Efficient computation of matrix-vector products with full observation weighting matrices in data assimilation

Hu, G. ORCID: https://orcid.org/0000-0003-4305-3658 and Dance, S. ORCID: https://orcid.org/0000-0003-1690-3338
(2021) Efficient computation of matrix-vector products with full observation weighting matrices in data assimilation. Quarterly Journal of the Royal Meteorological Society, 147 (741). pp. 4101-4121. ISSN 1477-870X doi: https://doi.org/10.1002/qj.4170 Available at https://centaur.reading.ac.uk/100366/

Received: 3 March 2021 | Revised: 12 July 2021 | Accepted: 18 September 2021

DOI: 10.1002/qj.4170

Quarterly Journal of the Royal Meteorological Society RMetS

RESEARCH ARTICLE

Efficient computation of matrix–vector products with full observation weighting matrices in data assimilation

Guannan Hu[1] | **Sarah L. Dance**[1,2]

[1]Department of Meteorology, University of Reading, Reading, UK

[2]Department of Mathematics and Statistics, University of Reading, Reading, UK

Correspondence
G. Hu, Department of Meteorology, University of Reading, Reading, UK
Email: guannan.hu@reading.ac.uk

Funding information
Engineering and Physical Sciences Research Council, Grant/Award Number: UK EPSRC DARE project (EP/P002331/1)

Abstract

Recent studies have demonstrated improved skill in numerical weather prediction via the use of spatially correlated observation error covariance information in data assimilation systems. In this case, the observation weighting matrices (inverse error covariance matrices) used in the assimilation may be full matrices rather than diagonal. Thus, the computation of matrix–vector products in the variational minimization problem may be very time-consuming, particularly if the parallel computation of the matrix–vector product requires a high degree of communication between processing elements. Hence, we introduce a well-known numerical approximation method, called the fast multipole method (FMM), to speed up the matrix–vector multiplications in data assimilation. We explore a particular type of FMM that uses a singular value decomposition (SVD-FMM) and adjust it to suit our new application in data assimilation. By approximating a large part of the computation of the matrix–vector product, the SVD-FMM technique greatly reduces the computational complexity compared with the standard approach. We develop a novel possible parallelization scheme of the SVD-FMM for our application, which can reduce the communication costs. We investigate the accuracy of the SVD-FMM technique in several numerical experiments: we first assess the accuracy using covariance matrices that are created using different correlation functions and lengthscales, then investigate the impact of reconditioning the covariance matrices on the accuracy, and finally examine the feasibility of the technique in the presence of missing observations. We also provide theoretical explanations for some numerical results. Our results show that the SVD-FMM technique can compute the matrix–vector product with good accuracy in a wide variety of circumstances and, hence, has potential as an efficient technique for assimilation of a large volume of observational data within a short time interval.

KEYWORDS

data assimilation, fast multipole method, matrix–vector multiplication, observation error covariance matrix, parallelization

1 | INTRODUCTION

In variational data assimilation (e.g., Lorenc *et al.*, 2000; Rawlins *et al.*, 2007), a nonlinear least-squares problem is solved, where observations and model forecasts are blended, taking account of their uncertainties. The minimization procedure involves time-consuming computations of large matrix–vector products. In this paper, we focus on the matrix–vector products involving the observations. These take the form $\mathbf{R}^{-1}\mathbf{d}$, where $\mathbf{R}^{-1} \in \mathbb{R}^{m\times m}$ is the inverse of the observation error covariance matrix and $\mathbf{d} \in \mathbb{R}^m$ is the observation-minus-model departure vector (see Section 2).

In some practical numerical weather prediction applications, observation errors are assumed to be uncorrelated, resulting in the matrix $\mathbf{R}$ being diagonal. This reduces the number of operations required to compute matrix–vector products in the minimization, and is a pragmatic strategy when the characteristics of the observation uncertainty are not well understood (e.g., Liu and Rabier, 2003). However, a number of idealized and operational studies have shown that there are significant benefits to treating full observation error covariance matrices in terms of analysis information content, analysis accuracy, and forecast skill, even when knowledge of the observation error correlations is only approximate (e.g., Healy and White, 2005; Stewart *et al.*, 2008; 2013; Weston *et al.*, 2014; Simonin *et al.*, 2019; Bédard and Buehner, 2020). In particular, implementation of spatial observation error correlations modifies the lengthscales of the observation increments computed by the assimilation (e.g., Rainwater *et al.*, 2015; Fowler *et al.*, 2018; Simonin *et al.*, 2019), which may be especially important for multiscale systems such as convection-permitting numerical weather prediction and reanalyses (e.g., Dance *et al.*, 2019; Hu and Franzke, 2020) or coupled ocean–atmosphere systems (e.g., Hu and Franzke, 2017). Furthermore, practical methods have been developed to estimate the observation uncertainty characteristics for a range of observation types (see the reviews by Janjić *et al.* (2018) and Tandeo *et al.* (2020)). For example, Doppler radar radial winds, geostationary satellite data, and atmospheric motion vectors (AMVs) have been shown to exhibit strong spatial error correlations (Waller *et al.*, 2016a; 2016b; 2019; Cordoba *et al.*, 2017; Honda *et al.*, 2018). We note that, in practical applications, the matrix $\mathbf{R}$ is typically treated as block diagonal with one block per observation type (for a given time). The observation errors between different types of observations are assumed to be uncorrelated.

To give an idea of the expected size of the spatially correlated observation error covariance matrices, we consider geostationary satellite observations from the Spinning Enhanced Visible and Infrared Imager (SEVIRI) instrument (Schmetz *et al.*, 2002; Waller *et al.*, 2016b; Michel, 2018). SEVIRI measures top-of-atmosphere radiances in 12 channels, with a 15-min repeat cycle and at approximately 3 km spatial resolution (excluding the high-resolution visible (HRV) channel; Schmetz *et al.*, 2002; Waller *et al.*, 2016b). Each image consists of around 10^7 pixels for the thermal infrared and solar channels (Schmetz *et al.*, 2002). Hence, there are a vast number of SEVIRI observations, even for a regional domain. For instance, the number of SEVIRI observations within the domain covered by the operational AROME model from Météo-France is around 4×10^5 (Seity *et al.*, 2011). In operational assimilation applications, to avoid treating spatially correlated observation errors, spatial thinning of SEVIRI observations is typically applied and this reduces the number of observations by several orders of magnitude, but also reduces their benefits in improving forecast skill (Waller *et al.*, 2016b; Michel, 2018). The next generation of meteorological satellites will produce observations with even higher spatial resolutions (WMO, 2017, Table 1). For example, the Flexible Combined Imager (FCI) on the Meteosat Third Generation (MTG) satellite and the Advanced Baseline Imager (ABI) on the Geostationary Operational Environmental Satellite-R (Schmit *et al.*, 2017) both take measurements at approximately 0.5–2 km spatial resolution, and the Advanced Geosynchronous Radiation Imager (AGRI) on the Fengyu-4 geostationary meteorological satellite produces observations at a resolution from 1–4 km (Yang *et al.*, 2017).

Accounting for spatially correlated error statistics in the data assimilation algorithm has potential to increase the computational cost for several reasons: (a) the computation of large matrix–vector products using parallel computing techniques, (b) the need to compute products using the inverse covariance matrix which may be different in each assimilation cycle, and (c) changes to the convergence behavior of the minimization procedure. We now review these issues in more detail.

The first issue is that the parallel computation of a large matrix–vector product with observation data distributed across multiple processing elements (PEs) may become expensive due to excessive communications between PEs and load imbalance overheads (Deng, 2012). Different partitions of the matrix will lead to different parallelization schemes, which will be described in Section 2. In general, to reduce the time spent on communication operations, it is necessary to reduce the number and/or size of messages transferred. In the context of data assimilation, much of this communication can be avoided if all the observations with mutually correlated errors are assigned to one PE, as in Simonin *et al.* (2019). However, this relies on the data volume of observations with mutually correlated errors being small enough for the product to be calculated on one

node. It is an open question how to implement computations for larger data volumes with distributed data. While our paper is focused on parallel implementations for variational assimilation, studies such as Anderson (2003) and Nino-Ruiz *et al.* (2019) address parallel implementations for ensemble methods.

The second issue is that there is a need to use the inverse observation error covariance matrix ($\mathbf{R}^{-1}$) in the computations. The most commonly used observation uncertainty diagnosis techniques provide an estimate of the observation error covariance matrix itself rather than its inverse (Desroziers *et al.*, 2005). Since the observation distribution changes each assimilation cycle (due to quality control, etc.) there is a different inverse matrix each cycle. Simonin *et al.* (2019) deal with this by using a Cholesky decomposition method (Golub and Van Loan, 1996), which avoids the need to compute the inverse covariance matrix directly and is applicable to any form of error covariance matrix. Guillet *et al.* (2019) model the inverse of a spatially correlated observation error covariance matrix directly using a diffusion operator and fast unstructured meshing techniques. However, this method only deals with spatial error correlations and it is unclear whether this approach is also suitable when spatial error correlations and interchannel error correlations are combined.

The third issue is that the use of correlated observation error statistics changes the convergence behavior of the minimization procedure. Indeed, the pre-operational experiments of Weston *et al.* (2014) showed problems with the minimization that were improved by reconditioning the observation error covariance matrix. The convergence of the minimization procedure has been further studied by Tabeart *et al.* (2018; 2020a; 2021), who found that the minimum eigenvalue of the observation error covariance matrix is a key parameter governing the speed of convergence. Hence, the use of reconditioning methods is common in operational applications with correlated observation error statistics (e.g., Bormann *et al.*, 2016; Campbell *et al.*, 2017; Tabeart *et al.*, 2020a; 2020b).

The aim of this paper is to carry out an investigation of a method to accelerate parallel matrix–vector products with distributed observational data, using a novel application of a well-known numerical approximation algorithm, the fast multipole method (FMM; Rokhlin, 1985; Greengard and Rokhlin, 1997). We explore a particular type of FMM that uses a singular value decomposition (SVD) (Gimbutas and Rokhlin, 2003). We call this method the SVD-FMM. The key idea of the SVD-FMM is to split up the computation of the matrix–vector product into separate, near-field and far-field calculations. The near-field calculations are done by standard matrix–vector multiplication of a local submatrix and subvector. The far-field calculations are carried out using the singular values and singular vectors of the submatrices of $\mathbf{R}^{-1}$. The submatrices and subvectors are determined by selecting specific rows and columns of $\mathbf{R}^{-1}$ corresponding to a partition of observations in the domain. The number of singular vectors employed in the calculation is chosen by the user. We will show that usually only a small number of singular vectors is needed to obtain a good accuracy (Section 5). The SVD-FMM technique reduces the number of floating-point operations required compared with the standard approach for matrix–vector multiplication. Additionally, we develop a novel possible parallel algorithm for the SVD-FMM for our application in data assimilation, which can greatly reduce the costs of communication between PEs. The SVD-FMM allows us to assimilate a large volume of observational data in a short time interval, and has the potential to be used in practical applications.

In this initial investigation, our numerical results focus on evaluating the accuracy of the SVD-FMM. Thus our experiments are carried out for an idealized problem, using serial rather than parallel computing. Nevertheless, we do compare the efficiency of the proposed parallelization scheme with different parallel formulations of standard matrix–vector multiplications in terms of communication costs. In our current implementation, the method needs to be applied after we obtain a representation of the matrix $\mathbf{R}^{-1}$. In order to have a clear focus solely on computing matrix–vector products, we do not address the computation of the inverse observation error covariance matrix. Instead, we assume that the inverse observation error covariance is already known.

In our experiments we show that the SVD-FMM can work well with the inverses of a variety of covariance matrices. In particular, we apply the SVD-FMM to the inverses of covariance matrices created using different correlation functions and lengthscales. We also use the reconditioning methods of Tabeart *et al.* (2020b) together with the SVD-FMM to gain insight into how the accuracy of the SVD-FMM should change with different levels of reconditioning. In practice, the observation distribution varies each assimilation cycle due to factors such as quality control and the removal of cloudy satellite radiance observations. Therefore, we further carry out some experiments to demonstrate that the SVD-FMM is feasible even if there are some missing observations.

The rest of this paper is organized as follows: We provide the parallel formulations of standard matrix–vector multiplication in Section 2. In Section 3 we present our novel algorithm for the SVD-FMM and compare the complexity and the communication costs between the SVD-FMM and standard, parallel matrix–vector multiplication. In Section 4 we explain our experimental design for

our idealized experiments. In particular, we describe the generation of observations and observation error covariance matrices as well as the reconditioning methods. In Section 5 we show the results of our numerical experiments with varying correlation functions, lengthscales, reconditioning methods, and condition numbers. We also show how the results change with missing observations. Finally we give a summary in Section 6 and conclude that our proposed algorithm has potential for use in operational data assimilation for fast computation of large matrix–vector products.

2 | PARALLELIZATION OF STANDARD MATRIX–VECTOR MULTIPLICATION

In this section we describe three distinct standard parallel formulations for computing large matrix–vector products (see Grama *et al.*, 2003, section 8.1; Deng, 2012, section 6.2.1). We also discuss how to exploit the symmetric structure of $\mathbf{R}^{-1}$ for parallelization (Geus and Röllin, 2001). These matrix–vector products arise in the solution of the variational minimization problem in data assimilation in the form

$$\mathbf{q} = \mathbf{A}\mathbf{d}, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{m\times m}$ denotes the inverse of the observation error covariance matrix, $\mathbf{d} \in \mathbb{R}^m$ denotes the observation-minus-model departure vector, and $\mathbf{q} \in \mathbb{R}^m$ denotes the result of the multiplication. For the purposes of this description, we assume that the matrix $\mathbf{A}$ and vector $\mathbf{d}$ are already known.

To see how matrix–vector products in the form of Equation (1) arise in data assimilation, we consider the observation penalty term of the variational data assimilation cost function. This is given by multiplying the inverse observation error covariance matrix by the observation-minus-model differences

$$J_o = \frac{1}{2}\{\mathbf{y} - H(\mathbf{x})\}^{\mathrm{T}}\mathbf{R}^{-1}\{\mathbf{y} - H(\mathbf{x})\}, \tag{2}$$

where $\mathbf{y} \in \mathbb{R}^m$ denotes the observation vector, $\mathbf{x} \in \mathbb{R}^n$ denotes model state, and H denotes the observation operator that maps model state to the observation ($H : \mathbb{R}^n \to \mathbb{R}^m$; e.g., Lorenc *et al.*, 2000; Rawlins *et al.*, 2007; Simonin *et al.*, 2019; Nichols, 2010). To solve the variational minimization problem, the gradient of Equation (2) is needed (e.g., Simonin *et al.*, 2019)

$$\frac{\partial J_o}{\partial \mathbf{x}} = \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\{\mathbf{y} - H(\mathbf{x})\}. \tag{3}$$

Thus matrix–vector products of the form of Equation (1) arise in the computation of both the cost function and its gradient, with $\mathbf{A} = \mathbf{R}^{-1}$ and $\mathbf{d} = \mathbf{y} - H(\mathbf{x})$.

The parallelization schemes for computing Equation (1) start with the distribution of observations over PEs. We consider a simple-domain decomposition, in which the observations are distributed over a number of PEs according to their geophysical locations. This will result in a split of the components of matrix $\mathbf{A}$ and vector $\mathbf{d}$ across PEs. We will introduce four different partitions of the matrix (see Figure 1). Each of them will lead to a unique parallelization scheme. The communication costs of each scheme depend on various parameters, including (a) the time to prepare a message for transmission, (b) the time it takes for a message to travel (latency), (c) how many words can traverse per second (bandwidth), (d) how many PEs to communicate with, and (e) the message size (Grama *et al.*, 2003). Since the first three parameters are determined by the configuration of the parallel machine, we discuss the communication costs for different parallelization schemes using the last two parameters.

2.1 | Row-wise partitioning

We first consider a row-wise partitioning, in which case each PE stores one row or several rows of $\mathbf{A}$ and one element or a portion of $\mathbf{d}$. Figure 1a illustrates the partitioning using four PEs. In order for each PE to perform its computation, we need to distribute the full vector among all the PEs. This requires an all-to-all broadcast. Based on table 4.1 in Grama *et al.* (2003), the communication cost of this communication operation is $t_s \log B + t_w m$, where B is the number of PEs, t_s is the startup time, and t_w is the per-word transfer time (Grama *et al.*, 2003). The two time parameters depend on computer architecture and performance. After the communication, each PE multiplies its m/B rows with the vector, which requires on the order of m^2/B floating-point operations.

2.2 | Column-wise partitioning

Instead of storing row(s), each PE can store the column(s) of $\mathbf{A}$. Figure 1b shows an example of using four PEs. With a column-wise partitioning, each PE can calculate a partial result locally. Each PE multiplies m/B columns of $\mathbf{A}$ with m/B elements of $\mathbf{d}$, which requires $\mathcal{O}(m^2/B)$ floating-point operations. After the local computation, we need to perform an all-to-one reduction to sum up the partial results given by each PE, which takes time $(t_s + t_w m)\log B$ (Grama *et al.*, 2003, table 4.1). After

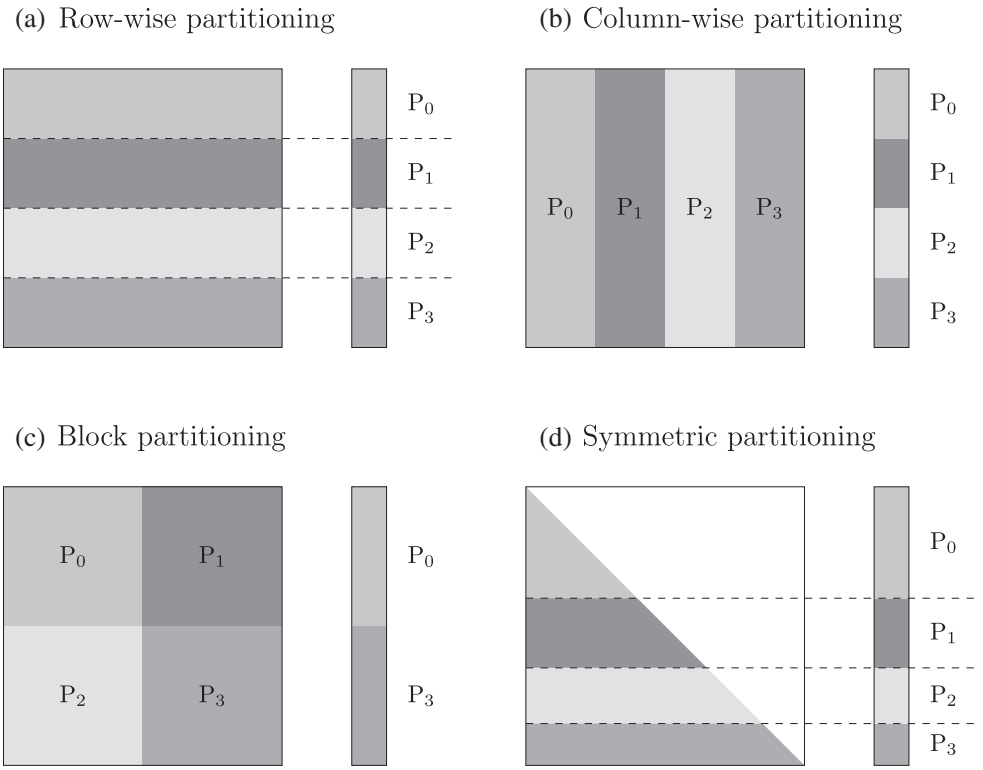

FIGURE 1 Different ways of partitioning matrices (represented by squares) and vectors (represented by bars) for parallel computation of the matrix–vector products. Different colors demonstrate the portions of matrices and vectors that are distributed over four PEs (P_0, P_1, P_2, and P_3)

the all-to-one reduction only one PE contains the full vector **q**, thus we may need to redistribute **q**. This requires a scatter operation, which takes time $t_s \log B + t_w m$ (Grama *et al.*, 2003, table 4.1). Although the column-wise partitioning circumvents the use of an expensive all-to-all broadcast operation, it increases the message size for data transfer operations. The message size for the all-to-one reduction is m, whereas the message size for the all-to-all broadcast used for the row-wise partitioning is m/B. It should be noticed that, in data assimilation applications, the matrix **A** is known to be symmetric, which means that the parallel algorithm using column-wise partitioning can also be used with row-wise partitioning.

2.3 | Block 2-D partitioning

The row-wise partitioning and column-wise partitioning are 1-D partitioning. We now consider a 2-D partitioning, which distributes the blocks of **A** among PEs. An example of the block 2-D partitioning is shown in Figure 1c. Note that the matrix **A** is equally separated by four PEs, while the vector **d** is only distributed among two PEs that own the diagonal blocks of **A**, that is, P_0 and P_3. The first step is for each PE that stores a portion of **d** to broadcast that portion to the other PEs in the same column, which requires a column-wise one-to-all broadcast. The communication time of this operation is $(t_s + t_w m/\sqrt{B}) \log \sqrt{B}$. The next step is to perform a row-wise all-to-one reduction, which requires another $(t_s + t_w m/\sqrt{B}) \log \sqrt{B}$ times. Grama *et al.* (2003) showed that computing the matrix–vector product using the block 2-D partitioning of the matrix can be faster than using 1-D partitioning for the same number of PEs. However, a potential problem is that, when using the block 2-D partitioning, the vector elements are not distributed among all the PEs and this can result in load imbalance.

2.4 | Partitioning for symmetric matrices

A possible partitioning of **A** and **d** taking into account the symmetric structure of **A** is shown in Figure 1d. The first step is to exchange the portions of **d** among PEs. This requires an all-to-all broadcast with an average message size of m/B. It should be noticed that not all PEs need to send their portion of **d** to all other PEs. In the case of four PEs, P_0 sends data to P_1, P_2 and P_3, P_1 to P_2 and P_3, and P_2 to P_3. The second step is to multiply the vector elements with the local part of the matrix. The last step is to exchange the results of local computation using an all-to-one reduction. The average message size for this operation is m/B. In the case of four PEs, P_0 collects the results from all other PEs, P_1 collects the results from P_2 and P_3, and P_2 collects the result from P_3. The last PE does not need to collect the results from other PEs. Slightly different parallelization schemes may be used for this kind of partitioning, depending on the computer architecture (Geus and Röllin,

2001). This partitioning saves the storage space of each PE (only half of the matrix is stored) and keeps the load balanced. Furthermore, this partitioning has a very large advantage for sparse matrices, in which case each PE only needs to communicate with neighboring PEs (Geus and Röllin, 2001).

In the next section, we describe a different approach for calculating large matrix–vector products that is applicable to any matrix (most useful for full or dense matrices). This approach reduces the computational cost for serial calculations of Equation (1) and the message size for communication operations.

3 | APPLICATION OF THE FAST MULTIPOLE METHOD (FMM) TO DATA ASSIMILATION

The fast multipole method (FMM) was originally presented for the rapid evaluation of all pairwise interactions between a large number of charged particles (Rokhlin, 1985; Greengard and Rokhlin, 1997). The mathematical form of this fast summation is equivalent to Equation (1). While the direct calculation of Equation (1) requires $\mathcal{O}(m^2)$ work, the FMM needs only $\mathcal{O}(m)$ operations. In addition, the FMM relies on a hierarchical division of the computational domain, which is well suited for parallel computing. In the classical FMM, the matrix $\mathbf{A}$ is given by some functions describing pairwise interactions between particles, such as those describing gravitational and electrostatic potentials. In our application, however, the matrix $\mathbf{A}$ is obtained by inverting the matrix $\mathbf{R}$. Therefore, we need a generalized FMM that does not require an underlying analytic function (Gimbutas and Rokhlin, 2003).

3.1 | Separation of the matrix–vector product into near- and far-field terms

Since the SVD-FMM is not commonly used in meteorology, we first explain the key idea behind the technique and give a simple example to aid the reader's understanding, before going into the mathematical details in later subsections. The basic idea of the SVD-FMM is to calculate Equation (1) in two parts:

$$\mathbf{q} = \mathbf{A}(\cdot, \mathbf{I}_1)\mathbf{d}(\mathbf{I}_1) + \mathbf{A}(\cdot, \mathbf{I}_2)\mathbf{d}(\mathbf{I}_2), \tag{4}$$

where the column indices of $\mathbf{A}$ and the indices of $\mathbf{d}$ are divided into two mutually independent sets denoted by $\mathbf{I}_1$ and $\mathbf{I}_2$. The indices are selected according to a partition of observations (described in more detail in Section 3.2). The first part of Equation (4), referred to as the near-field calculation, is computed using a standard matrix–vector multiplication, and the second part, referred to as the far-field calculation, is estimated using an approximation. In the SVD-FMM, the far-field calculation is computed using the singular value decomposition (SVD) of $\mathbf{A}(\cdot, \mathbf{I}_2)$. We provide a simple example to aid the reader's understanding.

Example 1. Suppose we have a matrix $\mathbf{A} \in \mathbb{R}^{2\times 3}$ and a vector $\mathbf{d} \in \mathbb{R}^3$ given by

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \quad \text{and} \quad \mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}.$$

A standard matrix–vector multiplication gives

$$\mathbf{q} = \mathbf{A}\mathbf{d} = \begin{pmatrix} a_{11}d_1 + a_{12}d_2 + a_{13}d_3 \\ a_{21}d_1 + a_{22}d_2 + a_{23}d_3 \end{pmatrix}.$$

We can also compute $\mathbf{q}$ by partitioning the problem as in Equation (4) by letting $\mathbf{I}_1 = \{2\}$ (the near field) and $\mathbf{I}_2 = \{1, 3\}$ (the far field), such that

$$\mathbf{q} = \begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix} \cdot d_2 + \begin{pmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{pmatrix} \cdot \begin{pmatrix} d_1 \\ d_3 \end{pmatrix}. \tag{5}$$

The singular value decomposition (SVD) of the submatrix in the second term is given by

$$\begin{pmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{pmatrix} = \sum_{i=1}^{2} \mathbf{u}_i \sigma_i (\mathbf{v}_i)^T,$$

where $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^2$ are the orthonormal left singular vectors, σ_1, σ_2 are the scalar singular values, and $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^2$ are the orthonormal right singular vectors. If $\sigma_1 \gg \sigma_2$, then we can truncate the SVD to give an approximation for the far-field term of Equation (5) as

$$\mathbf{q} \approx \begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix} \cdot d_2 + \mathbf{u}_1 \sigma_1 (\mathbf{v}_1)^T \cdot \begin{pmatrix} d_1 \\ d_3 \end{pmatrix}.$$

More generally, for a larger problem, we can use only the first few singular vectors and singular values to estimate the far-field calculation. If the matrix $\mathbf{A}$ remains fixed, we can use the same singular vectors and singular values to compute $\mathbf{q}$ for any vector $\mathbf{d}$. This will reduce the cost of calculating Equation (1) (unless the matrix $\mathbf{A}$ is sparse). Furthermore, storing the singular vectors and singular values requires less memory than storing the full far-field submatrix. Note that, if the matrix $\mathbf{A}$ changes too often, then we would need to perform the SVD again and again, which would be computationally expensive.

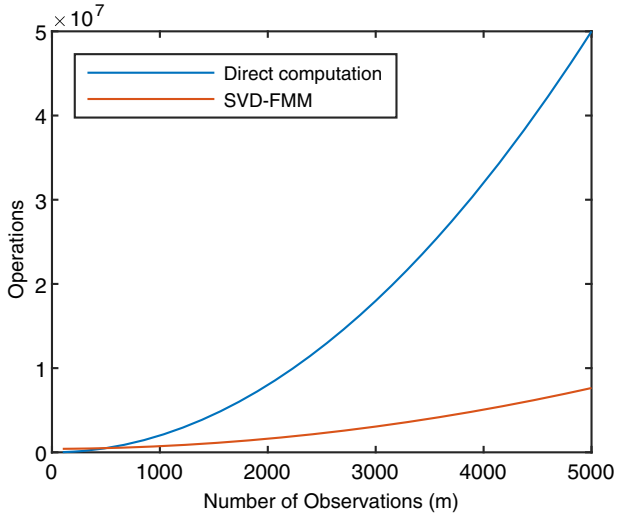

FIGURE 2 The number of floating-point operations for the direct computation of the matrix–vector product and SVD-FMM with our configuration of boxes and $p = 10$ [Colour figure can be viewed at wileyonlinelibrary.com]

The example we have discussed in this section only shows the basic idea of the SVD-FMM. The actual SVD-FMM algorithm is multi-level and more complicated. Before we can address this, we explain the partitioning of the observations that is needed for the multi-level algorithm.

3.2 | Partition of observations into a hierarchical structure of nested boxes

In this section, we describe the partition of observations and explain the notation that we use. Note that we use the nomenclature found in the FMM literature (e.g., Gimbutas and Rokhlin, 2003). Suppose that we have m observations. These could be located on a latitude–longitude grid, or irregularly distributed. We first find a minimal square (or rectangle) that covers the locations of all of the observations, and then hierarchically subdivide the observation domain into smaller and smaller boxes to generate a *box-tree*, a hierarchical structure of nested boxes. An example is given in Figure 3. In the tree structure, *level 0* refers to the biggest box that covers the entire observation domain, and *level* $l+1$ refers to the boxes that are obtained from *level* l by subdividing each box into four smaller boxes of equal size. In our particular example, we choose level 3 as the highest level, which is the minimal number of levels required for the present SVD-FMM approach. The boxes at the highest level are called *leaf boxes*.

In practical applications, the number of levels is determined such that the averaged number of observations in the leaf boxes is smaller than a prescribed value. For unevenly distributed observations, an adaptive tree structure could be created (Gimbutas and Rokhlin, 2003), in which smaller boxes are generated only where data are dense.

We number the boxes in all levels except level 0 by a *Z-order curve* as shown in Figure 3 (Gargantini, 1982). This facilitates easy formulae for box indexing, as will be described later in this subsection. We call boxes *neighbors* if they are on the same level and connect to each other. In each level, the *near field* of a box b is made of itself and all its neighbors. The *far field* of the box b is made of everything else. We use $\mathcal{N}_b$ and $\mathcal{F}_b$ to denote the lists of boxes that are in the near field and far field of box b, respectively. For example, the near field of box 16 in Figure 3 consists of the nine shaded boxes, namely $\mathcal{N}_{16} = \{7, 10, 11, 13, 15, 16, 17, 18, 19\}$, and the far field of box 16 is the other seven boxes, i.e., $\mathcal{F}_{16} = \{4, 5, 6, 8, 9, 12, 14\}$. The smallest number of boxes in a box's near field is 4, which occurs when the box is on a corner of the observation domain.

In a box-tree the *children* of box b in level l, denoted by $\mathcal{C}_b$, refer to the four boxes in level $l+1$ that are subdivided from itself, such as $\mathcal{C}_4 = \{20, 21, 22, 23\}$. In the Z-curve ordering, the indices of the children of box b are given by $4b+4$, $4b+5$, $4b+6$, and $4b+7$. The *parent* of box b, denoted by $\mathcal{P}_b$, refers to the box in a coarser level that contains box b. For instance, $\mathcal{P}_{20} = \mathcal{P}_{21} = \mathcal{P}_{22} = \mathcal{P}_{23} = \{4\}$.

The *interaction list* of box b, denoted by $\mathcal{L}_b$, is the set of boxes which are children of the neighbors of box b's parents and which are in the far field of box b. For example, $\mathcal{L}_{68} = \{32, 33, 34, 44, 45, 48, 49, 50, 51, 56, 58, 64, 65, 66, 67, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83\}$ and $\mathcal{L}_{16} = \{4, 5, 6, 8, 9, 12, 14\}$. Note that the interaction list of a box in level 2 is exactly the same as its far field.

3.3 | The SVD-FMM algorithm

In this section we describe the multi-level SVD-FMM algorithm (Gimbutas and Rokhlin, 2003). For concreteness we choose a particular configuration of boxes (as shown in Figure 3) to describe the algorithm, but it can be generalized to other configurations. For each box b we can write Equation (4) as

$$\mathbf{q}(\mathbf{I}_b) = \mathbf{A}(\mathbf{I}_b, \mathbf{I}_{\mathcal{N}_b})\mathbf{d}(\mathbf{I}_{\mathcal{N}_b}) + \mathbf{A}(\mathbf{I}_b, \mathbf{I}_{\mathcal{F}_b})\mathbf{d}(\mathbf{I}_{\mathcal{F}_b}), \tag{6}$$

where $\mathbf{I}_b$, $\mathbf{I}_{\mathcal{N}_b}$ and $\mathbf{I}_{\mathcal{F}_b}$ denote the sets of observation indices in box b, $\mathcal{N}_b$ and $\mathcal{F}_b$, respectively, and $\mathbf{A}(\mathbf{I}_b, \mathbf{I}_{\mathcal{N}_b}) = \{a_{i,j} | i \in \mathbf{I}_b,\ j \in \mathbf{I}_{\mathcal{N}_b}\}$ and $\mathbf{A}(\mathbf{I}_b, \mathbf{I}_{\mathcal{F}_b}) = \{a_{i,j} | i \in \mathbf{I}_b,\ j \in \mathbf{I}_{\mathcal{F}_b}\}$ denote submatrices of $\mathbf{A}$ that comprise specific rows and columns of $\mathbf{A}$. We call $\mathbf{q}(\mathbf{I}_b) = \{q_i | i \in \mathbf{I}_b\}$ the *target* and $\mathbf{d}(\mathbf{I}_{\mathcal{F}_b}) = \{d_j | j \in \mathbf{I}_{\mathcal{F}_b}\}$ the *source*.

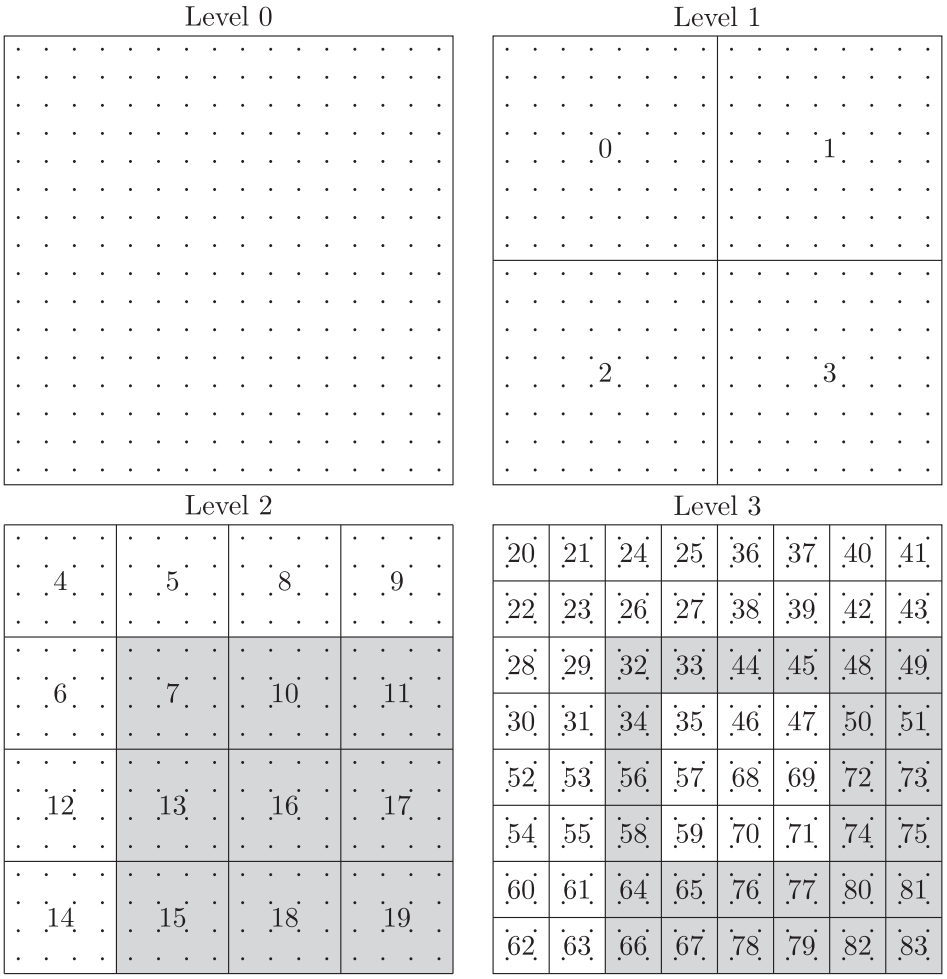

FIGURE 3 Illustration of the hierarchy of boxes of level 0–3 in our example. Level 0 refers to the box that covers the entire observation domain, while level $l+1$ refers to the boxes that are obtained by equally subdividing each box in level l into four. The dots represent the observation locations, and the numbers are the box indices. The shaded boxes in level 2 are the boxes that are in the near field of box 16, while the shaded boxes in level 3 are the boxes that are in the interaction list of box 68. The definitions of near field and interaction list can be found in the main text

The first matrix–vector product in Equation (6) is calculated from the near-field components. It is computed directly without using any approximation. The second matrix–vector product in Equation (6), containing the far-field components, is estimated by matrix decomposition, using a multi-level approach exploiting the hierarchical box-tree structure established in Section 3.2. Before giving the mathematical details, we first give an overview of the algorithm.

The key idea of the multi-level approach is to perform the SVD of the submatrices of $\mathbf{A}$ given by $\mathbf{A}(\mathbf{I}_b, \mathbf{I}_{F_b})$ and use the singular vectors and singular values just obtained to create a *multipole expansion*, a *local expansion*, and three *translation operators*, and then use them to estimate the result. The multipole expansion can be interpreted as a short representation of the subvector of $\mathbf{d}$ given by $\mathbf{d}(\mathbf{I}_b)$, which is obtained by projecting the components of $\mathbf{d}(\mathbf{I}_b)$ onto the basis given by p singular vectors, and hence is a short vector with p components. The local expansion can be considered as the short representation of the subvector of $\mathbf{d}$ given by $\mathbf{d}(\mathbf{I}_{F_b})$, which is also a vector with p components. It is computed from the multipole expansions of a group of boxes that are in b's interaction list using the translation operators. The translation operators allow exploitation of the hierarchical structure established in the box-tree, by transforming the projection from one basis of singular vectors into another basis of singular vectors. There are three kinds of translation operators: the *multipole-to-multipole* (M2M) translation operator transforms the multipole expansion of a box to the multipole expansion of its parent (see Figure 4), the *multipole-to-local* (M2L) translation operator translates the multipole expansion of a box to the local expansion of another box in the same level (see Figure 5), and the *local-to-local* (L2L) translation operator converts the local expansion of a nonleaf box to the local expansion of its children (see Figure 6).

Due to the use of these expansions and translation operators, the SVD-FMM computes matrix–vector products more efficiently than the standard approach, which is reflected in both the serial efficiency (algorithmic complexity) and parallel efficiency (see Sections 3.5 and 3.6 for more details).

We now describe the mathematical steps of the SVD-FMM algorithm in more detail. A schematic illustrating the algorithm is given in Figure 7.

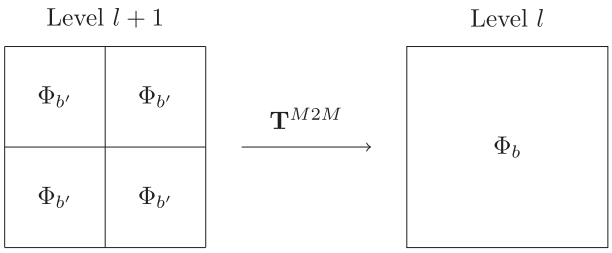

FIGURE 4 Illustration of the multipole-to-multipole translation operator: $\mathbf{T}^{M2M}$ transforms the multipole expansions ($\Phi_{b'}$) of box b' to the multipole expansion (Φ_b) of box b, where box b is the parent of box b'

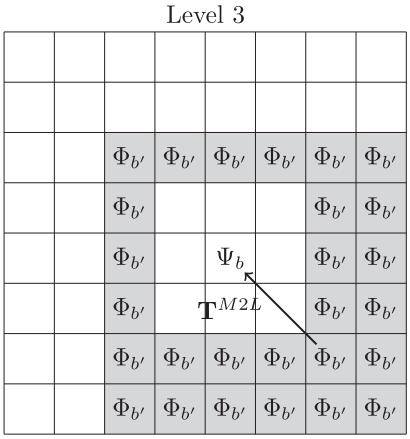

FIGURE 5 Illustration of the multipole-to-local translation operator: $\mathbf{T}^{M2L}$ transforms the multipole expansions ($\Phi_{b'}$) of box b' to the local expansion (Ψ_b) of box b, where box b' is in the interaction list of box b. The arrow illustrates the transformation from one particular b' to b. For each b' we have a unique $\mathbf{T}^{M2L}$

Initialization: *Compute the SVD of submatrices of* $\mathbf{A}$ *and the translation operators*. We compute truncated SVDs of submatrices of $\mathbf{A}$ for each box b in level 2 and 3:

$$\mathbf{A}(\mathbf{I}_{\mathcal{F}_b}, \mathbf{I}_b) \approx \sum_{k=1}^{p} \mathbf{u}_k^{src,b} s_k^{src,b} \left(\mathbf{v}_k^{src,b}\right)^{\mathrm{T}},$$
$$\mathbf{A}(\mathbf{I}_b, \mathbf{I}_{\mathcal{F}_b}) \approx \sum_{k=1}^{p} \mathbf{u}_k^{tgt,b} s_k^{tgt,b} \left(\mathbf{v}_k^{tgt,b}\right)^{\mathrm{T}}, \tag{7}$$

where $\mathbf{u}_k^{src,b} \in \mathbb{R}^{m_{\mathcal{F}_b}}$ and $\mathbf{u}_k^{tgt,b} \in \mathbb{R}^{m_b}$ are the k-th left singular vectors, $s_k^{src,b}$ and $s_k^{tgt,b}$ are the k-th singular values, and $\mathbf{v}_k^{src,b} \in \mathbb{R}^{m_b}$ and $\mathbf{v}_k^{tgt,b} \in \mathbb{R}^{m_{\mathcal{F}_b}}$ are the k-th right singular vectors. The symbols $m_{\mathcal{F}_b}$ and m_b denote the length of $\mathbf{I}_{\mathcal{F}_b}$ and $\mathbf{I}_b$, respectively. The symbol p denotes the number of singular vectors and the number of singular values.

(Note that matrix $\mathbf{A}$ is symmetric for the data assimilation problem discussed here. Therefore, in practice we actually only need to compute one SVD per box b, and for each box we have $\mathbf{u}_k^{tgt,b} = \mathbf{v}_k^{src,b}$, $\mathbf{s}_k^{tgt,b} = \mathbf{s}_k^{src,b}$ and $\mathbf{v}_k^{tgt,b} = \mathbf{u}_k^{src,b}$. However, we retain the different notations for the purpose of clarifying the algorithm.)

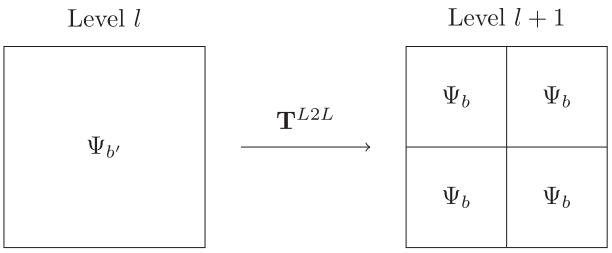

FIGURE 6 Illustration of the local-to-local translation operator: $\mathbf{T}^{L2L}$ transforms the local expansion ($\Psi_{b'}$) of box b' to the local expansions (Ψ_b) of box b, where box b is the child of box b'

The singular vectors and singular values are then used to generate multipole expansions and local expansions and a set of translation operators. The M2M translation operator (see Figure 4) converts the multipole expansion of a child box into the multipole expansion of its parent and is defined as

$$\mathbf{T}^{M2M}(k,k') = \sum_{i \in \mathbf{I}_{b'}} v_{k,i}^{src,b} v_{k',i}^{src,b'}, \quad k,k' = 1, \ldots, p \tag{8}$$

for each $b' \in C_b$, where $v_{k,i}^{src,b}$ and $v_{k,i}^{src,b'}$ denote the elements of $\mathbf{v}_k^{src,b}$ and $\mathbf{v}_k^{src,b'}$ that correspond to the i-th observation in box b'. The M2L translation operator (see Figure 5) converts the multipole expansion of box b' into the local expansion of box b, for each b' in the interaction list of b. It is defined as

$$\mathbf{T}^{M2L}(k,k') = \sum_{i \in \mathbf{I}_{b'}} v_{k,i}^{tgt,b} v_{k',i}^{src,b'}, \quad k,k' = 1, \ldots, p \tag{9}$$

for each $b' \in \mathcal{L}_b$, where $v_{k,i}^{tgt,b}$ and $v_{k,i}^{src,b'}$ denote the elements of $\mathbf{v}_k^{tgt,b}$ and $\mathbf{v}_k^{src,b'}$ that correspond to the i-th observation in box b'. The L2L translation operator (see Figure 6) transfers the local expansion of a parent box to its children and is defined as

$$\mathbf{T}^{L2L}(k,k') = \sum_{i \in \mathbf{I}_{\mathcal{F}_{b'}}} v_{k,i}^{tgt,b} v_{k',i}^{tgt,b'}, \quad k,k' = 1, \ldots, p \tag{10}$$

for each $b' = \mathcal{P}_b$, where $v_{k,i}^{tgt,b}$ and $v_{k',i}^{tgt,b'}$ denote the elements of $\mathbf{v}_k^{tgt,b}$ and $\mathbf{v}_k^{tgt,b'}$ that correspond to the i-th observation in the far field of box b'.

Step 1: *Compute the multipole expansion for the leaf boxes.* The multipole expansion for each leaf box is computed by

$$\Phi_k^b = \sum_{j \in \mathbf{I}_b} v_{k,j}^{src,b} d_j, \tag{11}$$

where $b = 20, \ldots, 83$ and $k = 1, \ldots, p$.

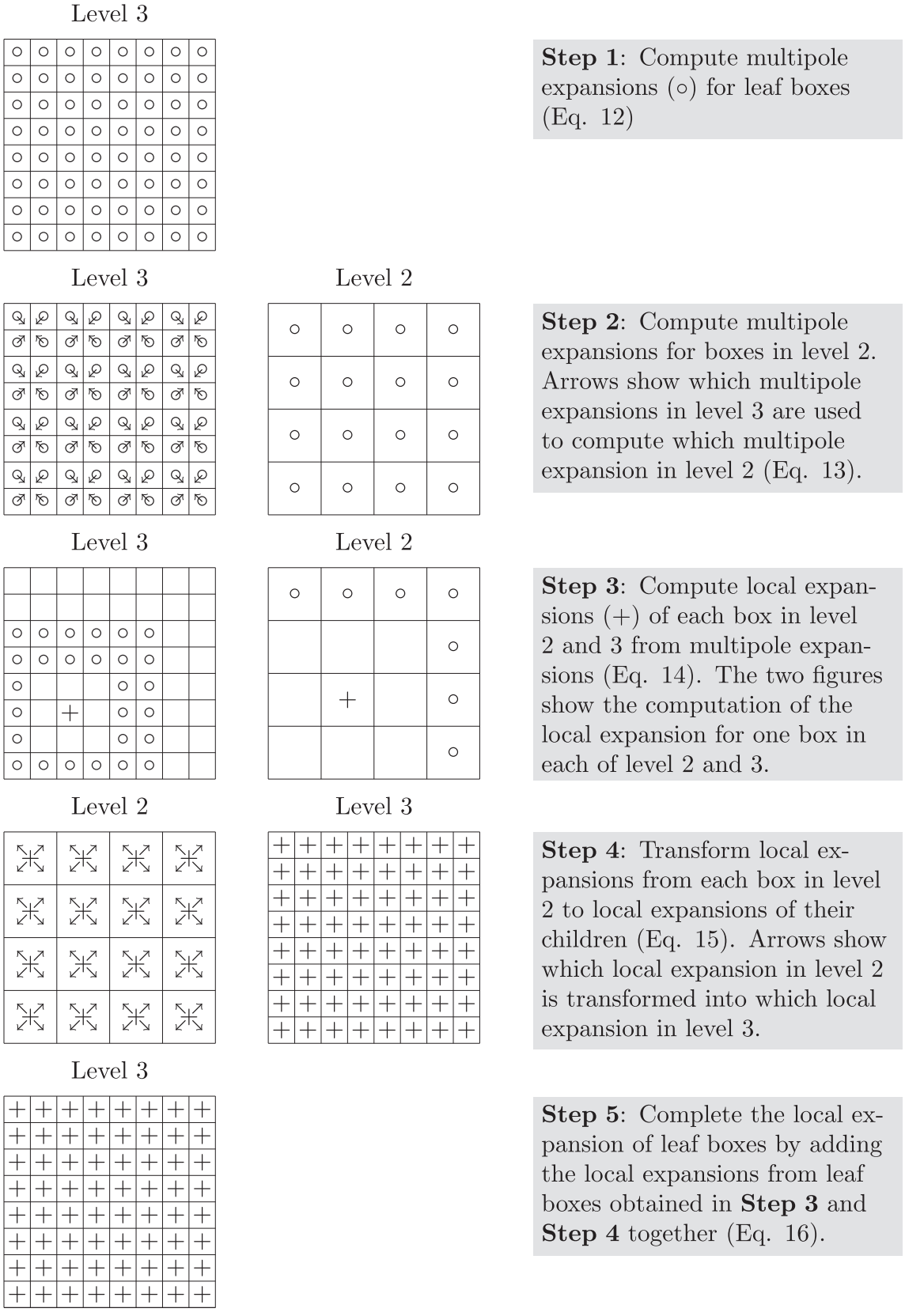

FIGURE 7 Illustration of the SVD-FMM algorithm

Step 2: *Compute the multipole expansion for the nonleaf boxes*. The multipole expansion for each box in level 2 is computed by translating the multipole expansions from its children in level 3 using the $\mathbf{T}^{M2M}$ translation operator,

$$\Phi_k^b = \sum_{b' \in C_b} \sum_{k'=1}^{p} \mathbf{T}^{M2M}(k,k')\Phi_{k'}^{b'}, \tag{12}$$

where $b = 4, \ldots, 19$, $k = 1, \ldots, p$, and b' denotes the child of b.

Step 3: *Compute the first part of the local expansion*. For each box in level 2 and 3, we translate the multipole expansions from boxes in its interaction list into local expansions by

$$\Psi_k^{b,(1)} = \sum_{b' \in \mathcal{L}_b} \sum_{k'=1}^{p} \mathbf{T}^{M2L}(k,k')\Phi_{k'}^{b'}, \tag{13}$$

where $b = 4, \ldots, 83$, $k = 1, \ldots, p$, and b' is in the interaction list of b.

Step 4: *Compute the second part of the local expansion*. We transfer the local expansions for each box in level 2 to their children in level 3 using the $\mathbf{T}^{L2L}$ translation operator:

$$\Psi_k^{b,(2)} = \sum_{k'=1}^{p} \mathbf{T}^{L2L}(k,k')\Psi_{k'}^{b',(1)}, \tag{14}$$

where $b = 20, \ldots, 83$, $k = 1, \ldots, p$, and b' is b's parent.

Step 5: *Complete the local expansion for the leaf boxes*. For each leaf box, the final local expansion is given by adding two parts together:

$$\Psi_k^b = \Psi_k^{b,(1)} + \Psi_k^{b,(2)}, \tag{15}$$

where $b = 20, \ldots, 83$ and $k = 1, \ldots, p$.

Final Step: *Adding the far-field calculation to the near-field calculation*. The final result for each leaf box ($b = 20, \ldots, 83$) is obtained by adding the far-field calculation to the near-field calculation:

$$q_i = q_i^{(1)} + q_i^{(2)}, \tag{16}$$

where $i \in \mathbf{I}_b$. The near-field calculation is given by

$$q_i^{(1)} = \sum_{j \in \mathbf{I}_{\mathcal{N}_b}} a_{i,j} d_j. \tag{17}$$

The far-field calculation is given by

$$q_i^{(2)} \approx \sum_{k=1}^{p} u_{k,i}^{tgt,b} s_k^{tgt,b} \Psi_k^b. \tag{18}$$

We note that the stepwise algorithm presented does not give a necessary order for the calculations. For example, the computation of local expansions for leaf boxes in step 3 can be carried out once step 1 is done.

3.4 | Accuracy

In this section we discuss which factors affect the accuracy of the SVD-FMM. The first aspect to consider is the relative magnitude of the near-field and far-field terms in Equation (6). The near-field term is computed directly, while the far-field term is approximated. Numerical roundoff error for the direct calculation of the near-field term is expected to be small. Therefore, the main source of error for the SVD-FMM is from the far-field calculation. Thus, if the magnitude of the near-field term is much larger than the far-field term, the error in the final result due to the SVD-FMM approximation will tend to be small.

The accuracy of the far-field calculation is determined by the accuracy of the truncated SVD of the submatrices $\mathbf{A}(\mathbf{I}_{\mathcal{F}_b}, \mathbf{I}_b)$ or $\mathbf{A}(\mathbf{I}_b, \mathbf{I}_{\mathcal{F}_b})$ in Equation (7) and the accuracy of the translation operators given by Equations (8–10). The truncation error in Equation (7) is dependent on the number of singular vectors used (p) and the $(p+1)$th singular value of the submatrices $\mathbf{A}(\mathbf{I}_{\mathcal{F}_b}, \mathbf{I}_b)$ or $\mathbf{A}(\mathbf{I}_b, \mathbf{I}_{\mathcal{F}_b})$ (e.g., Bernstein, 2009, Fact 9.14.28). Additionally, Gimbutas and Rokhlin (2003) show that the error bounds on the translation operators rely on the $(p+1)$th singular value of the submatrices. Thus, the optimal number of the singular vectors used in the SVD-FMM may depend on the application. It should be determined by considering the trade-off between accuracy and efficiency, as the more the singular vectors are used, the more accurate the results but the slower the computation.

The SVD-FMM algorithm uses truncated SVDs (Equation (7)) and, thus, may cause rank deficiency of the matrix $\mathbf{A}$. Nevertheless, the Hessian of the full variational problem will still be full rank (e.g.,Tabeart *et al.*, 2021). In addition, the matrix $\mathbf{A}$ used in the solution of the variational problem is not exactly the inverse of the matrix $\mathbf{R}$. However, only the matrix $\mathbf{A}$ (and not $\mathbf{R}$) is used in the solution of the variational problem. Furthermore, studies have shown that even approximate forms of spatial observation error correlations provide significant benefits to analysis accuracy compared with diagonal approximations (e.g., Healy and White, 2005; Stewart *et al.*, 2008; 2013).

3.5 | Algorithmic complexity

In this section we compare the number of floating-point operations (flops) required for computing matrix–vector products using the standard approach and the SVD-FMM.

Let $s = \overline{m_b}$ be the average number of observations in a leaf box (for b in the highest level), and $\mathrm{B} = \mathrm{m}{\cdot}\mathrm{s}^{-1}$ be the number of leaf boxes, where m is the total number of observations. Computing Equation (11) for a fixed b and a fixed k requires $2m_b$ operations (m_b multiplication and m_b add operations). Hence, calculating Equation (11) for each value of k requires $2m_b p$ operations. Finally, for each value of b, Equation (11) requires $\sum_{b=20}^{83} 2m_b p = 2mp$ operations. Similarly, Equation (12) requires $2Bp^2$ operations. Equation (13) requires less than $2 \times 27Bp^2$ operations for leaf boxes and $2 \times 12 \times B/4 \times p^2$ operations for boxes in level 2, because there are at most 27 entries in the interaction list of each leaf box and 12 of each box at level 2. Equation (14) requires $2Bp^2$ operations. Equation (17) requires at most $2 \times 9\,ms$ operations, since the maximum number of boxes in the near field of each box is 9. Finally, Equation (18) requires $2mp$ operations. Thus, the operation count of the entire algorithm (excluding the initialization step) approximately sums to $18ms + 4mp + 64Bp^2$ or $18ms + 4mp + 64(m/s)p^2$.

The SVDs and translation operators only need to be computed once if the distribution of observations does not change. The singular vectors, singular values, and translation operators can be used for any vector $\mathbf{d}$. However, if the distribution of the observations varies too often, then the computational costs in the initialization step and the computational cost of inverting the observation error covariance matrix could make the algorithm very expensive.

For comparison, the direct matrix–vector multiplication of Equation (1) requires $2m^2$ operations. In data assimilation, $\mathbf{q}$ is often computed using a forward and backward substitution (Golub and Van Loan, 1996; Weston *et al.*, 2014; Simonin *et al.*, 2019), which also requires $\mathcal{O}(m^2)$ operations. Figure 2 compares the operation counts for direct computation and the SVD-FMM with our configuration of boxes and $p = 10$. We observe that, once the number of observations exceeds 500, the SVD-FMM requires fewer floating-point operations than direct matrix–vector multiplication, and the difference increases with the number of observations.

3.6 | Parallelization

In this section we describe a novel parallel algorithm for the SVD-FMM. The FMM has several opportunities for parallelism (Greengard and Gropp, 1990), and our approach is not the only possibility. We include this section to provide a preliminary exploration of the potential of the SVD-FMM as a practical technique for operational data assimilation. However, we note that, for our experimental results (Section 5), we have not used this parallel algorithm. The number of observations considered in our idealized experiments is much smaller than in operational applications, so that the serial calculations can be done within an acceptable time.

We should first distribute the matrix and vector elements across PEs according to the partitioning of observations in the box-tree. For each leaf box b, we should assign the subvector of $\mathbf{d}$ given by $\mathbf{d}(\mathbf{I}_b)$ and the submatrix of $\mathbf{A}$ given by $\mathbf{A}(\mathbf{I}_b, \mathbf{I}_b \cup \mathbf{I}_{\mathcal{F}_b})$ to one PE. For each box b in level 2, we should allocate the submatrix of $\mathbf{A}$ given by $\mathbf{A}(\mathbf{I}_b, \mathbf{I}_{\mathcal{F}_b})$ to one PE. For our particular configuration of boxes, we have 64 leaf boxes and 16 nonleaf boxes in level 2, hence we could use 80 PEs. However, to make our parallelization scheme easily comparable with the parallel formulations of matrix–vector multiplication given in Sections 2.1–2.4, we actually discuss the case where we choose 16 PEs out of 64 and let them store the data for level 2 boxes. We describe a possible parallelization of each mathematical step of the SVD-FMM as follows:

- *Parallelization of the initialization step.*
  - Each PE can calculate Equation (7) independently. Once the singular vectors and singular values are obtained, the submatrices stored on each PE can be discarded.
  - To compute Equation (8), each PE that is assigned to a parent box should send $\mathbf{v}_k^{src}$ to three PEs that are assigned to its children. The calculation of the M2M translation operators would require 16 one-to-all broadcast operations to be performed simultaneously. The message size for each operation should be $mp/16$, and four PEs should be involved.
  - To compute Equation (9), each PE should send a portion of $\mathbf{v}_k^{tgt}$ to at most 27 PEs, because the maximum number of boxes in an interaction list is 27. The calculation of the M2L translation operators should require an all-to-all broadcast with the message size of mp/B.
  - To compute Equation (10), each PE should send a portion of $\mathbf{v}_k^{tgt}$ to the PE that is assigned to its parent. The computation of the L2L translation operators would require 16 all-to-one reduction operations to be carried out in the same time. The message size for each operation should be $m_{\mathcal{F}_b} p$ for b being a level 2 box. The communications required for calculating three translation operators can be done together using an all-to-all broadcast. After the initialization step, each PE that is assigned only to a leaf box will store $\mathbf{v}_k^{src}$, s_k, $\mathbf{T}^{M2M}$ and $\mathbf{T}^{M2L}$, and those assigned to both a leaf box and a level 2 box will store $\mathbf{T}^{M2L}$ and $\mathbf{T}^{L2L}$.

TABLE 1 Communication time for four distinct parallel formulations of matrix–vector multiplication with full matrices and the parallelization scheme for the SVD-FMM

Parallelization scheme	Communication operation	# PEs	Message size	Communication time
Row-wise	All-to-all broadcast	B	m/B	$t_s \log B + t_w m$
Column-wise	All-to-one reduction	B	m	$(t_s + t_w m) \log B$
	Scatter	B	m/B	$t_s \log B + t_w m$
Block	One-to-all broadcast	$\sqrt{B} - 1$	$m/\sqrt{B}$	$(t_s + t_w m/\sqrt{B}) \log \sqrt{B}$
	All-to-one reduction	$\sqrt{B} - 1$	$m/\sqrt{B}$	$(t_s + t_w m/\sqrt{B}) \log \sqrt{B}$
Symmetric	All-to-all broadcast	B	$\approx m/B$	$< t_s \log B + t_w m$
	All-to-all reduction	B	$\approx m/B$	$< t_s \log B + t_w m$
SVD-FMM	All-to-one reduction	4	p	$(t_s + t_w p) \log(4)$
	All-to-all broadcast	B	p, $2p$ or m/B	$< t_s \log B + t_w m$
	One-to-all broadcast	4	p	$(t_s + t_w p) \log(4)$

Note: The number of observations is m, the number of leaf boxes is B, and the number of singular vectors is p. # PEs represents the number of PEs that are involved in a communication operation. The symbols t_s and t_w are the startup time and the per-word transfer time, respectively, which are determined by the configuration of the parallel machine.

- *Parallelization of step 1*. This step is perfectly parallel.
- *Parallelization of step 2*. Each PE should compute $\mathbf{T}^{M2M} \cdot \Phi$ and then send the result to the PE that is assigned to its parent. Computing the multipole expansions for level 2 boxes would require an all-to-one reduction. The message size for this communication operation should be p.
- *Parallelization of step 3*. Each PE should compute $\mathbf{T}^{M2L} \cdot \Phi$ and then send the result to another PE. Each PE should collect the partial result of Equation (13) from at most 27 PEs. This step would need an all-to-all broadcast with a message size of p or $2p$.
- *Parallelization of step 4*. Each PE that is assigned to a box in level 2 should compute $\mathbf{T}^{L2L} \cdot \Phi$ and then send the result to the PEs that are assigned to its children. This step would use an one-to-all broadcast with a message size of p.
- *Parallelization of step 5*. This step is perfectly parallel.
- *Parallelization of final step*. The far-field calculation for each leaf box is perfectly parallel. For the computation of the near-field calculation, each PE should obtain the elements of $\mathbf{d}$ from at most eight PEs, which is the maximum number of neighbors for one box. This would require an all-to-all broadcast. The average message size for this operation is m/B. We can use one all-to-all broadcast to complete the communication tasks for this step and step 3.

Table 1 summarizes the communication costs for the SVD-FMM and four parallel formulations of the matrix–vector multiplications described in Section 2. The major advantage of using the SVD-FMM is that the message size for each communication operation is dramatically reduced.

In the proposed parallelization scheme, we suggested using B PEs and letting one of every four PEs conduct the computations for boxes in level 2. This could be particularly useful if the supercomputer configuration has nonuniform memory access (NUMA) nodes with locally shared memory (Grama *et al.*, 2003). In this case, we could avoid the remote communications required in step 2, step 4, and similar steps in the initialization. Alternatively, we could assign the tasks for level 2 and level 3 to different PEs, that is, using 80 PEs for our configuration of boxes. This could allow each PE to carry out a similar amount of work. More generally, we note that the practical implementation of the given parallelization scheme should be adjusted to suit the configuration of the supercomputer.

4 | EXPERIMENTAL DESIGN

Our numerical experiments are designed to demonstrate the potential of applying the SVD-FMM to compute the matrix–vector products involved in operational data assimilation. We investigate the accuracy of using the SVD-FMM under different scenarios that may occur in practical applications. We conduct serial calculations rather than using parallel computing because in this initial study we have chosen to study idealized problems of moderate size as this is sufficient for our goals.

For this initial study, we have not included matrix inversion as part of the SVD-FMM algorithm. However,

to obtain our experimental results, we require an inverse observation error covariance matrix. We use the INV function in MATLAB (MATLAB, R2020b(a)) to compute the inverses of $\mathbf{R}$ and reconditioned $\mathbf{R}$. The generation of the $\mathbf{R}$ matrix is described in Section 4.2, and reconditioning methods are given in Section 4.3. The INV function uses an LDL decomposition (Golub and Van Loan, 1996). The required SVDs of submatrices of $\mathbf{R}^{-1}$ (denoted $\mathbf{A}$) are computed using the SVDS function in MATLAB (MATLAB, R2020b(b)), which uses a Lanczos method and is efficient for finding a few singular values and vectors of a large matrix (Larsen, 1998; Baglama and Reichel, 2005).

In the results section (Section 5) we use the log of root-mean-squared error (RMSE) to assess the accuracy of SVD-FMM, which is defined as

$$\log(\mathrm{RMSE}) = \log\left(\sqrt{\frac{\sum_{i=1}^{m}\left(q_i^{fmm} - q_i^{ref}\right)^2}{m}}\right), \tag{19}$$

where the superscript *fmm* denotes the matrix–vector product computed using the SVD-FMM and *ref* denotes the matrix–vector product obtained by the standard approach. Note that the log(RMSE) shown in Section 5 is averaged over 100 realizations of q_i^{fmm}, where each one uses a different $\mathbf{d}$.

4.1 | Observation distribution and observation-minus-model departures

To simulate observation data, we assume our observations are regularly distributed over a region from 54° to 60° N and 6° W to 6° E with a grid length of 12 km. This is similar to moderately thinned geostationary satellite data used in operational forecasting over the UK (Waller *et al.*, 2016b). This results in 3,456 observations and an error covariance matrix of size 3,456 × 3,456. For convenience, we directly sample the observation-minus-model departure vector $\mathbf{d}$ from a Gaussian distribution with mean zero and (innovation) covariance given by $\mathbf{R}$ plus background error covariance. The background error covariance is modeled using the SOAR correlation function with a lengthscale of 20 km and a standard deviation of 0.6 (Section 4). The correlation lengthscale and standard deviation are selected according to Ballard *et al.* (2016) to be appropriate for kilometer-scale numerical weather prediction. The results presented in Section 5 have been averaged over 100 realizations of $\mathbf{d}$.

In practice, the observation distribution may be different at each assimilation cycle because of factors such as quality control. Therefore, in some of our experiments, we have also applied the SVD-FMM to compute the matrix–vector product with randomly chosen missing observations.

4.2 | Modeling of the observation error covariance matrix

Modeling observation error covariance matrices using correlation functions is a useful approach to deal with observation error correlations (e.g., Stewart *et al.*, 2013; Tabeart *et al.*, 2018). Here we use several correlation functions to model the observation error covariance matrices. The first is the Gaussian correlation function (e.g., Haben, 2011),

$$\mathbf{C}(i,j) = \exp\left(-\frac{|r_{i,j}|^2}{2l^2}\right), \tag{20}$$

where $l > 0$ is the correlation lengthscale and $r_{i,j}$ denotes the great-circle distance between two observations. The second is the first-order autoregressive (FOAR) correlation function, also called the Markov correlation function (e.g., Stewart *et al.*, 2013),

$$\mathbf{C}(i,j) = \exp\left(\frac{-|r_{i,j}|}{l}\right). \tag{21}$$

We also use the SOAR correlation function (e.g., Daley, 1994; Tabeart *et al.*, 2018),

$$\mathbf{C}(i,j) = \left(1 + \frac{|r_{i,j}|}{l}\right)\exp\left(\frac{-|r_{i,j}|}{l}\right), \tag{22}$$

and the Matérn 5/2 correlation function (e.g., Rasmussen and Williams, 2006),

$$\mathbf{C}(i,j) = \left(1 + \frac{\sqrt{5}r_{i,j}}{l} + \frac{5r_{i,j}^2}{3l^2}\right)\exp\left(-\frac{\sqrt{5}r_{i,j}}{l}\right). \tag{23}$$

The observation error covariance matrix can be generated using the correlation functions by

$$\mathbf{R} = \mathbf{DCD}, \tag{24}$$

where $\mathbf{D}$ is a diagonal matrix whose diagonal elements are a prescribed standard deviation. For our experiments, we choose the standard deviation as one and the correlation lengthscales as $l = 80,\ 160,\ 240$ km. These correlation lengthscales are selected based on the horizontal error correlations estimated for geostationary satellite data used in operational kilometer-scale data assimilation (Waller *et al.*, 2016b) and for AMVs (Cordoba *et al.*, 2017). It should be noted that the Markov and Matérn 5/2

correlation functions may lead to a sparse **A**, in which case the SVD-FMM may not be optimal to use. We use these correlation functions in our experiments because we wish to show that the SVD-FMM can be applied to a variety of matrices.

4.3 | Reconditioning of the observation error covariance matrix

In practical applications, observation error covariance matrices are often ill-conditioned (Haben *et al.*, 2011a; Tabeart *et al.*, 2020b). Moreover, if the matrix **R** has a large condition number, this can lead to poor convergence of the minimization problem in variational data assimilation (Weston *et al.*, 2014; Tabeart *et al.*, 2018; 2020a). In our applications, the condition number of the matrices that are created using correlation functions is dependent on the chosen function and correlation lengthscale (Haben, 2011; Haben *et al.*, 2011b).

We use two common matrix reconditioning techniques to reduce the matrix condition number in our experiments: the ridge regression method and the minimum eigenvalue method (Weston *et al.*, 2014; Bormann *et al.*, 2016; Campbell *et al.*, 2017; Tabeart *et al.*, 2020b). In the ridge regression (RR) method, the reconditioned matrix is given by

$$\mathbf{R}_{RR} = \mathbf{R} + \delta \mathbf{I}, \tag{25}$$

where **I** is the identity matrix and

$$\delta = \frac{\lambda_{max} - \lambda_{min}\kappa_{req}}{\kappa_{req} - 1}, \tag{26}$$

where λ_{max} is the maximum eigenvalue of **R**, λ_{min} is the minimum eigenvalue of **R**, and κ_{req} is the required new condition number.

The minimum eigenvalue (ME) method changes the eigenvalue spectrum of the matrix **R** by setting all eigenvalues smaller than a threshold value to the threshold value. The threshold value (T) is given in terms of the required condition number as

$$T = \lambda_{max}/\kappa_{req}. \tag{27}$$

The reconditioned matrix is then constructed via

$$\mathbf{R}_{ME} = \mathbf{E}\Lambda_{ME}\mathbf{E}^{\mathrm{T}}, \tag{28}$$

where **E** is a square matrix whose columns are the eigenvectors of **R** and Λ_{ME} is a diagonal matrix with the diagonal elements being the new eigenvalues.

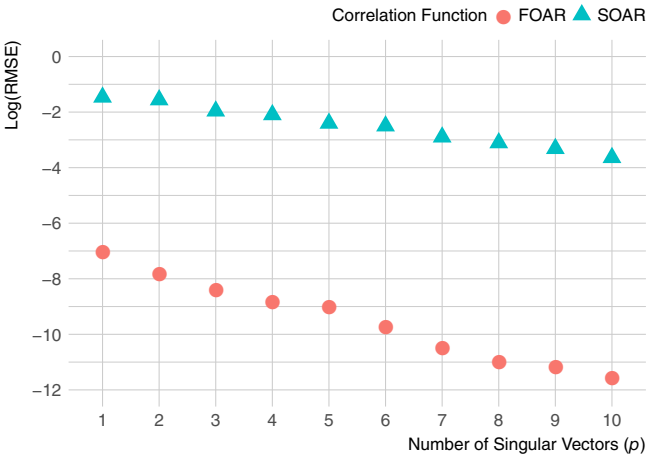

FIGURE 8 The log(RMSE) for the SVD-FMM against the number of singular vectors (p) used in the approximation. The matrices **A** are given by inverting the FOAR and SOAR covariance matrices with correlation lengthscale $l = 80$ km. The log(RMSE) is averaged over 100 realizations of **d** [Colour figure can be viewed at wileyonlinelibrary.com]

5 | RESULTS

5.1 | The number of singular vectors and the size of singular values

We carried out an experiment to assess the accuracy of the SVD-FMM, as the number of singular vectors (p) changes. As discussed in Section 3.4, we expect the accuracy to depend on both p and the $(p+1)$th singular values for each submatrix. Figure 8 provides the log(RMSE) for the SVD-FMM with the FOAR and SOAR covariance matrices as a function of p. As anticipated, the log(RMSE) decreases, and hence the accuracy increases, as more singular vectors are used. Moreover, Figure 9 demonstrates that the log(RMSE) for the SVD-FMM using p singular vectors has an approximately linear relationship with the log of the mean $(p+1)$th singular value of the submatrices, which is given by

$$\log(\bar{s}_{p+1}) = \log\left(\bar{s}^{src,b}_{p+1}\right) = \log\left(\bar{s}^{tgt,b}_{p+1}\right) \tag{29}$$

for $p = 1, \ldots, 9$ and $b = 4, \ldots, 83$. Additionally, we note that the SVD-FMM with the FOAR covariance matrix is more accurate than with the SOAR covariance matrix for all the values of p considered. This is because the mean $(p+1)$th singular value of the submatrices of the FOAR matrix is consistently smaller than that of the SOAR matrix (Figure 9). We have also carried out several other experiments using different correlation lengthscales, different correlation functions, and reconditioned covariance

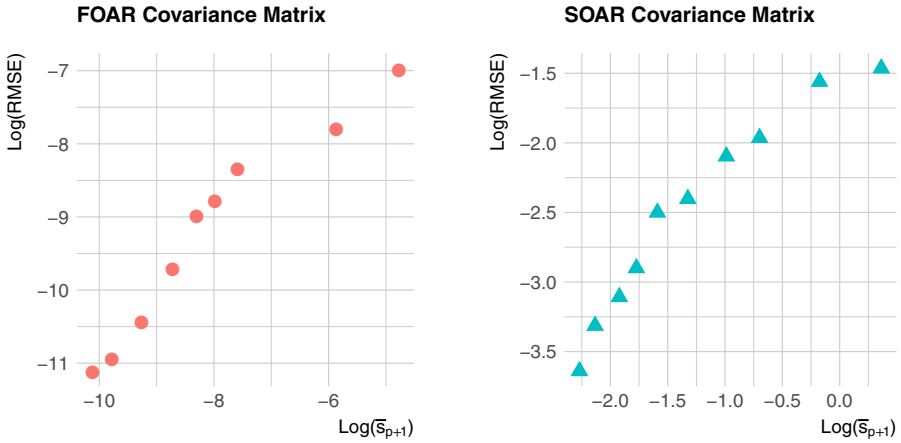

FIGURE 9 The log(RMSE) for the SVD-FMM using p singular vectors against the log of mean $(p+1)$th singular value of the submatrices of the inverse FOAR and SOAR covariance matrices used in Figure 8 [Colour figure can be viewed at wileyonlinelibrary.com]

matrices and found similar qualitative results: the accuracy of the SVD-FMM using p singular vectors relies on the $(p+1)$th singular values of the submatrices.

These results using the mean singular values of the submatrices provide some guidance as to how to set the value of p in a given application. However, as they require the computation of the SVDs of all of the relevant submatrices of $\mathbf{A}$, they are time-consuming to compute.

5.2 | Varying correlation lengthscale

Different observations can exhibit correlated errors over different lengthscales. To examine how correlation length affects the accuracy of the SVD-FMM, we use three correlation lengthscales for the SOAR correlation function: $l = 80$, 160, and 240 km. Figure 10 reveals an increase of the log(RMSE) of the SVD-FMM with correlation lengthscale. Nevertheless, we still only need a few singular vectors to obtain a small log(RMSE). However, if we desire to obtain a given accuracy, we may need to use a larger value of p for longer lengthscales. In addition, we note that the magnitude of the elements of $\mathbf{A}$ increases as correlation lengthscale increases. For a smaller lengthscale, the far-field elements of $\mathbf{A}$ are close to zero and could be neglected to give a sparse matrix. Moreover, compared with the SOAR correlation function, the Matérn and Markov functions are more likely to give a sparse $\mathbf{A}$ if the correlation lengthscale is small.

The variation of the accuracy of the SVD-FMM with correlation lengthscale can also be explained by the singular values of the submatrices. We find in our numerical experiments (not shown) that the singular values of the submatrices are smaller when the correlation lengthscale is smaller. It is known that the maximum singular value of the full matrix (inverse SOAR covariance matrix) also decreases as the correlation lengthscale reduces (Haben, 2011, section 5.3.2). Therefore, we investigated whether it

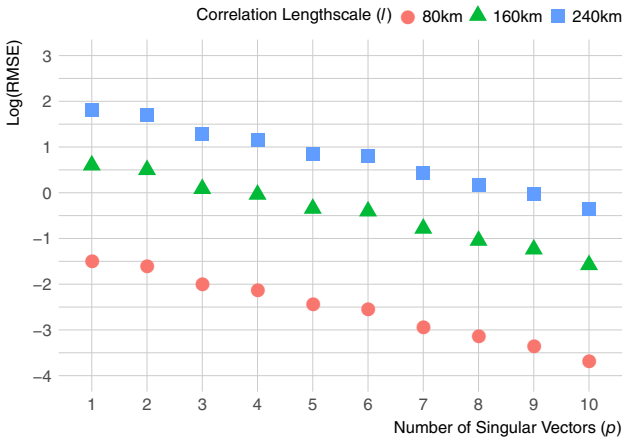

FIGURE 10 As Figure 8, but matrices $\mathbf{A}$ are given by inverting the SOAR covariance matrices with three correlation lengthscales [Colour figure can be viewed at wileyonlinelibrary.com]

would be possible to use the singular values of the full matrix to estimate the accuracy of the SVD-FMM. Thompson (1972) showed that the singular values of the submatrices are bounded by the singular values of the full matrix, i.e.,

$$\sigma_i(\mathbf{B}) \leq \sigma_i(\mathbf{A}), \quad i = 1, 2, 3, \ldots, \min\{\alpha, \beta\} \tag{30}$$

where $\mathbf{B} \in \mathbb{R}^{\alpha \times \beta}$ denotes a submatrix of $\mathbf{A}$, $\sigma_i(\mathbf{B})$ denotes the i-th singular value of $\mathbf{B}$, and $\sigma_i(\mathbf{A})$ denotes the i-th singular value of $\mathbf{A}$. Equality can be obtained for some choices of $\mathbf{B}$. However, since the dimensions of the submatrices used in SVD-FMM are much smaller than the dimensions of the full matrix, our numerical experiments showed that $\sigma_i(\mathbf{B})$ is typically much smaller than $\sigma_i(\mathbf{A})$ in practice. We compared the maximum singular values of different full matrices numerically and found that they could provide a rough guide to relative accuracies: for a given value of p, the best accuracy was obtained for the full matrix with the smallest maximum singular value.

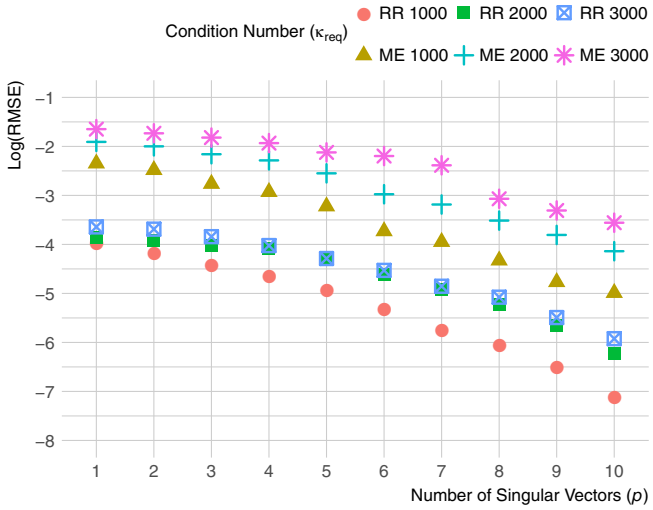

FIGURE 11 As Figure 8, but matrices $\mathbf{A}$ are given by inverting reconditioned SOAR covariance matrices with correlation lengthscale $l = 80$ km. The ridge regression (RR) and minimum eigenvalue (ME) methods are used to reduce the condition number of SOAR covariance matrix to target values (κ_{req}) [Colour figure can be viewed at wileyonlinelibrary.com]

5.3 | Reconditioned covariance matrices

The observation error covariance matrices used in data assimilation procedures are often ill-conditioned, and thus, in order to invert them, we need to reduce their condition numbers using proper techniques (Tabeart *et al.*, 2020b). Reconditioning the covariance matrices may also improve the convergence behavior of the minimization procedure (Weston *et al.*, 2014; Tabeart *et al.*, 2020a; 2021). We evaluate how reconditioning affects the accuracy of the SVD-FMM; we reduce the condition number of the SOAR covariance matrix with $l = 80$ km to three required new condition numbers ($\kappa_{req} = 1{,}000$, 2,000, and 3,000) using the RR and ME methods. Figure 11 shows that reconditioning the SOAR correlation matrix can improve the accuracy of SVD-FMM; the smaller the condition number of the SOAR covariance matrices after reconditioning, the better the accuracy.

We also find that the RR method gives smaller log(RMSE) than the ME method for the same required condition number. This difference is caused by the different ways the two reconditioning methods reduce the condition number, which leads to different size singular values of the full inverse covariance matrices satisfying:

$$\sigma_{max}(\mathbf{A}_{RR}) < \sigma_{max}(\mathbf{A}_{ME}), \tag{31}$$

where $\sigma_{max}(\mathbf{A}_{RR})$ and $\sigma_{max}(\mathbf{A}_{ME})$ denote the maximum singular value of the reconditioned matrices obtained using the RR and ME method, respectively. Combing Equation (30) and Equation (31), the leading singular values of the submatrices of $\mathbf{A}_{RR}$ are expected to typically be smaller than the leading singular values of the submatrices of $\mathbf{A}_{ME}$. Hence, the SVD-FMM with reconditioned matrices using RR method will usually have smaller error. A derivation of Equation (31) is presented in Appendix.

5.4 | Varying correlation functions

To provide more numerical evidence for the wide applicability of the SVD-FMM, we use other correlation functions than SOAR and FOAR to create covariance matrices. To allow for comparison with Figure 11, we use the RR method to reduce the condition number of all the matrices to 1,000 before inverting them. Figure 12 shows that the SVD-FMM can work well with the inverses of the matrices given by different correlation functions. We note that, although the condition number of each matrix is identical after reconditioning, the accuracy of the SVD-FMM still relates to the original condition numbers; the larger the condition number prior to reconditioning, the greater the log(RMSE). In our experiments, the FOAR correlation matrix has a condition number on the order of thousands, while the Gaussian correlation matrix is extremely poorly conditioned and has a condition number on the order of 10^{15} prior to reconditioning.

By comparing Figure 12 with Figure 11, we also observe that reducing the condition number of the Gaussian correlation matrix to 1,000 using the RR method gives larger log(RMSE) than reducing the condition number of the SOAR correlation matrix to 3,000 using the same method. Therefore, to acquire a comparable accuracy using the SVD-FMM with the same value of p, we may need to reduce the condition number of two matrices to different values. The matrix with a larger condition number prior to reconditioning may need a smaller condition number after reconditioning.

5.5 | Removing a portion of observations

In operational data assimilation, the number of observations varies each assimilation cycle due to quality control, among other reasons. This will lead to a decrease of the dimension of covariance matrix and disrupt the structure of the matrix. The resultant covariance matrix lacks some of the rows and columns of the original one. The missing observations may cause a problem if a leaf box becomes empty or contains fewer observations than p. This could be solved by using a different partition of observations. In our experiment, we randomly select the

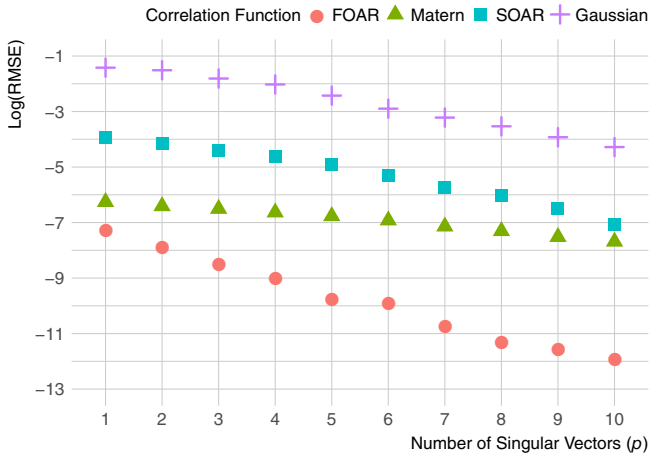

FIGURE 12 As Figure 8, but matrices **A** are given by inverting reconditioned Gaussian, FOAR, SOAR, and Matérn covariance matrices with a correlation lengthscale of 80 km. All covariance matrices are reconditioned to have a new condition number of 1,000 using the RR method prior to inversion [Colour figure can be viewed at wileyonlinelibrary.com]

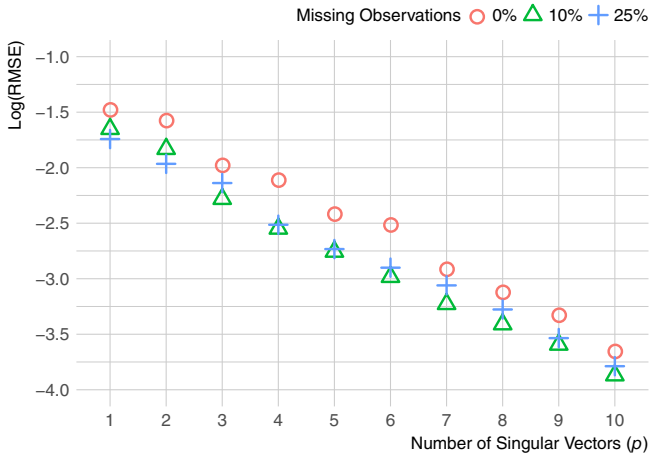

FIGURE 13 As Figure 8, but matrices **A** are given by inverse SOAR covariance matrices with correlation lengthscale $l = 80$ km and randomly chosen missing observations [Colour figure can be viewed at wileyonlinelibrary.com]

locations of the missing observations, and with up to 25% missing observations, every leaf box always contains enough observations. Figure 13 shows that the SVD-FMM can still work well with missing observations without losing accuracy. In our experiments using an inverse SOAR covariance matrix, we find that the missing observations actually lead to a slight decrease in the log(RMSE) compared with the full set of observations. The difference in the log(RMSE) between removing 10% of the observations and removing 25% of the observations is not statistically significant.

6 | CONCLUSION AND DISCUSSION

Some observations have been shown to exhibit strong spatial error correlations, e.g., Doppler radar radial winds, geostationary satellite data, and AMVs. Accounting for this information in data assimilation systems can improve analysis accuracy and forecast skill. However, it can make the computation of the products of observation weighting matrices and observation-minus-model departure vectors very expensive, in terms of not only the computational complexity, but also the communication costs in parallel computing. We have therefore investigated the use of the SVD-FMM for the rapid computation of these matrix–vector products. This numerical approximation method is best suited for full or dense precision matrices. If the observation weighting matrix is sparse, it might be appropriate to only compute the near-field calculation. Moreover, the SVD-FMM is suitable for use for large problems, when the overhead of computing the SVD-FMM expansions is outweighed by the reductions in floating-point operations and communication costs compared with alternative approaches, and where each PE has enough memory to store its part of the error covariance matrix of these observations. The exact range of the number of observations that makes the SVD-FMM useful depends on the computer architecture and the constraints of the operational schedule.

We proposed a novel possible parallelization scheme for the SVD-FMM in our application. In comparison with the parallel formulations of direct matrix–vector multiplication (Section 2), the parallelization scheme for the SVD-FMM largely reduces the size of the messages transferred. This reduces communication costs, making the use of full observation error covariance matrices associated with large spatial extents potentially feasible in operational data assimilation.

In our idealized experiments, we have examined the accuracy of the SVD-FMM, in terms of applying it to the inverses of various observation error covariance matrices. These matrices are created using commonly used correlation functions, such as Gaussian and SOAR correlation functions, and different correlation lengthscales. In some of our experiments, we have applied reconditioning methods to the covariance matrices before inverting them. We have also investigated the feasibility of the SVD-FMM with randomly chosen missing observations. We find consistent results as discussed in Section 3.4: (a) the accuracy of the SVD-FMM increases as more singular vectors are used, and (b) the variation of the accuracy is approximately linear with the mean spectrum of singular values of the submatrices used in the approximation. Furthermore, our

experiments indicate that a comparison of the maximum singular values of the full inverse matrices themselves can be used as a rough guide to determine which matrices will give more accurate results with the SVD-FMM.

The most computationally expensive parts of our current implementation of the SVD-FMM are the inversions of covariance matrices and the SVDs of the submatrices of inverse matrices. They need to be re-performed every time the observation error covariance matrix is changed. This happens when the number of observations or the distribution of observations are changed. Nevertheless, our work has shown that the SVD-FMM has potential for use in operational data assimilation for fast computation of the products of the inverse observation error covariance matrices and observation-minus-model departure vectors. This will allow a large volume of observational data to be assimilated within a short time interval, which is particularly important for applications such as convection-permitting hazardous weather forecasting.

ACKNOWLEDGEMENTS

The authors thank Amos Lawless, Nancy K. Nichols, David Simonin, and Joanne A. Waller for useful discussions. G. Hu and S. L. Dance were funded in part by the UK EPSRC DARE project (EP/P002331/1). The MATLAB code for SVD-FMM developed by the authors is available on request from the corresponding author.

AUTHOR CONTRIBUTIONS

Guannan Hu: conceptualization; data curation; formal analysis; investigation; methodology; validation; visualization; writing – original draft; writing – review and editing. **Sarah L. Dance:** conceptualization; funding acquisition; methodology; project administration; resources; supervision; validation; writing – original draft; writing – review and editing.

CONFLICT OF INTEREST

We declare we have no competing interests.

ORCID

Sarah L. Dance https://orcid.org/0000-0003-1690-3338

REFERENCES

Anderson, J.L. (2003) A local least squares framework for ensemble filtering. *Monthly Weather Review*, 131, 634–642.

Baglama, J. and Reichel, L. (2005) Augmented implicitly restarted Lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing*, 27, 19–42.

Ballard, S.P., Li, Z., Simonin, D. and Caron, J.-F. (2016) Performance of 4D-Var NWP-based nowcasting of precipitation at the Met Office for summerdate 2012. *Quarterly Journal of the Royal Meteorological Society*, 142, 472–487. https://doi.org/10.1002/qj.2665.

Bernstein, D.S. (2009) *Matrix Mathematics: Theory, Facts, and Formulas* (2nd edn). Princeton, NJ: Princeton University Press.

Bormann, N., Bonavita, M., Dragani, R., Eresmaa, R., Matricardi, M. and McNally, A. (2016) Enhancing the impact of IASI observations through an updated observation-error covariance matrix. *Quarterly Journal of the Royal Meteorological Society*, 142, 1767–1780.

Bédard, J. and Buehner, M. (2020) A practical assimilation approach to extract smaller-scale information from observations with spatially correlated errors: An idealized study. *Quarterly Journal of the Royal Meteorological Society*, 146, 468–482. https://doi.org/10.1002/qj.3687.

Campbell, W.F., Satterfield, E.A., Ruston, B. and Baker, N.L. (2017) Accounting for correlated observation error in a dual-formulation 4D variational data assimilation system. *Monthly Weather Review*, 145, 1019–1032.

Cordoba, M., Dance, S.L., Kelly, G.A., Nichols, N.K. and Waller, J.A. (2017) Diagnosing atmospheric motion vector observation errors for an operational high-resolution data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 143, 333–341. https://doi.org/10.1002/qj.2925.

Daley, R. (1994) *Atmospheric Data Analysis*. Cambridge, UK: Cambridge University Press.

Dance, S.L., Ballard, S.P., Bannister, R.N., Clark, P., Cloke, H.L., Darlington, T., Flack, D.L.A., Gray, S.L., Hawkness-Smith, L., Husnoo, N., Illingworth, A.J., Kelly, G.A., Lean, H.W., Li, D., Nichols, N.K., Nicol, J.C., Oxley, A., Plant, R.S., Roberts, N.M., Roulstone, I., Simonin, D., Thompson, R.J. and Waller, J.A. (2019) Improvements in forecasting intense rainfall: results from the FRANC (Forecasting rainfall exploiting new data assimilation techniques and novel observations of convection) project. *Atmosphere*, 10. https://www.mdpi.com/2073-4433/10/3/125.

Deng, Y. (2012) *Applied Parallel Computing*. Singapore: World Scientific Publishing Co. Pte. Ltd.

Desroziers, G., Berre, L., Chapnik, B. and Poli, P. (2005) Diagnosis of observation, background and analysis-error statistics in observation space. *Quarterly Journal of the Royal Meteorological Society*, 131, 3385–3396. https://doi.org/10.1256/qj.05.108.

Fowler, A.M., Dance, S.L. and Waller, J.A. (2018) On the interaction of observation and prior error correlations in data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 144, 48–62. https://doi.org/10.1002/qj.3183.

Gargantini, I. (1982) An effective way to represent quadtrees. *Communications of the ACM*, 25, 905–910.

Geus, R. and Röllin, S. (2001) Towards a fast parallel sparse symmetric matrix-vector multiplication. *Parallel Computing*, 27, 883–896.

Gimbutas, Z. and Rokhlin, V. (2003) A generalized fast multipole method for nonoscillatory kernels. *SIAM Journal on Scientific Computing*, 24, 796–817.

Golub, G.H. and Van Loan, C.F. (1996) *Matrix Computations*. Baltimore, Maryland: Johns Hopkins University Press.

Grama, A., Karypis, G., Kumar, V. and Gupta, A. (2003) *Introduction to Parallel Computing*. Addison-Wesley.

Greengard, L. and Gropp, W.D. (1990) A parallel version of the fast multipole method. *Computers and Mathematics with Applications*, 20, 63–71.

Greengard, L. and Rokhlin, V. (1997) A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 135, 280–292.

Guillet, O., Weaver, A.T., Vasseur, X., Michel, Y., Gratton, S. and Gürol, S. (2019) Modelling spatially correlated observation errors in variational data assimilation using a diffusion operator on an unstructured mesh. *Quarterly Journal of the Royal Meteorological Society*, 145, 1947–1967. https://doi.org/10.1002/qj.3537.

Haben, S., Lawless, A. and Nichols, N. (2011a) Conditioning and preconditioning of the variational data assimilation problem. *Computers and Fluids*, 46, 252–256.

Haben, S.A. (2011). Conditioning and preconditioning of the minimisation problem in variational data assimilation. PhD Thesis, University of Reading.

Haben, S.A., Lawless, A. and Nichols, N. (2011b) Conditioning of incremental variational data assimilation, with application to the Met Office system. *Tellus A: Dynamic Meteorology and Oceanography*, 63, 782–792. https://doi.org/10.1111/j.1600-0870.2011.00527.x.

Healy, S. and White, A. (2005) Use of discrete Fourier transforms in the 1D-Var retrieval problem. *Quarterly Journal of the Royal Meteorological Society*, 131, 63–72.

Honda, T., Miyoshi, T., Lien, G.-Y., Nishizawa, S., Yoshida, R., Adachi, S.A., Terasaki, K., Okamoto, K., Tomita, H. and Bessho, K. (2018) Assimilating All-Sky Himawari-8 Satellite Infrared Radiances: A Case of Typhoon Soudelor (2015). *Monthly Weather Review*, 146, 213–229.

Hu, G. and Franzke, C.L.E. (2017) Data assimilation in a multi-scale model. *Mathematics of Climate and Weather Forecasting*, 3, 118–139. https://doi.org/10.1515/mcwf-2017-0006.

Hu, G. and Franzke, C.L.E. (2020) Evaluation of daily precipitation extremes in reanalysis and gridded observation-based data sets over Germany. *Geophysical Research Letters*, 47. e2020GL089624.

Janjić, T., Bormann, N., Bocquet, M., Carton, J., Cohn, S., Dance, S., Losa, S., Nichols, N., Potthast, R. and Waller, J. (2018) On the representation error in data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 144, 1257–1278.

Larsen, R. (1998). https://tidsskrift.dk/daimipb/article/view/7070.

Liu, Z.-Q. and Rabier, F. (2003) The potential of high-density observations for numerical weather prediction: a study with simulated observations. *Quarterly Journal of the Royal Meteorological Society*, 129, 3013–3035.

Lorenc, A., Ballard, S., Bell, R., Ingleby, N., Andrews, P., Barker, D., Bray, J., Clayton, A., Dalby, T. and Li, D. (2000) Office global three-dimensional variational data assimilation scheme. *Quarterly Journal of the Royal Meteorological Society*, 126, 2991–3012.

MATLAB (R2020b(a)). Available at: https://uk.mathworks.com/help/matlab/ref/inv.html. Accessed 23 November 2020.

MATLAB (R2020b(b)). Available at: https://uk.mathworks.com/help/matlab/ref/svds.html. Accessed 23 November 2020.

Michel, Y. (2018) Revisiting Fisher's approach to the handling of horizontal spatial correlations of observation errors in a variational framework. *Quarterly Journal of the Royal Meteorological Society*, 144, 2011–2025. https://doi.org/10.1002/qj.3249.

Nichols, N.K. (2010) *Mathematical Concepts of Data Assimilation, 13–39*. Berlin: Springer Berlin Heidelberg.

Nino-Ruiz, E.D., Sandu, A. and Deng, X. (2019) A parallel implementation of the ensemble Kalman filter based on modified Cholesky decomposition. *Journal of Computational Science*, 36. 100654.

Rainwater, S., Bishop, C.H. and Campbell, W.F. (2015) The benefits of correlated observation errors for small scales. *Quarterly Journal of the Royal Meteorological Society*, 141, 3439–3445. https://doi.org/10.1002/qj.2582.

Rasmussen, C.E. and Williams, C.K.I. (2006) *Gaussian Processes in Machine Learning*. Cambridge, MA: MIT Press.

Rawlins, F., Ballard, S., Bovis, K., Clayton, A., Li, D., Inverarity, G., Lorenc, A. and Payne, T. (2007) The Met Office global four-dimensional variational data assimilation scheme. *Quarterly Journal of the Royal Meteorological Society*, 133, 347–362.

Rokhlin, V. (1985) Rapid solution of integral equations of classical potential theory. *Journal of computational physics*, 60, 187–207.

Schmetz, J., Pili, P., Tjemkes, S., Just, D., Kerkmann, J., Rota, S. and Ratier, A. (2002) An introduction to Meteosat Second Generation (MSG). *Bulletin of the American Meteorological Society*, 83, 977–992.

Schmit, T.J., Griffith, P., Gunshor, M.M., Daniels, J.M., Goodman, S.J. and Lebair, W.J. (2017) A closer look at the ABI on the GOES-R series. *Bulletin of the American Meteorological Society*, 98, 681–698.

Seity, Y., Brousseau, P., Malardel, S., Hello, G., Bénard, P., Bouttier, F., Lac, C. and Masson, V. (2011) The AROME-France convective-scale operational model. *Monthly Weather Review*, 139, 976–991.

Simonin, D., Waller, J.A., Ballard, S.P., Dance, S.L. and Nichols, N.K. (2019) A pragmatic strategy for implementing spatially correlated observation errors in an operational system: an application to Doppler radial winds. *Quarterly Journal of the Royal Meteorological Society*, 145, 2772–2790.

Stewart, L.M., Dance, S. and Nichols, N. (2008) Correlated observation errors in data assimilation. *International journal for numerical methods in fluids*, 56, 1521–1527.

Stewart, L.M., Dance, S.L. and Nichols, N.K. (2013) Data assimilation with correlated observation errors: experiments with a 1-D shallow water model. *Tellus A: Dynamic Meteorology and Oceanography*, 65, 19546.

Tabeart, J.M., Dance, S.L., Haben, S.A., Lawless, A.S., Nichols, N.K. and Waller, J.A. (2018) The conditioning of least-squares problems in variational data assimilation. *Numerical Linear Algebra with Applications*, 25. e2165.

Tabeart, J.M., Dance, S.L., Lawless, A.S., Migliorini, S., Nichols, N.K., Smith, F. and Waller, J.A. (2020a) The impact of using reconditioned correlated observation-error covariance matrices in the Met Office 1D-Var system. *Quarterly Journal of the Royal Meteorological Society*, 146, 1372–1390.

Tabeart, J.M., Dance, S.L., Lawless, A.S., Nichols, N.K. and Waller, J.A. (2021) New bounds on the condition number of the Hessian of the preconditioned variational data assimilation problem. *Numerical Linear Algebra with Applications*. https://doi.org/10.1002/nla.2405. e2405.

Tabeart, J.M., Dance, S.L., Lawless, A.S., Migliorini, S., Nichols, N.K., Smith, F. and Waller, J.A. (2020b) Improving the condition number of estimated covariance matrices. *Tellus A: Dynamic Meteorology and Oceanography*, 72, 1–19.

Tandeo, P., Ailliot, P., Bocquet, M., Carrassi, A., Miyoshi, T., Pulido, M. and Zhen, Y. (2020) A review of innovation-based methods to jointly estimate model and observation error covariance matrices in ensemble data assimilation. *Monthly Weather Review*, 148, 3973–3994.

Thompson, R. (1972) Principal submatrices IX: interlacing inequalities for singular values of submatrices. *Linear Algebra and its Applications*, 5, 1–12.

Waller, J., Simonin, D., Dance, S., Nichols, N. and Ballard, S. (2016a) Diagnosing observation error correlations for Doppler radar radial winds in the Met Office UKV model using observation-minus-background and observation-minus-analysis statistics. *Monthly Weather Review*, 144, 3533–3551.

Waller, J.A., Ballard, S.P., Dance, S.L., Kelly, G., Nichols, N.K. and Simonin, D. (2016b) Diagnosing horizontal and inter-channel observation error correlations for SEVIRI observations using observation-minus-background and observation-minus-analysis statistics. *Remote sensing*, 8, 581.

Waller, J.A., Bauernschubert, E., Dance, S.L., Nichols, N.K., Potthast, R. and Simonin, D. (2019) Observation error statistics for Doppler radar radial wind superobservations assimilated into the DWD COSMO-KENDA System. *Monthly Weather Review*, 147, 3351–3364.

Weston, P., Bell, W. and Eyre, J. (2014) Accounting for correlated error in the assimilation of high-resolution sounder data. *Quarterly Journal of the Royal Meteorological Society*, 140, 2420–2429.

WMO (2017). Guidelines on best practices for achieving user readiness for new meteorological satellites. WMO-No. 1187. Geneva, Switzerland: World Meteorological Organization (WMO).

Yang, J., Zhang, Z., Wei, C., Lu, F. and Guo, Q. (2017) Introducing the new generation of Chinese geostationary weather satellites, Fengyun-4. *Bulletin of the American Meteorological Society*, 98, 1637–1658.

How to cite this article: Hu, G. & Dance, S.L. (2021) Efficient computation of matrix–vector products with full observation weighting matrices in data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 1–21. Available from: https://doi.org/10.1002/qj.4170

APPENDIX A. MAXIMUM SINGULAR VALUES OF RECONDITIONED MATRICES USING RR AND ME METHODS

In this appendix we derive the result given in Equation (31). The RR method increases each of eigenvalues of the original matrix by the same amount via adding an increment given by Equation (26) to the diagonal. In contrast, the ME method changes only the smallest eigenvalues to a value given by Equation (27). Let $\lambda_{min}(\mathbf{R}_{RR})$ denote the minimum eigenvalue of $\mathbf{R}_{RR}$ and $\lambda_{min}(\mathbf{R}_{ME})$ denote the minimum eigenvalue of $\mathbf{R}_{ME}$, then we have

$$\begin{aligned}\lambda_{min}(\mathbf{R}_{RR}) &= \frac{\lambda_{max} - \lambda_{min}\kappa_{req}}{\kappa_{req} - 1} + \lambda_{min} \\ &> \frac{\lambda_{max} - \lambda_{min}\kappa_{req}}{\kappa_{req}} + \lambda_{min} = \frac{\lambda_{max}}{\kappa_{req}} = \lambda_{min}(\mathbf{R}_{ME}).\end{aligned} \tag{A1}$$

The reciprocal of the minimum eigenvalue of a matrix is the maximum eigenvalue of its inverse, hence we have

$$\lambda_{max}(\mathbf{A}_{RR}) = \frac{1}{\lambda_{min}(\mathbf{R}_{RR})} < \frac{1}{\lambda_{min}(\mathbf{R}_{ME})} = \lambda_{max}(\mathbf{A}_{ME}). \tag{A2}$$

Since the eigenvalues and singular values of a symmetric positive semidefinite matrix are the same (Bernstein, 2009, Definition 5.6.1), we obtain

$$\sigma_{max}(\mathbf{A}_{RR}) = \lambda_{max}(\mathbf{A}_{RR}) < \lambda_{max}(\mathbf{A}_{ME}) = \sigma_{max}(\mathbf{A}_{ME}) \tag{A3}$$

as required.