding wavelet compressed $M_{(i)}$ matrices

matrices $A_{(1)}, A_{(2)}, A_{(3)}$ via these compressed matrices using the relations (4.4). For 100 Gaussian random vectors $x \in \mathbb{R}^{1500}$ and $y \in \mathbb{R}^{1000}$ we compare, using (4.4), the results of the operations $A_{(i)}x$ versus $M_{(i)}W^{-T}x$, $A^T_{(i)}y$ versus $W^{-1}M^T_{(i)}y$ and $A^T_{(i)}A_{(i)}x$ versus $W^{-1}M^T_{(i)}M_{(i)}W^{-T}x$ for $i = 1, 2, 3$ corresponding to the three matrices. The resulting percent errors (i.e. fractions such as $E = 100\frac{\|A_{(1)}x - M_{(1)}W^{-T}x\|}{\|A_{(1)}x\|}$ and likewise for the other operations) are plotted in column 2 of Fig. 3, where we plot median values over 10 trials and in each trial utilize 100 Gaussian random vectors x and y. Notice that in the first case, where the matrix was chosen to not compress well, the errors are high. In the other two cases, the operations with matrices $A_i^T A_i$ ($i = 2, 3$) are approximated well. It is especially interesting that this is the case for the second matrix, where some of the rows are not wavelet compressible.

Next, we use the $M_{(i)}$ matrices to compute the low rank SVD of $A_{(i)}$ with $k = 200$ to achieve further size reduction. That is, we use the randomized SVD algorithm previously shown where we utilize matrix M to approximate all necessary operations with A. Once the low rank SVD components U_k, Σ_k, and V_k are obtained, we compare the same operations with A as before to the approximation via the low rank SVD:

$$Ax \text{ to } U_k \Sigma_k V_k^T x; \quad A^T y \text{ to } V_k \Sigma_k U_k^T y; \quad A^T A x \text{ to } V_k \Sigma_k^2 V_k^T x.$$

For comparison, for each matrix, we also compute the low rank SVD with the full $A_{(i)}$, without using $M_{(i)}$ to approximate matrix–vector operations. We expect this to give a more accurate low rank SVD. The corresponding percent errors (such as $E = 100\frac{\|A_{(1)}x - U_k \Sigma_k V_k^T x\|}{\|A_{(1)}x\|}$) for the operations are shown in column 3 of Fig. 3. In all cases, the plotted lines are median values obtained over ten separate trials. The result is interesting but somewhat expected because of the use of randomization in the computation: the low rank SVD computed via M produces similar results to that computed via A even if for some particular row vectors of A, the relation (4.2) is not satisfied. However, notice that this does not hold for matrix $A_{(1)}$ whose rows are not wavelet compressible. From the last column of Fig. 3, we see differences between the results of the low rank SVD computed with $A_{(1)}$ and with $M_{(1)}$.

Next, we make a synthetic data vector x, and use the three matrices $A_{(1)}$, $A_{(2)}$, $A_{(3)}$ to construct the right hand side $b_{(i)} = A_{(i)}x + \nu$ with ν a Gaussian random noise vector (we choose to use 10 percent noise relative to the norm of $b_{(i)}$). We then try to reconstruct x with the various approximation schemes by computing solutions to the Tikhonov problem with smoothing $(A^T A + \lambda_1 I + \lambda_2 L^T L)\bar{x} = A^T b$, where for L we take the tridiagonal matrix with elements $(-1, 2, -1)$. In Fig. 4, we present the results of various approximation schemes we described. In particular, we plot the following solutions:

$$\begin{aligned}
(A^T A + \lambda_1 I + \lambda_2 L^T L)\bar{x} &= (A^T b)\\
(W^{-1} M^T M W^{-T} + \lambda_1 I + \lambda_2 L^T L)^{-1} x_{wav} &= (W^{-1}(M^T b))\\
(V_k \Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L) x_{svd1} &= (V_k \Sigma_k U_k^T b)\\
(V_k \Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L) x_{svd2} &= (A^T b)\\
(\Sigma_k^2 + \lambda_1 I_k + \lambda_2 V_k^T L^T L V_k) y_{svd3} &= (\Sigma_k U_k^T b); \quad x_{svd3} = V_k y_{svd3}.
\end{aligned}$$

In each case, we loop over 40 linearly spaced values of λ_1 and λ_2 (effecting the degree of norm and smoothing penalty, respectively) and choose the values so that the residual norm $||Ax_{sol} - b||_2$ of the solution is closest to the norm of the noise vector $||\nu||_2$. In Fig. 4, we plot the on the first row the true solution vector x followed by the solutions obtained using the full matrix A. On the second row, we plot for each matrix type $(A_{(i)})$, the residual norms of the different solutions relative to the noise norm. On rows three to six, we plot the different solutions obtained with the various approximations schemes for the matrices $(A_{(i)})$. We observe that in each case, we can obtain reasonable reconstructions using the approximation schemes we introduced. The wavelet compressed approach is the most accurate with respect to the full solution, followed by the two svd methods. The $k \times k$ method (x_{svd3}) produces a suitable reconstruction for the third matrix $A_{(3)}$, whose rows are all wavelet compressible. On the other hand, the $k \times k$ method does not work well for the first two matrices. In summary, the synthetic

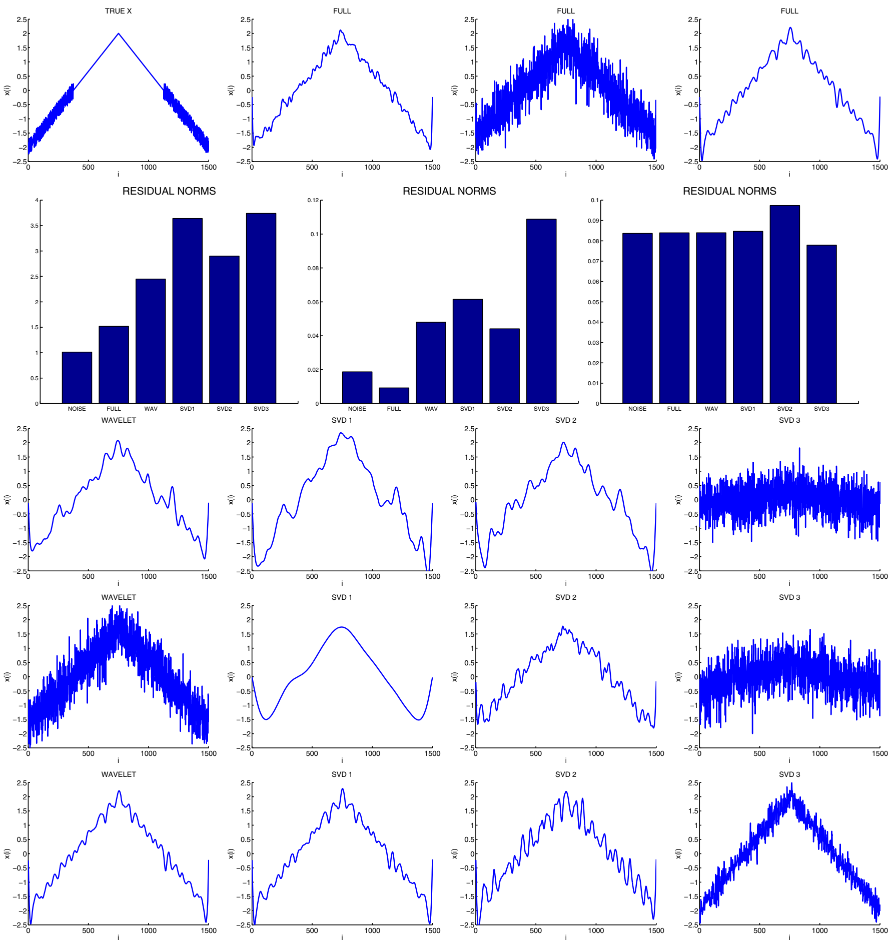

Fig. 4 *Row 1* actual signal x and reconstructions using the full matrices $A_{(1)}, A_{(2)}, A_{(3)}$. *Row 2 bar plots* of noise norm and solution residual norm values for each matrix system. *Rows 3–5 plots* of reconstructed solutions using the different compressed schemes with wavelet compression and low rank SVD for $A_{(1)}$ (*row 3*), $A_{(2)}$ (*row 4*), $A_{(3)}$ (*row 5*). For each SVD solution shown, the low rank SVD was obtained via the corresponding $M_{(i)}$ matrix

data examples show that in many practical cases, wavelet compression and low rank SVD techniques can be used together to obtain approximate regularized solutions, with the SVD matrices obtained using operations with the wavelet compressed matrix instead of the original matrix.

6.2 Examples with real data

We now illustrate examples with real data from an application in seismic tomography. We will keep our description of the problem and setup concise. Much details can be found in Simons et al. (2011) and other mentioned references. In short, we have a matrix A and a right hand side vector b from which we would like to obtain a vector x corresponding to corrections to a spherically symmetric model (which varies only with depth) of the seismic wave speeds in the Earth's interior. The idea is that these corrections can be used together with the spherically symmetric model in order to construct a three dimensional model of the wave speeds. The data comes from measurements made by seismometers on the surface of the Earth of different earthquakes in the Earth's interior.

The rows of our matrix A correspond to earthquake–receiver pairs, the number of which is very high (almost 3 million). It is to our advantage to include as many such pairs as possible. The more rows we include, the more information we include in the system and the more detailed the solution and hence model, which can be obtained. Each row is constructed from a surface wave data set (van Heijst and Woodhouse 1999), which has information corresponding to energy waves from earthquakes only close to the Earth's surface. The columns of the matrix correspond to the coordinate system that is used to grid the interior of the Earth between the surface and the core mantle boundary. Each row of the matrix A is a sensitivity kernel (Marquering et al. 1998), that is defined over a cubed-sphere coordinate system (Ronchi et al. 1996), in which the contents at the surface of a sphere of a given radius are projected onto six faces of a cube. We divide the region within the Earth between the core-mantle boundary and the surface into 37 depth layers each divided laterally into 6 chunks subdivided into 128×128 voxels. Each row of the matrix A (a kernel) has information for each of the 37 depth layers (corresponding to different radii from the core-mantle boundary to the Earth's surface) (Simons et al. 2011). This translates into approximately 3.6 million columns.

The matrix A is sparse, having approximately 1.5 % nonzeros. The resulting matrix is thus very large: the dimensions of the matrix A are $2{,}968{,}933 \times 3{,}637{,}248$ and it is approximately 3 TB in size on the disk in a double precision sparse format. The reason for the large size is apparent from a typical sparse storage scheme which stores the dimensions, the total number of nonzeros, the number of nonzerors in each row (or column), and the column (or row) indices of all the nonzeros, followed by the floating point values of all the nonzeros. We typically use integers to represent everything but the floating point values for which we use floats or doubles. The resulting binary file can easily be several terabytes in size when the dimensions and number of nonzeros are large.

Since the matrix A is too large for us to handle directly, we split the matrix A into 20 different blocks:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_{20} \end{bmatrix}.$$

In our illustrations, we will use also the smaller submatrix A_1 of the full matrix A. The submatrix has dimensions $438{,}674 \times 3{,}637{,}248$ and is about 115 GB in uncompressed form. We can load this matrix into memory. In Fig. 5, we show the fist 2000 singular values of A_1 and A (approximated numerically via the randomized low rank SVD algorithm) with the first singular value scaled to be 1. We note that the singular values

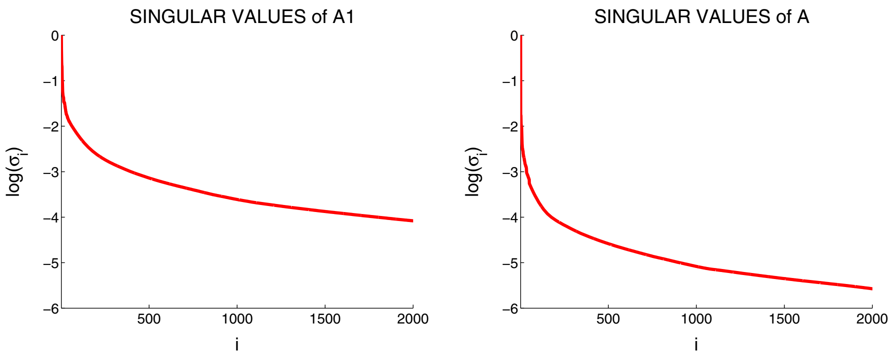

Fig. 5 First 2000 singular values of A_1 and A (numerically approximated)

of A drop off significantly faster than those of A_1 because A is a much larger matrix with significantly more linear dependence. This type of singular value behavior is common for matrices from similar applications, so as we illustrate later in this section, the low rank approximation techniques we describe here work relatively well even when the rank k is marginal compared to matrix dimensions. It's important to note again that our schemes rely mostly on operations with the $A_1^T A_1$ and $A^T A$ matrices for which the decay of the singular values is very rapid, being the square of the illustrated rate for A_1 and A.

In order get an idea of the structure and wavelet compressibility of our matrices, we take a look at a randomly chosen row of A, which represents a sensitivity kernel and its representation with different numbers of wavelet coefficients as per (4.2), using the same CDF 9–7 transform as before. In Fig. 6, we plot the sensitivity kernel near the surface of the Earth (at 135 km depth). That is, we plot part of a row of matrix, representing a certain depth layer near the surface. From the figure, we can clearly see that the kernel looks like a continuous image and is hence similar to a row of matrix $A_{(3)}$ in the previous section, which as we saw, was wavelet compressible. In the top of Fig. 6, the leftmost plot is the original kernel while the rightmost plot is the reconstructed kernel with about 10 % of the coefficients retained after transforming. We see a notable degradation in quality. However, when we keep about 25 % of the largest coefficients, we have much less noticeable reconstruction error. We clearly observe that while some details are lost as less coefficients are retained, the majority of the structure is preserved. We have performed such plots of several randomly chosen rows and we conclude that our matrix A is at least as good for wavelet compression as synthetic matrix $A_{(2)}$ (where at least a subset of the rows compressed well), but likely significantly better, with most rows being wavelet compressible. In Fig. 6, we also plot a curve of the percent error $E = 100\frac{\|r-(W^{-1}(\mathbb{T}(Wr^T)))^T\|}{\|r\|}$ versus the percent of coefficients retained by the thresholding function. By percent coefficients retained we mean the quantity $100\frac{\mathrm{nnz}(\mathbb{T}(Wr^T))}{\mathrm{nnz}(Wr^T)}$, where r is either the whole row vector or part of a row (corresponding either to all depth layers or to a certain depth near the surface) and nnz is the number of nonzeros. Notice that the error over all depths (all the entries of the kernel row) is greater than just at the particular depth layer at which it is plotted; but it is acceptable as long as we keep about 25 % or more coefficients after transforming.

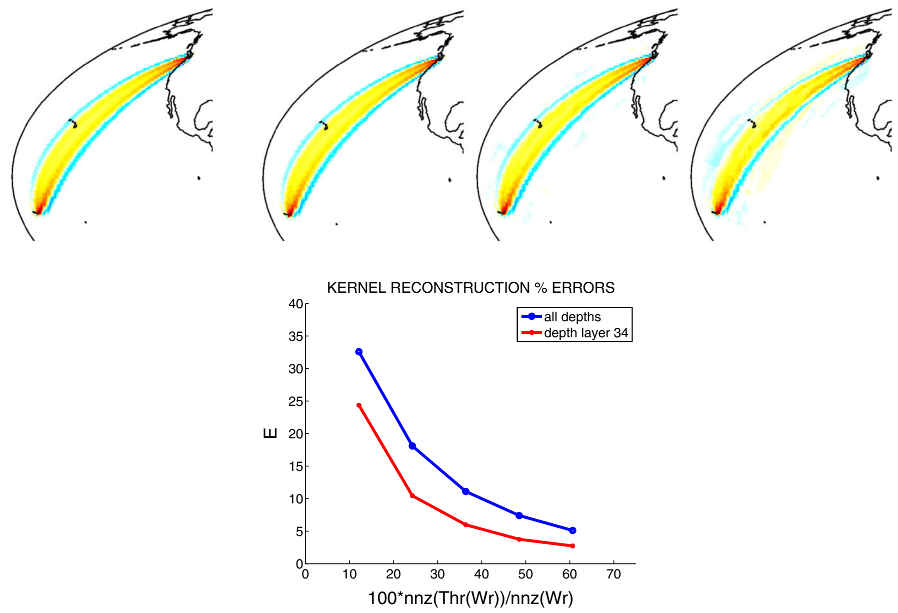

Fig. 6 Original kernel r and reconstructed compressed kernels $(W^{-1}(\mathbb{T}(Wr^T)))^T$ (at 135 km depth) with different numbers of coefficients retained after thresholding: approximately 48, 24, and 10 % coefficients, respectively. The *bottom plot* shows the percent error curve between the reconstructed and original kernel versus the number of nonzeros retained: errors for all depths and only for the displayed depth are shown

Since we find that the rows of A are in large part wavelet compressible, we will again use wavelet compression and the low rank SVD, in order to approximate matrix vector operations with the matrices A and A_1 and the solutions:

$$(A_1^T A_1 + \lambda I)\bar{x}_1 = A_1^T b \quad \text{and} \quad (A^T A + \lambda I)\bar{x}_2 \\ = A^T b \quad \text{and} \quad (A^T A + \lambda_1 I + \lambda_2 L^T L)\bar{x}_3 = A^T b$$

with L a Laplacian smoothing operator, which we build from scratch as a sparse matrix. Just as with our synthetic data examples, we first obtain the wavelet thresholded matrices M_1 and M corresponding to A_1 and A and use these smaller matrices to obtain the low rank SVD of the A_1 and A matrices, to achieve further compression. Notice also that as our data comes from a surface wave data set, the resolution of our inversions is primarily limited to a region close to the Earth's surface, a point we remind the reader of several times in this section.

6.2.1 Wavelet and SVD compression with smaller matrix A_1

We now discuss the results of some experiments with matrix A_1 which was just small enough for us to load in RAM in uncompressed form. We form the corresponding wavelet thresholded matrix $M_1 = \mathbb{T}(A_1 W^T)$ by replacing each row r of A_1 by

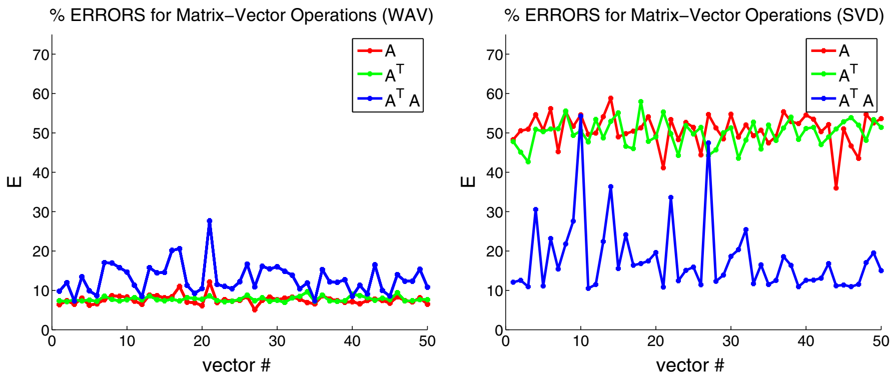

Fig. 7 Percent errors for 50 Gaussian random vectors x and y between the vectors A_1x, $A_1^T y$, $A_1^T A_1 x$ and their approximations through wavelet compressed and low rank SVD methods

$(\mathbb{T}(Wr^T))^T$. We retain one third of the largest coefficients by absolute magnitude. The full matrix A_1 is of size 115 GB while the matrix M_1 computed with our chosen threshold comes out to be 35 GB. In Fig. 7 we show the errors that result when we use the compressed matrix M_1 to approximate matrix vector operations with A_1. For 50 random Gaussian vectors x and y compatible with the dimensions of A_1 and A_1^T, we plot the percent errors between A_1x and $M_1W^{-T}x$, $A_1^T y$ and $W^{-1}M_1^T y$, and between $A_1^T A_1 x$ and $W^{-1}M_1^T M_1 W^{-T}x$. The error quantity for the first case is simply $E = 100\frac{\|A_1x-M_1W^{-T}x\|}{\|A_1x\|}$, as before in the synthetic data tests.

We use the same CDF 9–7 wavelet transform as in the synthetic tests for W, but do not build W explicitly as a matrix and cannot obtain the inverse-transpose matrix W^{-T} by transposing the inverse of W. This is because W is a very large $n \times n$ matrix and is very costly to build for large n. Hence, we instead use a routine for applying W and W^{-T} to vectors. Unlike with synthetic data where W^{-T} is exact, the implemented routine for the inverse transpose transform is approximate. We programmed the inverse transpose routine by applying the forward transform with the inverse filters but it did not exactly equal to the inverse of the transpose of W because of complicated boundary data treatment. We see that this increases the errors somewhat when approximating matrix–vector operations with A_1 and $A_1^T A_1$. We see that the error for approximating the operation $A_1^T A_1 x$ is for some vectors higher than the approximation for $A_1 x$ and $A_1^T y$. However, from the figure we see that all operations are approximated with errors below about 20 percent (which, although significant, will not give rise to large errors in regularized solutions).

Next, as we previously did with synthetic data, we go on to compute the approximate low rank SVD of $A_1 \approx U_{1_k}\Sigma_{1_k}V_{1_k}^T$ using the wavelet compressed matrix M_1 to approximate matrix–vector operations with A_1 in the randomized low rank SVD algorithm. The dimensions and sizes of the various matrices turn out as follows:

- A_1, dimensions (438,674 $\times$ 3,637,248), size is 115 GB.
- M_1, dimensions (438,674 $\times$ 3,637,248), size is 35 GB.
- U_{1_k}, Σ_{1_k}, V_{1_k}, dimensions (438,674 $\times$ 2000), (2000 $\times$ 2000), (3,637,248 $\times$ 2000), sizes are 7 GB, 30 MB, 55 GB ($\approx$62 GB total).

We show the errors that result in approximating matrix–vector operations with A_1 and A_1^T using the low rank SVD in the same Fig. 7 where we plot, for 50 randomly generated vectors x and y, percent errors between $A_1 x$ and $U_{1_k}\Sigma_{1_k}V_{1_k}^T x$, $A_1^T y$ and $V_{1_k}\Sigma_{1_k}U_{1_k}^T y$, and between $A_1^T A_1 x$ and $V_{1_k}\Sigma_{1_k}^2 V_{1_k}^T x$. The error quantity for the first case is simply $E = 100\frac{\|A_1 x - U_{1_k}\Sigma_{1_k}V_{1_k}^T x\|}{\|A_1 x\|}$, as before in the synthetic data tests. From the figure we see that for approximating the $A_1^T A_1 x$ operation with $V_{1_k}\Sigma_{1_k}^2 V_{1_k}^T x$, the errors are similar to those obtained via the wavelet thresholded $W^{-1}M_1^T M_1 W^{-T}x$ approximation, though they do jump to about 50 % for a few vectors in the set. The errors are significantly lower for the approximated $A_1^T A_1 x$ operation then for operations with A_1 or A_1^T individually. This is because the decay of singular values of $A_1^T A_1$ is much more rapid than that of A_1 and the matrix is thus well approximated with a low rank k. Notice, however, that for the low rank SVD of A_1, the total size of the SVD components (which are not sparse matrices) is greater than the size of the matrix M_1. Hence, it may not be very practical to use the low rank SVD decomposition for this smaller matrix. However, it is useful to use in this case for illustrative purposes.

We go on to obtain some approximate regularized solutions using the wavelet compressed matrix M_1 and the low rank SVD components U_{1_k}, Σ_{1_k}, V_{1_k} and compare to the full solution we get with matrix A_1. The solutions we plot in Fig. 8 are obtained by doing 250 iterations of the CG algorithm for the systems listed below.

$$\begin{aligned}
(A_1^T A_1 + \lambda I)x_1 &= A_1^T b_1 && \text{solution with full matrix } A_1\\
(W^{-1}M_1^T M_1 W^{-T} + \lambda I)x_2 &= W^{-1}M_1^T b_1 && \text{wavelet compressed solution with } M_1\\
(V_{k_1}\Sigma_{k_1}^2 V_{k_1}^T + \lambda I)x_3 &= V_{k_1}\Sigma_{k_1}U_{k_1}^T b_1 && \text{replacing all instances of } A_1 \text{ by low rank SVD}\\
(V_{k_1}\Sigma_{k_1}^2 V_{k_1}^T + 5\lambda I)x_4 &= W^{-1}M_1^T b_1 && \text{using the low rank SVD only on the left hand side.}
\end{aligned} \tag{6.1}$$

In the figure, we plot the solution at a certain depth near the surface because the data set we used in the construction of A (and hence A_1) is a surface wave data set, so there is minimal resolution far down from the surface. We mention more on this later in this section. At the depth we show, the differences between the solutions are very small. The SVD solutions do show some minor degradations. We have observed the same behavior slightly above and below the current depth: that is, for all regions where we have significant resolution with our data set. Notice that the wavelet compressed solution x_2 is very close to the full solution. With x_3 and x_4 small differences can be observed. The latter solution x_4 actually reveals somewhat more details than x_3. Note also that in Fig. 8 we plot the depth profiles for each solution, where we show a depth slice for a section of the Earth, from the surface to the core mantle boundary. As expected, nonzero data is only present at depth layers near the surface and the quality of the approximations decrease at the bottom layers. The loss of detail with the low rank SVD solutions at the lower layers is visible in these plots.

Also in Fig. 8 we show the plots of solution norm and χ^2 value versus iteration for the different solutions. The norm of the solution is the ℓ_2 norm of the iterate x^n at iteration n. The χ^2 value is calculated using the formula:

$$\chi^2 = \frac{1}{P}\sum_{k \text{ not outlier}} |r_k^n|^2,$$

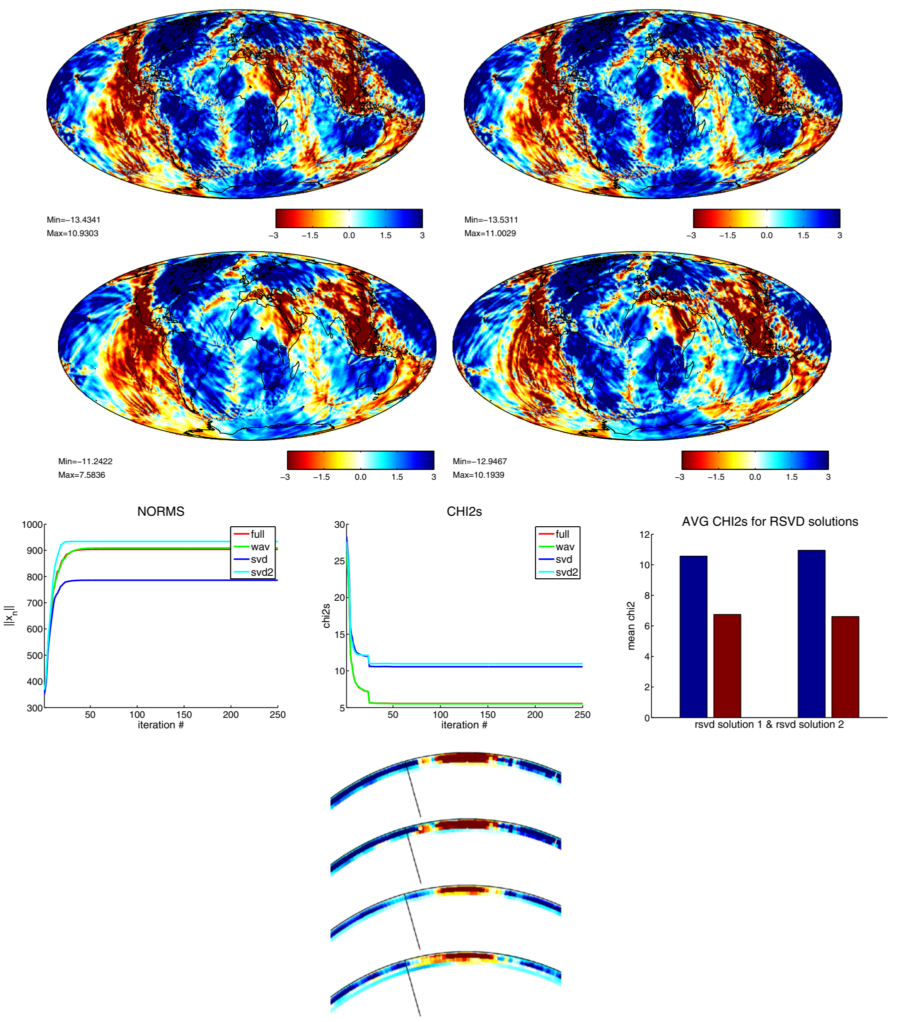

Fig. 8 *Rows 1–2* regularized solutions x_1 (full matrix—*row 1*, *left*), x_2 (wavelet), x_3 (svd 1—*row 2*, *left*), and x_4 (svd 2) from (6.1) plotted at 135 km depth. *Row 3* solution norms and χ^2 values versus iteration, *bar plot* comparing average χ^2 of the two SVD solutions computed using the low rank SVD matrix and the wavelet compressed matrix. *Row 4* depth profiles of the four solutions in a portion of the globe with variations (the *top arcs* represent the Earth's surface)

where $r^n = A_1 x^n - b$ and $P = m - m_0$ (number of rows minus number of outliers). For each datum, we estimate standard errors in the data before inversion, then scale the system to be univariant (i.e. all standard errors are equal to 1). We define outliers as entries of the vector r^n that are not within three standard errors. In the inversions we present, the outliers are identified after 5 and 25 iterations, corresponding to dips in the χ^2 that may be seen in the plots. Since our systems are univariant, we would like for the χ^2 of the converged solution to be close to one. However, this is not possible for this data set without including extra correction terms for spatial uncertainty in the earthquake coordinates and instrument error in the data. Hence the χ^2 values are quite a bit higher. In the figure, we can see that the curves for the full and wavelet thresholded case are very close to each other; the first SVD solution has a lower norm and the second a slightly higher solution norm at the chosen value of λ.

For the χ^2 calculation, we calculate the product $A_1 x^n$ using the full matrix A_1 in the solution x_1, using the approximation $M_1 W^{-T} x^n$ in the solution x_2, and using the approximation $U_{1_k} \Sigma_{1_k} V_{1_k}^T x^n$ in the solutions x_3 and x_4. Notice that since the operation $A_1 x^n$ is not as well approximated as the operation $A_1^T A_1 x^n$, we have a noticeable difference in χ^2 values between solutions x_1 and x_2 and between x_3 and x_4. For the latter two solutions, the calculated χ^2 value comes out higher than it really is. To illustrate this fact, we include in Fig. 8 a bar plot which shows the the mean χ^2 value after 50 iterations from the two SVD solutions x_3 and x_4 computed using the low rank matrix $U_{1_k} \Sigma_{1_k} V_{1_k}^T$ and using the full matrix A_1. The same solutions have correspondingly lower χ^2 values when the residual $r^n = A_1 x^n - b_1$ is approximated via $A_1 x^n - b_1$ instead of $U_{1_k} \Sigma_{1_k} V_{1_k}^T x^n - b_1$. Thus, while the solutions themselves are approximated well with the SVD approximations, quantities such as χ^2 which involve calculations with A_1 instead of $A_1^T A_1$ can be far less accurate when computed with the low rank SVD matrices. Given the results with the matrix A_1, we summarize a few key points which we observe.

- In the case of matrix A_1 which is not so large, wavelet thresholding makes the most sense, as the low rank SVD does not provide compression, unless the $k \times k$ methods are used. This is because the low rank SVD matrices are dense while the original matrix is sparse.
- Approximate solutions with both wavelet thresholding and the low rank SVD are quite accurate compared to those with the full matrix.
- In matrix vector operations, the error in the approximation to operations with $A_1^T A_1$ is significantly less than for the approximations to operations with A_1 and A_1^T. Hence, quantities such as χ^2 are not accurately computed if the low rank SVD matrix is used to compute the residual; instead one should use the wavelet compressed or full matrix (for one computation) to accurately estimate the χ^2 value of the solution vector.
- When computed with A_1 or M_1, the χ^2 values for the approximate solutions are very similar to that of the full solution.
- The difference between the full and approximate solutions becomes significant at lower depths, where the data set resolution is poor.

6.2.2 Wavelet and SVD compression with matrix A

We now describe some results of wavelet and low rank SVD compression for our very large matrix A. Due to the size of A, even after wavelet compression, the resulting M is too big to load into RAM all at once on a single machine. For this reason, we do not compute the wavelet thresholded M in one shot. Instead we operate on blocks of A at a time and construct the block based:

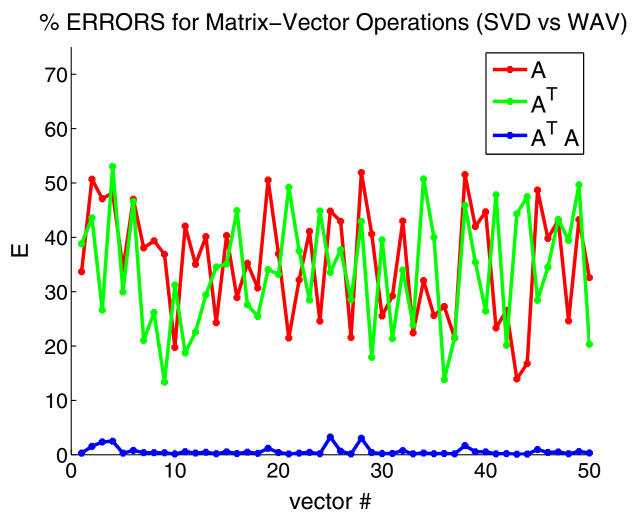

Fig. 9 Percent errors in matrix–vector operations for 50 Gaussian random vectors with A, A^T, and $A^T A$ approximated via the wavelet compressed matrix M and compared to results obtained with the low rank SVD (obtained via M)

$$M = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{20} \end{bmatrix} = \begin{bmatrix} \mathbb{T}(A_1 W^T) \\ \mathbb{T}(A_2 W^T) \\ \vdots \\ \mathbb{T}(A_{20} W^T) \end{bmatrix}$$

This way, operations with A can be approximated using relations (4.5) and the components of M can be stored in parallel over several different machines.

We now state the sizes and dimensions of the matrices involved:

- A, dimensions (2,968,933 $\times$ 3,637,248), size is 3.2 TB (approximate, never computed).
- M, dimensions (2,968,933 $\times$ 3,637,248), size is 1 TB.
- U_k, Σ_k, V_k, dimensions (2,968,933 $\times$ 2000), (2000 $\times$ 2000), (3,637,248 $\times$ 2000), sizes are 45 GB, 30 MB, 55 GB ($\approx$100 GB total).

Notice that in this case, for the much larger matrix A, the low rank SVD provides for very substantial memory savings.

Since we cannot use A directly, we can only compare results with the wavelet compressed matrix M to results obtained with the low rank SVD $A_k = U_k \Sigma_k V_k^T$. As before, we have first formed M and then used M in the randomized SVD scheme to form the approximate low rank SVD of A. We again used $k = 2000$ (a very small number relative to the dimensions of A). In Fig. 9, we plot the percent errors for matrix vector operations done with the computed low rank SVD compared to those approximated via the wavelet thresholded matrix M. We plot the percent errors for 50 random Gaussian vectors x and y compatible with the dimensions of A and A^T: that is, between $MW^{-T}x$ (approximating Ax) and $U_k \Sigma_k V_k^T x$, $W^{-1}M^T y$ (approximating $A^T y$) and $V_k \Sigma_k U_k^T y$ and between $W^{-1}M^T M W^{-T} x$ (approximating $A^T A x$) and

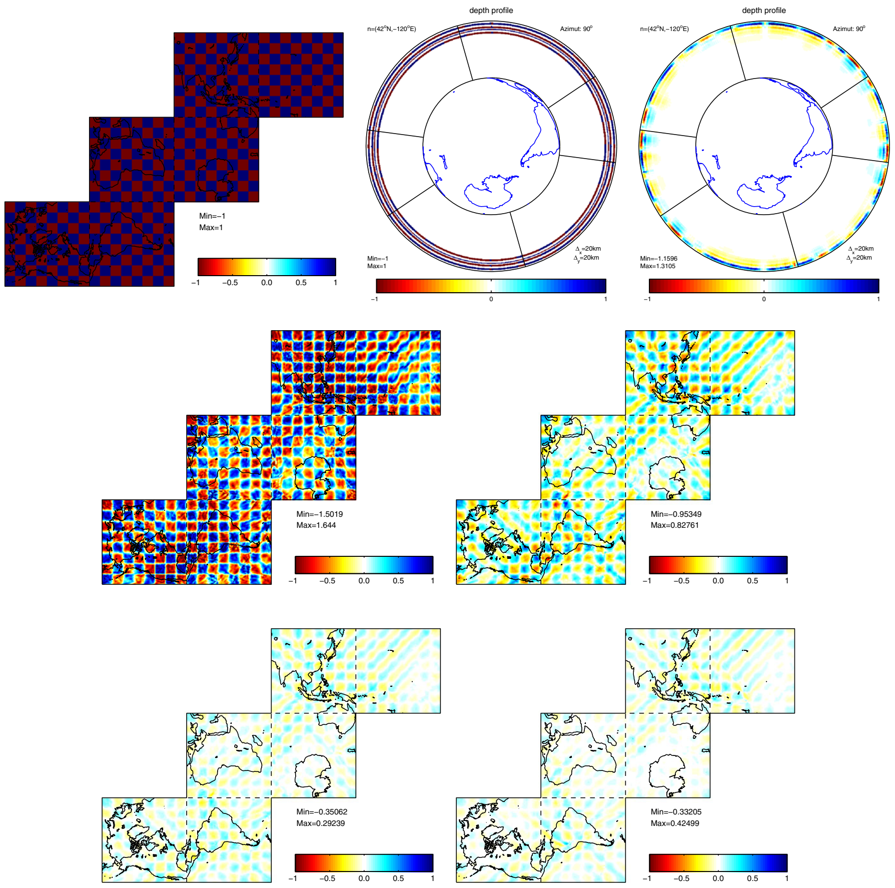

Fig. 10 Checkerboard model and reconstructions at different depths. *Row 1* synthetic x_{chk} model and it's depth profile from the surface to the core mantle boundary followed by the depth profile of the reconstructed solution x_{chkrec}. At adjacent layers, checkerboards differ only by a sign change. *Row 2* reconstructed layers 34 and 32 (135 and 316 km depth). *Row 3* reconstructed layers 30 and 28 (428 and 586 km depth)

$V_k \Sigma_k^2 V_k^T x$. The error quantity for the first case is simply $E = 100\frac{\|MW^{-T}x - U_k \Sigma_k V_k^T x\|}{\|MW^{-T}x\|}$. The plots again indicate that the operation $A^T Ax$ is likely to be well approximated even with a low rank k we choose. In this case, for the large A, the singular values of $A^T A$ decay very rapidly, with the square of the decay rate observed in Fig. 5.

As previously mentioned, the matrices we use come from a surface wave data set (van Heijst and Woodhouse 1999), such that only the top few depth layers near the surface carry nonzero information and even the bottom of these layers can already offer limited resolution. Thus the quality of approximations can vary somewhat for different depth layers. In order for the reader to have an idea of the data set we use, we present some checkerboard reconstructions using the matrix A and a synthetically constructed checkerboard model x_{chk}. We define x_{chk} to be a checkerboard grid, over the top few layers (near the surface). The result is plotted in Fig. 10 using the depth profile (a cross-section plot showing the model representation over all depth layers) and corresponding cubed-sphere representations at certain depths (we plot at each depth layer shown the projection onto the six cube faces). Then we form $b = Ax_{chk}$ and solve the regularized system $(A^T A + \lambda I)x_{chkrec} = A^T b$ with $\lambda = 5$. We plot the solution x in Fig. 10 using the same formats. We use the wavelet transformed and thresholded matrix $M = \mathbb{T}(AW^T)$ to approximate the matrix vector operations with A. The comparison between x_{chk} and the corresponding reconstruction x_{chkrec} gives us a summary of what the data set can pick up. In particular, we see from Fig. 10 that the resolution is limited to layers near the surface and gets worse with increasing depth, as expected. Also and perhaps more important is that the checkers used are about the size of what we we can successfully resolve. We have tried using smaller checkers which did not lead to good reconstructions, even for depth layers near the surface.

The checkerboard test shows the clear limitation of the matrix A: we are unable to resolve features at all depths, nor are we able to resolve particularly small features. Hence, we expect that we can safely use relatively high compression ratio approximation methods we have discussed (using aggressive thresholding with wavelet based approximation and small k relative to matrix dimension in the SVD based schemes). Even though the solutions which result from these methods may not resolve some fine scale features in comparison with using the full (or even wavelet thresholded) matrix, it is important to keep in mind that these fine scale features which appears in the more detailed solutions may not be realistically explainable by the data we have available. This is true in many applications similar to ours.

We will consider the following linear systems for approximating the regularized solution to $Ax = b$:

$$\begin{aligned}
(W^{-1}M^T MW^{-T} + \lambda I)x_5 &= W^{-1}M^T b \quad \text{wavelet compressed solution for } A\\
(V_k\Sigma_k^2 V_k^T + \lambda I)x_6 &= V_k\Sigma_k U_k^T b \quad \text{replacing all instances of } A \text{ by low rank SVD}\\
(V_k\Sigma_k^2 V_k^T + 10\lambda I)x_7 &= W^{-1}M^T b \quad \text{using the low rank SVD only on the left hand side}
\end{aligned}$$

with $\lambda = 1$. We also show the following solutions corresponding to the system with Laplacian smoothing included:

$$\begin{aligned}
(W^{-1}M^T MW^{-T} + \lambda_1 I + \lambda_2 L^T L)x_8 &= W^{-1}M^T b \quad \text{wavelet compressed with smoothing}\\
(V_k\Sigma_k^2 V_k^T + \lambda_1 I + \lambda_2 L^T L)x_9 &= V_k\Sigma_k U_k^T b \quad \text{SVD 1 with smoothing}\\
(V_k\Sigma_k^2 V_k^T + 10\lambda_1 I + \lambda_2 L^T L)x_{10} &= W^{-1}M^T b \quad \text{SVD 2 with smoothing.}
\end{aligned}$$

The results for a depth layer close to the surface are given in Fig. 11. Again, we find that the results for depth layers around the given depth are quite similar to what we present. We can readily notice the effect of the smoothing operator L on the solutions. Notice that the wavelet compressed solution without smoothing offers a great deal of detail. However, based on our checkerboard experiments, it's unlikely that the smaller scale features we find in this detailed solution are real, since they are generally smaller than the checkers we used in our resolution test. In the figure, we also plot the same plots as for the smaller matrix A_1, including plots of the solution norms, χ^2 values, and of the depth profiles of the solutions x_8, x_9, x_{10} (with Laplacian smoothing). We find similar behavior in the 3 solutions without the Laplacian. As before, we plot a bar chart showing the χ^2 of the SVD based solutions using the SVD and wavelet

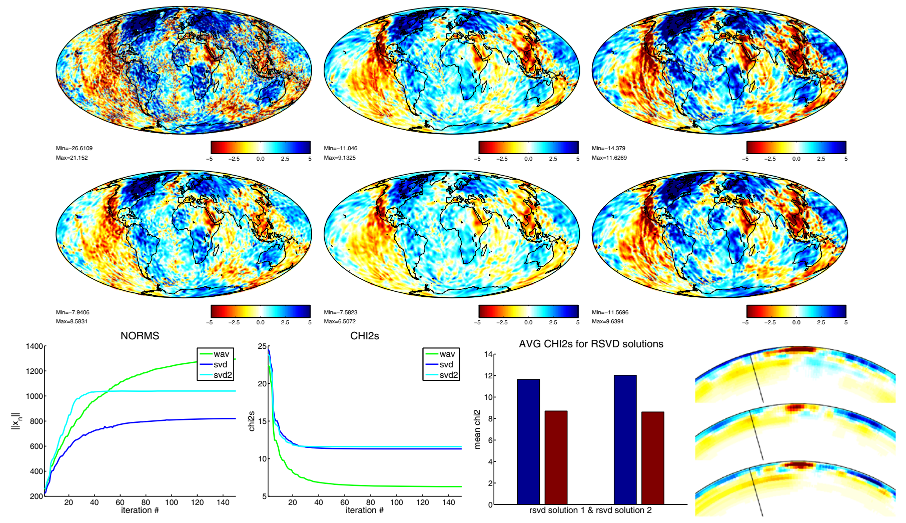

Fig. 11 Plots for regularized solutions x_5, x_6, x_7 (*row 1*) and x_8, x_9, x_{10} (*row 2*). *First* and *second row* solutions plotted at 135 km depth. *Third row* norms of solution and χ^2 value at each iteration, *bar plot* of average χ^2 of the two SVD solutions computed with the low rank SVD matrix and the wavelet compressed matrix, depth profiles of solutions x_8, x_9, and x_{10} in a portion of the globe with variations (the *top arcs* represent the Earth's surface)

compressed matrices. We see similar behavior in the sense that if χ^2 is computed with matrix M, it is close to that of the wavelet compressed solution. The depth profile plots in Fig. 11 show significant differences between the wavelet compressed and low rank SVD solutions at lower depths, although the resolution there is likely very low.

Given the results with the big matrix A, we summarize a few key points which we observe.

- Both the wavelet thresholded and the low rank SVD approach allow us to use much smaller matrices and still resolve the main solution features (in the case of A, the low rank SVD components are collectively less than 30 times the size of the full matrix and offer superior compression gains).
- For a matrix of this size, block matrix techniques we have discussed are likely necessary for practical implementation, so that different parts of the matrices used can be stored on different machines. Blocking can be applied both to wavelet compression via (4.5) or to the low rank SVD schemes via e.g. (5.11).
- In matrix vector operations, the error in the approximation to operations with $A^T A$ is significantly less than for the approximations to operations with A and A^T. This again has implications for the χ^2 calculation as previously discussed.
- The solutions with the low rank SVD do show loss of detail when compared to the wavelet thresholding solution. There is significantly less loss of detail when Laplacian smoothing is used, since the smaller scale features are smoothed out in that case.
- A checkerboard test is a good way to measure matrix resolution. The smallest clearly resolved checker size corresponds roughly to the scale of properly resolved

features in the solution. If the resolution is poor, Laplacian smoothing should be used to avoid presenting false fine scale details. In this case, the low rank SVD solutions can offer a good approximation with the use of much smaller matrices.

7 Conclusions

We have presented the use of wavelet compression and low rank SVD techniques for obtaining approximate solutions to regularization problems. We illustrate the application of these techniques to ℓ_2 regularization for synthetic data and for a large scale inverse problem from seismic tomography, where we show the pros and cons of these approximation methods in a practical setting. We have also presented some mathematical analysis for the various SVD based schemes we have considered, showing interesting equivalence between different schemes with different memory requirements. The techniques we present are also well applicable to other types of optimization problems. In fact, the methods presented here can be of use to any application where matrix–vector operations with large matrices are required, especially if the matrices are not well conditioned and have nonlinear decay of singular values.

The wavelet compressed approach is found to be very accurate and gives close reconstructions to the true solution, assuming the data are wavelet compressible. Based on our experiments, applications utilizing similar data and wavelet transform can benefit from a compression ratio of at least 3 times, with minimal accuracy loss. In our examples, we used a simple one dimensional transform for each row. Recognizing the rows as multi-dimensional images and transforming them via a multi-dimensional transform would likely give even greater compression.

For large matrices, the compression with wavelets alone may not be sufficient. The low rank SVD approach can give significantly better compression ratios (>10) and resolve the main solution features. The low rank SVD can be obtained through an efficient randomized algorithm using operations with the smaller wavelet compressed matrix instead of the full matrix, so that the two compression techniques we present can be utilized together. The approaches we discuss lead to the use of $k \times n$ or $k \times k$ matrices (which can also be split in several smaller blocks), in place of the original $m \times n$ matrix, which can result in very substantial compression ratios.

For both wavelet compressed and low rank SVD based methods, the accuracy and compression ratio are inversely proportional and controlled by the user. In the case of wavelet compression, the time it takes to form the compressed matrix is nearly independent of the threshold used. However, for the computation of the low rank SVD, the work involved substantially grows as the rank k increases. Often, a checkerboard style test can be performed to see the resolution a data set is capable of. In large problems, the resolution possible with a given matrix is often limited. The approximation techniques we propose can often be well justified physically, as the fine scale details they may remove or smooth out may not be realistically resolved by the data set.

Acknowledgments The authors would like to thank the referees for very clear and helpful comments which resulted in significant improvements. We also like to thank Ignace Loris, Gunnar Martinsson and Frederik Simons for very helpful discussion. Support from the ERC (Advanced Grant 226837), the Defense Advanced Projects Research Agency (Contract N66001-13-1-4050) and the National Science Foundation (Contracts 1320652 and 0748488) is greatly appreciated.

References

Akansu, A.N., Haddad, R.A.: Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets. Academic Press Inc, Orlando (1992)

Calvetti, D., Morigi, S., Reichel, L., Sgallari, F.: Tikhonov regularization and the *L*-curve for large discrete ill-posed problems. J. Comput. Appl. Math. **123** (1–2), 423–446 (2000)

Chárlety, J., Voronin, S., Nolet, G., Loris, I., Simons, F.J., Sigloch, K., Daubechies, I.C.: Global seismic tomography with sparsity constraints: comparison with smoothing and damping regularization. J. Geophys. Res. Solid Earth (2013)

Cohen, A., Daubechies, I.C., Feauveau, J.-C.: Biorthogonal bases of compactly supported wavelets. Commun. Pure Appl. Math. **45**(5), 485–560 (1992)

Daubechies, I.C.: Orthonormal bases of compactly supported wavelets. Commun. Pure Appl. Math. **41**, 909–996 (1988)

Daubechies, I.C., Defrise, M., De Mol, C.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Commun. Pure Appl. Math. **57**(11), 1413–1457 (2004)

Debayle, E., Sambridge, M.: Inversion of massive surface wave data sets: model construction and resolution assessment. J. Geophys. Res. **109**, B02316 (2004)

Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev. **53**(2), 217–288 (2011)

Härdle, W., Kerkyacharian, G., Picard, D., Tsybokov, A.: Wavelets, approximation, and statistical applications. Lecture Notes in Statistics, vol. 129. Springer, New York (1998)

Lampe, J., Reichel, L., Voss, H.: Large-scale Tikhonov regularization via reduction by orthogonal projection. Linear Algebra Appl. **436**(8), 2845–2865 (2012). **(Special Issue dedicated to Danny Sorensen's 65th birthday)**

Markovsky, I.: Low rank approximation: algorithms, implementation, applications. In: Communications and Control Engineering. Springer, New York (2012)

Marquering, H., Nolet, G., Dahlen, F.A.: Three-dimensional waveform sensitivity kernels. Geophys. J. Int. **132**(3), 521–534 (1998)

Meyer, Y.: Wavelets: Algorithms and Applications. Society for Industrial and Applied Mathematics, Philadelphia (1993). **(Translated and revised by Robert D. Ryan)**

Nolet, G.: A Breviary of Seismic Tomography. Cambridge Univ. Press, Cambridge (2008)

Paige, C.C., Saunders, M.A.: LSQR: an algorithm for sparse linear equations and sparse least squares. ACM Trans. Math. Softw. **8**(1), 43–71 (1982)

Ronchi, C., Iacono, R., Paolucci, P.S.: The cubed sphere: a new method for the solution of partial differential equations in spherical geometry. J. Comput. Phys. **124**(1), 93–114 (1996)

Simons, F.J., Loris, I., Nolet, G., Daubechies, I.C., Voronin, S., Judd, J.S., Vetter, P.A., Chárlety, J., Vonesch, C.: Solving or resolving global tomographic models with spherical wavelets, and the scale and sparsity of seismic heterogeneity. Geophys. J. Int. **187**(2), 969–988 (2011)

Sweldens, W.: The lifting scheme: a new philosophy in biorthogonal wavelet constructions. In: Laine, A.F., Unser, M.A., Wickerhauser, M.V. (eds.) Wavelet applications in signal and image processing III. Proceedings of SPIE, vol. 2569, pp. 68–79 (1995)

Tikhonov, A.N.: Solution of incorrectly formulated problems and the regularization method. Sov. Math. Dokl. (1963)

Trefethen, L.N., Bau, D.: Numerical Linear Algebra. SIAM Philadelphia (1997)

van Heijst, H.-J., Woodhouse, J.H.: Global high-resolution phase velocity distributions of overtone and fundamental mode surface waves determined by mode branch stripping. Geophys. J. Int. **137**(3), 601–620 (1999)

Voronin, S., Martinsson, P.-G.: RSVDPACK: subroutines for computing partial singular value decompositions via randomized sampling on single core, multi core, and GPU architectures (2015). **ArXiv e-prints**

Wang, S., Zhang, Z.: Improving CUR matrix decomposition and the Nystrom approximation via adaptive sampling. J. Mach. Learn. Res. **14**(1), 2729–2769 (2013)

Woodbury, M.A.: Inverting modified matrices. In: Statistical Research Group, Memo. Rep. No. 42. Princeton University, Princeton (1950)