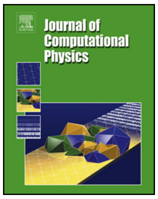
Journal of Computational Physics 357 (2018) 263–281

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp



A low-rank approach to the solution of weak constraint variational data assimilation problems

Melina A. Freitag*, Daniel L.H. Green

Department of Mathematical Sciences, University of Bath, Claverton Down, BA2 7AY, United Kingdom

ARTICLE INFO

Article history:
Received 22 February 2017
Received in revised form 17 September 2017
Accepted 26 December 2017
Available online 2 January 2018

Keywords:
Data assimilation
Weak constraint 4D-Var
Iterative methods
Matrix equations
Low-rank methods
Preconditioning

ABSTRACT

Weak constraint four-dimensional variational data assimilation is an important method for incorporating data (typically observations) into a model. The linearised system arising within the minimisation process can be formulated as a saddle point problem. A disadvantage of this formulation is the large storage requirements involved in the linear system. In this paper, we present a low-rank approach which exploits the structure of the saddle point system using techniques and theory from solving large scale matrix equations. Numerical experiments with the linear advection–diffusion equation, and the non-linear Lorenz-95 model demonstrate the effectiveness of a low-rank Krylov subspace solver when compared to a traditional solver.

1. Introduction

Data assimilation is a method for combining a numerical model with observations obtained from a physical system, in order to create a more accurate estimate for the true state of the system. One example where data assimilation is used is numerical weather prediction, however it is also applied in areas such as oceanography, glaciology and other geosciences.

A property which these applications all share is the vast dimensionality of the state vectors involved. In numerical weather prediction the systems have variables of order 10^8 [24]. In addition to the requirement that these computations to be solved quickly, the storage requirement presents an obstacle. In this paper we propose an approach for implementing the weak four-dimensional variational data assimilation method with a low-rank solution in order to achieve a reduction in storage space as well as computation time. The approach investigated here is based on a recent paper [38] which implemented this method in the setting of PDE-constrained optimisation. We introduce here a low-rank modification to GMRES in order to generate low-rank solutions in the setting of data assimilation.

This method was motivated by recent developments in the area of solving large sparse matrix equations, see [3,23,30,32,36,37], notably the Lyapunov equation

$$AX + XA^T = -BB^T$$

in which we solve for the matrix X, where A, B and X are large matrices of matching size. It is known that if the right hand side of these matrix equations are low-rank, there exist low-rank approximations to X [21]. There are a number of

* Corresponding author.
E-mail addresses: m.a.freitag@bath.ac.uk (M.A. Freitag), d.l.h.green@bath.ac.uk (D.L.H. Green).

https://doi.org/10.1016/j.jcp.2017.12.039
0021-9991/© 2018 Elsevier Inc. All rights reserved.

methods which iteratively generate low-rank solutions; see e.g. [13,26,30,32,36], and it is these ideas which are employed in this paper.

Alternative methods [14,31,39] have been considered for computing low-rank solutions, based on sequential data assimilation methods such as the Kalman filter [22,31]. Furthermore there have been developments in applying traditional model reduction techniques such as Balanced Truncation [29] and Principal Orthogonal Decomposition (POD) to data assimilation; e.g. [10,25]. In this paper we take a different approach, the data assimilation problem is considered in its full formulation, however the expensive solve of the linear system is done in a low-rank in time framework.

In the next section we introduce a saddle point formulation of weak constraint four dimensional variational data assimilation. Section 3 explains the connection between the arising linear system and the solution to matrix equations. We also introduce a low-rank approach to GMRES, and consider several preconditioning strategies. Numerical results are presented in Section 4, with an extension to time-dependent systems considered in Section 5.

2. Variational data assimilation

Variational data assimilation, initially proposed in [34,35] is one of two families of methods for data assimilation, the other being sequential data assimilation which includes the Kalman Filter and modifications [14,22,31].

We consider the discrete-time non-linear dynamical system

$$x_{k+1} = \mathcal{M}_k(x_k) + \eta_k, \tag{2.1}$$

where $x_k \in \mathbb{R}^n$ is the state of the system at time t_k and $\mathcal{M}_k : \mathbb{R}^n \to \mathbb{R}^n$ is the non-linear model operator which evolves the state from time t_k to t_{k+1} for $k = 0, \dots N-1$. The model errors are denoted η_k, and are assumed to be Gaussian with zero mean and covariance matrix $Q_k \in \mathbb{R}^{n\times n}$.

Observations of this system, $y_k \in \mathbb{R}^{p_k}$ at time t_k for $k = 0, \dots N$ are given by

$$y_k = \mathcal{H}_k(x_k) + \epsilon_k, \tag{2.2}$$

where $\mathcal{H}_k : \mathbb{R}^n \to \mathbb{R}^{p_k}$ is an observation operator, and ϵ_k is the observation error. In general, $p_k \ll n$. This observation operator $\mathcal{H}_k$ may also be non-linear, and may have explicit time dependence. The observation errors are assumed to be Gaussian, with zero mean and covariance matrix $R_k \in \mathbb{R}^{p_k\times p_k}$.

We assume that at the initial time we have an a priori estimate of the state, which we refer to as the background state, and denote x^b. This is commonly the result of a short-range forecast, or a previous assimilation, and is typically taken to be the first guess during the assimilation process. We assume that this background state has Gaussian errors with covariance matrix $B \in \mathbb{R}^{n\times n}$.

2.1. Four dimensional variational data assimilation (4D-Var)

Four dimensional variational data assimilation (4D-Var) is so called for three spatial dimensions, plus time, and to differentiate it from three-dimensional variational data assimilation (3D-Var), where we do not consider multiple observation times. In 4D-Var, we find an initial state which minimises both the weighted least squares distance to the background state x^b, and the weighted least squares distance between the model trajectory of this initial state x_k and the observations y_k for an assimilation window $[t_0, t_N]$. Mathematically, we can write this as a minimisation of a cost function, e.g. argmin $J(x)$, where

$$\begin{aligned} J(x) &= \underbrace{\frac{1}{2}(x_0 - x_0^b)^T B^{-1}(x_0 - x_0^b)}_{J_b} + \underbrace{\frac{1}{2}\sum_{i=0}^{N}(y_i - \mathcal{H}_i(x_i))^T R_i^{-1}(y_i - \mathcal{H}_i(x_i))}_{J_o} \\ &\quad \underbrace{+\frac{1}{2}\sum_{i=1}^{N}(x_i - \mathcal{M}_i(x_{i-1}))^T Q_i^{-1}(x_i - \mathcal{M}_i(x_{i-1})),}_{J_q} \\ &= \frac{1}{2}\|x_0 - x_0^b\|^2_{B^{-1}} + \frac{1}{2}\sum_{i=0}^{N}\|y_i - \mathcal{H}_i(x_i)\|^2_{R_i^{-1}} + \frac{1}{2}\sum_{i=1}^{N}\|x_i - \mathcal{M}_i(x_{i-1})\|^2_{Q_i^{-1}}, \end{aligned} \tag{2.3}$$

where $x = [x_0^T, x_1^T, \dots, x_N^T]^T$, and x_k is the model state at each timestep t_k for $k = 0, \dots, N$. This is known as *weak constraint* 4D-Var. The assumption of a perfect model, gives rise to *strong constraint* 4D-Var, and a simplification of the cost function, notably the removal of the J_q term.

The additional cost of weak constraint 4D-Var, and the difficulties in computing Q_k mean that it is not widely implemented in real world systems. However, accounting for this model error (with suitable covariances) would lead to improved accuracy, and the added potential of longer assimilation windows [17,18].

2.2. Incremental 4D-Var

To implement 4D-Var operationally, an incremental approach [11] is used, which is merely a form of Gauss–Newton iteration and generates an approximation to the solution of $x = \operatorname{argmin} J(x)$. We approximate the 4D-Var cost function by a quadratic function of an increment $\delta x^{(\ell)} = \left[(\delta x_0^{(\ell)})^T, (\delta x_1^{(\ell)})^T, \ldots, (\delta x_N^{(\ell)})^T\right]^T$ defined as

$$\delta x^{(\ell)} = x^{(\ell+1)} - x^{(\ell)}, \tag{2.4}$$

where $x^{(\ell)} = \left[(x_0^{(\ell)})^T, (x_1^{(\ell)})^T, \ldots, (x_N^{(\ell)})^T\right]^T$ denotes the ℓ-th iterate of the Gauss–Newton algorithm. Updating this estimate is implemented in an *outer loop*, whilst generating $\delta x^{(\ell)}$ is referred to as the *inner loop*. This increment $\delta x^{(\ell)}$ is a solution to the minimisation of the linearised cost function

$$\begin{aligned}\tilde{J}(\delta x^{(\ell)}) &= \frac{1}{2}(\delta x_0^{(\ell)} - b_0^{(\ell)})^T B^{-1}(\delta x_0^{(\ell)} - b_0^{(\ell)}) \\ &+ \frac{1}{2}\sum_{i=0}^{N}(d_i^{(\ell)} - H_i \delta x_i^{(\ell)})^T R_i^{-1}(d_i^{(\ell)} - H_i \delta x_i^{(\ell)}) \\ &+ \frac{1}{2}\sum_{i=1}^{N}(\delta x_i^{(\ell)} - M_i \delta x_{i-1}^{(\ell)} - c_i^{(\ell)})^T Q_i^{-1}(\delta x_i^{(\ell)} - M_i \delta x_{i-1}^{(\ell)} - c_i^{(\ell)}).\end{aligned} \tag{2.5}$$

Here $M_k \in \mathbb{R}^{n\times n}$ and $H_k \in \mathbb{R}^{n\times p_k}$, are linearisations of $\mathcal{M}_k$ and $\mathcal{H}_k$ about the current state trajectory $x^{(\ell)}$. For convenience and conciseness, we introduce

$$b_0^{(\ell)} = x_0^b - x_0^{(\ell)}, \tag{2.6}$$

$$d_k^{(\ell)} = y_k - \mathcal{H}_k(x_k^{(\ell)}), \tag{2.7}$$

$$c_k^{(\ell)} = \mathcal{M}_k(x_{k-1}^{(\ell)}) - x_k^{(\ell)}. \tag{2.8}$$

We define the following vectors in order to rewrite the cost function in a more compact form.

$$\delta x = \begin{bmatrix}\delta x_0 \\ \delta x_1 \\ \vdots \\ \delta x_N\end{bmatrix}, \quad \delta p = \begin{bmatrix}\delta x_0 \\ \delta q_1 \\ \vdots \\ \delta q_N\end{bmatrix},$$

where we have dropped the superscript for the outer loop iteration. These two vectors are related by $\delta q_k = \delta x_k - M_k \delta x_{k-1}$, or in matrix form

$$\delta p = L\delta x, \tag{2.9}$$

where

$$L = \begin{bmatrix} I & & & \\ -M_1 & I & & \\ & \ddots & \ddots & \\ & & -M_N & I\end{bmatrix} \in \mathbb{R}^{(N+1)n\times(N+1)n}. \tag{2.10}$$

Furthermore, we introduce the following matrices:

$$D = \begin{bmatrix} B & & & \\ & Q_1 & & \\ & & \ddots & \\ & & & Q_N\end{bmatrix} \in \mathbb{R}^{(N+1)n\times(N+1)n}, \quad \mathcal{R} = \begin{bmatrix} R_0 & & & \\ & R_1 & & \\ & & \ddots & \\ & & & R_N\end{bmatrix} \in \mathbb{R}^{\sum_{k=0}^{N} p_k \times \sum_{k=0}^{N} p_k},$$

$$\mathcal{H} = \begin{bmatrix} H_0 & & & \\ & H_1 & & \\ & & \ddots & \\ & & & H_N\end{bmatrix} \in \mathbb{R}^{(N+1)n\times \sum_{k=0}^{N} p_k}, b = \begin{bmatrix} b_0 \\ c_1 \\ \vdots \\ c_N\end{bmatrix} \in \mathbb{R}^{(N+1)n}, \quad d = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_N\end{bmatrix} \in \mathbb{R}^{\sum_{k=0}^{N} p_k}.$$

This allows us to write (2.5), with the superscripts dropped, as a function of δx:

$$\tilde{J}(\delta x) = \frac{1}{2}(L\delta x - b)^T D^{-1}(L\delta x - b) + \frac{1}{2}(\mathcal{H}\delta x - d)^T \mathcal{R}^{-1}(\mathcal{H}\delta x - d). \tag{2.11}$$

Minimising the cost function is equivalent to solving the linear system for the gradient. Indeed, taking the gradient of this cost function with respect to δx, we have

$$\nabla \tilde{J}(\delta x) = L^T D^{-1}(L\delta x - b) + \mathcal{H}^T \mathcal{R}^{-1}(\mathcal{H}\delta x - d). \tag{2.12}$$

Defining $\lambda = D^{-1}(b - L\delta x)$ and $\mu = \mathcal{R}^{-1}(d - \mathcal{H}\delta x)$, allows us to write the gradient at the minimum as

$$\nabla \tilde{J} = L^T \lambda + \mathcal{H}^T \mu = 0. \tag{2.13}$$

Additionally, we have

$$D\lambda + L\delta x = b, \tag{2.14}$$

$$\mathcal{R}\mu + \mathcal{H}\delta x = d, \tag{2.15}$$

and (2.13), (2.14) and (2.15) can be combined into a single linear system:

$$\begin{bmatrix} D & 0 & L \\ 0 & \mathcal{R} & \mathcal{H} \\ L^T & \mathcal{H}^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \\ \delta x \end{bmatrix} = \begin{bmatrix} b \\ d \\ 0 \end{bmatrix}, \tag{2.16}$$

which is solved for δx.

This equation is known as the saddle-point formulation for weak constraint 4D-Var, and allows us to exploit the saddle point structure for linear solves and preconditioning [5,8,38].

The saddle point matrix in (2.16), is a square symmetric indefinite matrix of size $\left(2n(N+1) + \sum_{k=0}^{N} p_k\right)$. In order to successfully solve this system we must use an iterative solver such as MINRES or GMRES as it is unfeasible with these large problem sizes to use a direct method. Additionally we require a good choice of preconditioner for a saddle point system [5–9,18], which in a data assimilation setting, has a $(1,2)$ block which is more computationally expensive than the $(1,1)$ block. The inexact constraint preconditioner [8] has been found to be an effective choice of preconditioner for the data assimilation problem [18], but application of this results in a nonsymmetric system necessitating the use of GMRES. We consider different preconditioning approaches in Section 3.4. Furthermore, to overcome the storage requirements of the matrix in (2.16), we wish to avoid forming it (and indeed as many of the submatrices as possible), which motivates the method described in the following section.

3. Low-rank approach

3.1. Kronecker formulation

As noted above, the matrix formed in the saddle point formulation is very large, as indeed are the vectors $\lambda, \mu, \delta x$. We wish to adapt the ideas developed in [38] in order to solve (2.16). This approach is dependent on the Kronecker product and the vec$(\cdot)$ operator; which are defined to be

$$\mathcal{A} \otimes \mathcal{B} = \begin{bmatrix} a_{11}\mathcal{B} & \cdots & a_{1n}\mathcal{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathcal{B} & \cdots & a_{mn}\mathcal{B} \end{bmatrix}, \quad \mathrm{vec}\,(\mathcal{C}) = \begin{bmatrix} c_{11} \\ \vdots \\ c_{1n} \\ \vdots \\ c_{mn} \end{bmatrix}.$$

We also make use of the relationship between the two:

$$(\mathcal{B}^T \otimes \mathcal{A})\mathrm{vec}\,(\mathcal{C}) = \mathrm{vec}\,(\mathcal{A}\mathcal{C}\mathcal{B})\,. \tag{3.1}$$

Employing this definition, we may rewrite (2.16) as

$$\begin{bmatrix} E_1 \otimes B + E_2 \otimes Q & 0 & I_{N+1} \otimes I_n + C \otimes M \\ 0 & I_{N+1} \otimes R & I_{N+1} \otimes H \\ I_{N+1} \otimes I_n + C^T \otimes M^T & I_{N+1} \otimes H^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \\ \delta x \end{bmatrix} = \begin{bmatrix} b \\ d \\ 0 \end{bmatrix}, \tag{3.2}$$

where we make the additional assumptions that $Q_i = Q$, $R_i = R$, $H_i = H$, $M_i = M$ and the number of observations $p_i = p$ for each i. The extended case relaxing this assumption is considered in Section 5. Here

$$C = \begin{bmatrix} 0 & & & \\ -1 & 0 & & \\ & \ddots & \ddots & \\ & & -1 & 0 \end{bmatrix}, \quad E_1 = \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix}, \quad \text{and } E_2 = \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}.$$

The matrices $C, E_1, E_2, I_{N+1} \in \mathbb{R}^{(N+1)\times(N+1)}$, whilst $B, Q, M, I_n \in \mathbb{R}^{n\times n}$, $H \in \mathbb{R}^{p\times n}$, and $R \in \mathbb{R}^{p\times p}$.

Using (3.1), we may rewrite (3.2) as the simultaneous matrix equations:

$$\begin{aligned} B\Lambda E_1 + Q\Lambda E_2 + X + MXC^T &= \mathbb{b}, \\ RU + HX &= \mathbb{d}, \\ \Lambda + M^T\Lambda C + H^T U &= 0, \end{aligned} \tag{3.3}$$

where we suppose $\lambda, \delta x, b, \mu$ and d are vectorised forms of the matrices $\Lambda, X, \mathbb{b} \in \mathbb{R}^{n\times(N+1)}$ and $U, \mathbb{d} \in \mathbb{R}^{p\times(N+1)}$ respectively. These are generalised Sylvester equations, which we solve for Λ, U and X, though for implementing incremental data assimilation, we require only δx and hence the solution X.

For standard Sylvester equations of the form $\mathcal{A}\mathcal{X} + \mathcal{X}\mathcal{B} = \mathcal{C}$, it is known that if the right hand side $\mathcal{C}$ is low-rank, then there exist low-rank approximate solutions [21]. Indeed, recent algorithms for solving these Sylvester equations have focused on constructing low-rank approximate solutions. These algorithms include Krylov subspace methods (see [37]) and ADI based methods (see [2,4,19]). It is this knowledge which motivates the following approach.

3.2. Existence of a low-rank solution

We wish to show that we can find a low-rank approximate solution to (3.2). Further to the assumption that the model and observations are not time-dependent, let us additionally assume that the model is linear and perfect. Thus $c_k = M_k(x_{k-1}) - x_k = 0$ for all k, giving

$$b = \begin{bmatrix} b_0 \\ c_1 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} b_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad \text{and hence } \mathbb{b} = \begin{bmatrix} b_0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{n\times(N+1)}. \tag{3.4}$$

Assuming R is non-singular, solving the second block-row of (3.2) for μ yields,

$$\mu = (I_{N+1} \otimes R^{-1})d - (I_{N+1} \otimes R^{-1}H)\delta x, \tag{3.5}$$

which when substituted into the third block-row of (3.2) gives

$$(I_{N+1} \otimes I_n + C^T \otimes M^T)\lambda - (I_{N+1} \otimes H^T R^{-1} H)\delta x = -(I_{N+1} \otimes H^T R^{-1})d. \tag{3.6}$$

Reformulating this as a matrix equation as before, we are left with the simultaneous (block-row) equations

$$X + MXC^T + B\Lambda E_1 + Q\Lambda E_2 = \mathbb{b} \tag{3.7}$$

$$\Lambda + M^T\Lambda C - H^T R^{-1} HX = -H^T R^{-1}\mathbb{d}. \tag{3.8}$$

Assuming M^{-1} exists, we multiply (3.8) by M^{-T} to obtain

$$M^{-T}\Lambda + \Lambda C = M^{-T}H^T R^{-1} HX - M^{-T}H^T R^{-1}\mathbb{d}. \tag{3.9}$$

Typically in real world applications, we only observe a small proportion of the state space. As such, the matrix $\mathbb{d}$ containing these observations is low-rank, as is the observation operator H. Hence the right hand side of (3.9) is low-rank.

To proceed with the proof of existence we make use of the result in [21], which states that for standard Sylvester equations of the form $\mathcal{A}\mathcal{X} + \mathcal{X}\mathcal{B} = \mathcal{C}$, if $\mathcal{A}$ and $\mathcal{B}$ have disjoint spectra, or typically spectra separated by a line, then for each matrix $\mathcal{C}$ of rank at most $k_{\mathcal{C}}$, and each $0 < \epsilon < 1$, there exists a matrix $\tilde{\mathcal{X}}$ which approximates the solution $\mathcal{X}$ by

$$\|\mathcal{X} - \tilde{\mathcal{X}}\|_2 \le \epsilon \|\mathcal{X}\|_2.$$

Here the rank of $\tilde{\mathcal{X}}$ is bounded by $\operatorname{rank}(\tilde{\mathcal{X}}) \le k_{\mathcal{C}} k_\epsilon$, where k_ϵ is dependent on the location of the spectra of $\mathcal{A}$ and $\mathcal{B}$. Given sufficiently separated spectra, this results in a low rank approximate solution $\tilde{\mathcal{X}}$. In the following, the spectra of interest are $\sigma(M)$ and $\sigma(C)$, thus to be disjoint, we require that $0 \notin \sigma(M)$. This is trivially satisfied by the assumption that M is invertible. To satisfy the stronger requirement that there is a line separating the spectra, the eigenvalues of M must be all positive or all negative.

Applying this result to (3.9), we have that Λ, or indeed an approximate solution $\tilde{\Lambda}$, is of low-rank.

Finally, multiplying (3.7) by M^{-1}, and substituting in $\tilde{\Lambda}$ gives another Sylvester equation of the form

$$M^{-1}X + XC^T = M^{-1}\left(\mathbb{b} - B\tilde{\Lambda}E_1 - Q\tilde{\Lambda}E_2\right). \tag{3.10}$$

From the assumption that the model is perfect, we see from (3.4) that $\mathbb{b}$ is indeed low-rank, being rank 1, and hence from above, so is $\tilde{\Lambda}$. Thus the right hand side of this Sylvester equation (3.10) is also low-rank. Applying once more the result from [21], we obtain the desired property that X is low-rank, or indeed there is an approximate solution $\tilde{X}$ to X which is low-rank.

We formulate this result as the following Theorem.

Theorem 3.1. *Consider the solution to the saddle point formulation of the linearised weak constraint 4D-Var problem (3.3). Let the model and observations be time-independent, with $M = M_k$, $R = R_k$, $H = H_k$, $Q = Q_k$ for all k. Furthermore, we assume there is no model error, and that the model operator M, and the covariance matrix R are invertible. If the number of observations $p \ll n$, then there exists a low-rank approximation $X_r = WV^T$ to X, where $\delta x = \mathrm{vec}\,(X)$.*

It is necessary to note that it would be unfeasible to compute low-rank solutions to (2.16) in such a way. Indeed in (3.9) the right hand side still contains X, however the observation operator allows us to know the right hand side is low-rank.

Furthermore we had to make a number of assumptions to obtain this result. Whilst the assumption that $p \ll n$ is realistic, the constant operators and covariance matrices are restrictive. However, as we will see in Section 5, relaxing some of these assumptions still results in low-rank solutions observed numerically.

3.3. Low-rank GMRES (LR-GMRES)

In order to implement the above, we suppose as in [1,38], that the matrices Λ, U, X in (3.3) have low-rank representations, with

$$\Lambda = W_\Lambda V_\Lambda^T, \qquad W_\Lambda \in \mathbb{R}^{n\times k_\Lambda}, V_\Lambda \in \mathbb{R}^{(N+1)\times k_\Lambda}, \tag{3.11}$$

$$U = W_U V_U^T, \qquad W_U \in \mathbb{R}^{p\times k_U}, V_U \in \mathbb{R}^{(N+1)\times k_U}, \tag{3.12}$$

$$X = W_X V_X^T, \qquad W_X \in \mathbb{R}^{n\times k_X}, V_X \in \mathbb{R}^{(N+1)\times k_X}, \tag{3.13}$$

where $k_\Lambda, k_U, k_X \ll n$ and $k_\Lambda, k_U, k_X \ll N$.

This allows us to rewrite (3.3) as follows:

$$\begin{aligned}
\begin{bmatrix} BW_\Lambda & QW_\Lambda & W_X & MW_X \end{bmatrix}\begin{bmatrix} V_\Lambda^T E_1 \\ V_\Lambda^T E_2 \\ V_X^T \\ V_X^T C^T \end{bmatrix} &= \mathbb{b}, \\
\begin{bmatrix} RW_U & HW_X \end{bmatrix}\begin{bmatrix} V_U^T \\ W_X^T \end{bmatrix} &= \mathbb{d}, \\
\begin{bmatrix} W_\Lambda & M^T W_\Lambda & H^T W_U \end{bmatrix}\begin{bmatrix} V_\Lambda^T \\ V_\Lambda^T C \\ V_U^T \end{bmatrix} &= 0.
\end{aligned} \tag{3.14}$$

Since using a direct solver would be infeasible, we use an iterative solver, in this case GMRES [33] to allow for flexibility in choosing a preconditioner, see Section 3.4. Algorithm 1 details a low-rank implementation of GMRES, which leads to low-rank approximate solutions to (3.2), making use of (3.14). Fundamentally this is the same as a traditional vector-based GMRES with a vector z, where instead here we have

$$\mathrm{vec}\left(\begin{bmatrix} Z_{11}Z_{12}^T \\ Z_{21}Z_{22}^T \\ Z_{31}Z_{32}^T \end{bmatrix}\right) = z.$$

Applying the concatenation $X_{k1} = [Y_{k1},\quad Z_{k1}]$, $X_{k2} = [Y_{k2},\quad Z_{k2}]$ for $k = 1, 2, 3$ is equivalent to the vector addition $x = y + z$, since $X_{k1}X_{k2}^T = Y_{k1}Y_{k2}^T + Z_{k1}Z_{k2}^T$ and hence

$$x = \mathrm{vec}\left(\begin{bmatrix} X_{11}X_{12}^T \\ X_{21}X_{22}^T \\ X_{31}X_{32}^T \end{bmatrix}\right) = \mathrm{vec}\left(\begin{bmatrix} Y_{11}Y_{12}^T + Z_{11}Z_{12}^T \\ Y_{21}Y_{22}^T + Z_{21}Z_{22}^T \\ Y_{31}Y_{32}^T + Z_{31}Z_{32}^T \end{bmatrix}\right) = y + z.$$

Note that here we employ the same notation as in [38], using the brackets {} as a concatenation and truncation operation. Furthermore, after applying the matrix multiplication and the preconditioning, we also truncate the resulting matrices. How this truncation could be implemented is also treated in [38], with options including a truncated singular value decomposition, possibly through Matlab's inbuilt `svds` function, or a skinny QR factorisation. In the numerical results to follow, we use a modification of the Matlab `svds` function.

In order to compute the inner product $\langle w, v^{(i)}\rangle$ which arises in GMRES when computing the entries of the Hessenberg matrix (see line 11 in Algorithm 1), we make use of the relation between the trace and vec operators:

$$\operatorname{trace}(A^T B) = \operatorname{vec}(A)^T \operatorname{vec}(B)\,.$$

Since here

$$\operatorname{vec}\left(\begin{bmatrix} W_{11}W_{12}^T \\ W_{21}W_{22}^T \\ W_{31}W_{32}^T \end{bmatrix}\right) = w \quad \text{and} \quad \operatorname{vec}\left(\begin{bmatrix} V_{11}^{(i)}(V_{12}^{(i)})^T \\ V_{21}^{(i)}(V_{22}^{(i)})^T \\ V_{31}^{(i)}(V_{32}^{(i)})^T \end{bmatrix}\right) = v^{(i)},$$

we see that we may compute the inner product $\langle w, v^{(i)}\rangle$ as

$$\begin{aligned}\langle w, v^{(i)}\rangle =&\operatorname{trace}\left((W_{11}W_{12}^T)^T(V_{11}^{(i)}(V_{12}^{(i)})^T)\right) + \operatorname{trace}\left((W_{21}W_{22}^T)^T(V_{21}^{(i)}(V_{22}^{(i)})^T)\right) \\ &+ \operatorname{trace}\left((W_{31}W_{32}^T)^T(V_{31}^{(i)}(V_{32}^{(i)})^T)\right),\end{aligned} \tag{3.15}$$

by considering the submatrices which make up the vectors w and $v^{(i)}$. Importantly however, the matrices formed in (3.15) do not exploit the low-rank nature of the submatrices, being $(N+1)\times(N+1)$ matrices. Fortunately, using the properties of the trace operator, we may consider instead:

$$\begin{aligned}\langle w, v^{(i)}\rangle &= \operatorname{trace}\left(W_{11}^T V_{11}^{(i)}(V_{12}^{(i)})^T W_{12}\right) + \operatorname{trace}\left(W_{21}^T V_{21}^{(i)}(V_{22}^{(i)})^T W_{22}\right) \\ &\quad + \operatorname{trace}\left(W_{31}^T V_{31}^{(i)}(V_{32}^{(i)})^T W_{32}\right),\end{aligned} \tag{3.16}$$

and hence compute the trace of smaller matrices. In line 11 of Algorithm 1, we compute (3.16) as `traceproduct`(W_{11}, $W_{12}, W_{21}, W_{22}, W_{31}, W_{32}, V_{11}^{(i)}, V_{12}^{(i)}, V_{21}^{(i)}, V_{22}^{(i)}, V_{31}^{(i)}, V_{32}^{(i)}$).

The matrix vector multiplication Az in traditional GMRES, is implemented in LR-GMRES by considering the low-rank form of the saddle point equations generated in (3.14). The concatenation is explicitly written in Algorithm 2 and is denoted `Amult` in Algorithm 1.

Note that we have considered traditional GMRES when implementing LR-GMRES, however it would require only a small modification to allow for restarted GMRES. All that remains to consider is preconditioning LR-GMRES, which is implemented in Algorithm 1 through the `Aprec` function.

Due to the truncation steps within the algorithm, introducing a low-rank approximation (by removing small singular values), LR-GMRES does not minimise the residual in the same sense as traditional GMRES. Hence LR-GMRES is more precisely a form of inexact GMRES.

3.4. Preconditioning LR-GMRES

We return to the saddle point problem in (2.16). Many approaches exist for preconditioning saddle point problems, a number of which are detailed in [5,6]. However, the data assimilation setting introduces an unusual situation where the $(1,2)$ block $\begin{bmatrix} L \\ \mathcal{H} \end{bmatrix}$ of the saddle point matrix is more computationally expensive than the $(1,1)$ block $\begin{bmatrix} D & 0 \\ 0 & \mathcal{R} \end{bmatrix}$. In [15,18] it is noted that the inexact constraint preconditioner [7–9] is an effective choice:

$$\mathcal{P} = \begin{bmatrix} D & 0 & \tilde{L} \\ 0 & \mathcal{R} & 0 \\ \tilde{L}^T & 0 & 0 \end{bmatrix}, \tag{3.17}$$

provided a good approximation $\tilde{L}$ to $L = I_{N+1} \otimes I_n + C \otimes M$ is chosen. Using an inexact constraint preconditioner requires the use of GMRES since the resulting system is nonsymmetric.

Two further requirements must be considered when implementing a preconditioner for LR-GMRES. In order to maintain the low-rank structure we wish to write this in Kronecker form, however we must also consider the inverse of the preconditioner. It is the implementation of the inverse in Kronecker form which allows us to write this as a simple matrix multiplication as in (3.14) for the saddle point matrix.

We present here a number of different choices of preconditioner for LR-GMRES.

Algorithm 1 Low-rank GMRES (LR-GMRES).

```
Choose $X_{11}^{(0)}, X_{12}^{(0)}, X_{21}^{(0)}, X_{22}^{(0)}, X_{31}^{(0)}, X_{32}^{(0)}$.
$\{\tilde{X}_{11}, \tilde{X}_{12}, \tilde{X}_{21}, \tilde{X}_{22}, \tilde{X}_{31}, \tilde{X}_{32}\} = \texttt{Amult}(X_{11}^{(0)}, X_{12}^{(0)}, X_{21}^{(0)}, X_{22}^{(0)}, X_{31}^{(0)}, X_{32}^{(0)})$.
$V_{11} = \{B_{11}, \quad -\tilde{X}_{11}\}$,      $V_{12} = \{B_{12}, \quad \tilde{X}_{12}\}$,
$V_{21} = \{B_{21}, \quad -\tilde{X}_{21}\}$,      $V_{22} = \{B_{22}, \quad \tilde{X}_{22}\}$,
$V_{31} = \{B_{31}, \quad -\tilde{X}_{31}\}$,      $V_{32} = \{B_{32}, \quad \tilde{X}_{32}\}$.
$\xi = [\xi_1, 0, \ldots, 0]$, $\xi_1 = \sqrt{\texttt{traceproduct}(V_{11}^{(1)}, \ldots, V_{11}^{(1)}, \ldots)}$.
for $k = 1, \ldots$ do
  $\{Z_{11}^{(k)}, Z_{12}^{(k)}, Z_{21}^{(k)}, Z_{22}^{(k)}, Z_{31}^{(k)}, Z_{32}^{(k)}\} = \texttt{Aprec}(V_{11}^{(k)}, V_{12}^{(k)}, V_{21}^{(k)}, V_{22}^{(k)}, V_{31}^{(k)}, V_{32}^{(k)})$
  $\{W_{11}, W_{12}, W_{21}, W_{22}, W_{31}, W_{32}\} = \texttt{Amult}(Z_{11}^{(k)}, Z_{12}^{(k)}, Z_{21}^{(k)}, Z_{22}^{(k)}, Z_{31}^{(k)}, Z_{32}^{(k)})$.
  for $i = 1, \ldots, k$ do
    $h_{i,k} = \texttt{traceproduct}(W_{11}, \ldots, V_{11}^{(i)}, \ldots)$,
    $W_{11} = \{W_{11}, \quad h_{i,k}V_{11}^{(i)}\}$,      $W_{12} = \{W_{12}, \quad V_{12}^{(i)}\}$,
    $W_{21} = \{W_{21}, \quad h_{i,k}V_{21}^{(i)}\}$,      $W_{22} = \{W_{22}, \quad V_{22}^{(i)}\}$,
    $W_{31} = \{W_{31}, \quad h_{i,k}V_{31}^{(i)}\}$,      $W_{32} = \{W_{22}, \quad V_{32}^{(i)}\}$.
  end for
  $h_{k+1,k} = \sqrt{\texttt{traceproduct}(W_{11}, \ldots, W_{11}, \ldots)}$
  $V_{11}^{(k+1)} = W_{11}/h_{k+1,k}$,      $V_{12}^{(k+1)} = W_{12}$,
  $V_{21}^{(k+1)} = W_{21}/h_{k+1,k}$,      $V_{22}^{(k+1)} = W_{22}$,
  $V_{31}^{(k+1)} = W_{31}/h_{k+1,k}$,      $V_{32}^{(k+1)} = W_{32}$.
  Apply Givens rotations to $k$th column of $h$, i.e.
  for $j = 1, \ldots k-1$ do
    $\begin{bmatrix} h_{j,k} \\ h_{j+1,k} \end{bmatrix} = \begin{bmatrix} c_j & s_j \\ -\bar{s}_j & c_j \end{bmatrix} \begin{bmatrix} h_{j,k} \\ h_{j+1,k} \end{bmatrix}$
  end for
  Compute $k$th rotation, and apply to $\xi$ and last column of $h$.
  $\begin{bmatrix} \xi_k \\ \xi_{k+1} \end{bmatrix} = \begin{bmatrix} c_k & s_k \\ -\bar{s}_k & c_k \end{bmatrix} \begin{bmatrix} \xi_k \\ 0 \end{bmatrix}$,      $h_{k,k} = c_k h_{k,k} + s_k h_{k+1,k}$,
                                                  $h_{k+1,k} = 0$.
  if $|\xi_{k+1}|$ sufficiently small then
    Solve $\tilde{H}\tilde{y} = \xi$, where the entries of $\tilde{H}$ are $h_{i,k}$.
    $Y_{11} = \{\tilde{y}_1 V_{11}^{(1)}, \ \ldots, \ \tilde{y}_k V_{11}^{(k)}\}$,      $Y_{12} = \{\tilde{y}_1 V_{12}^{(1)}, \ \ldots, \ \tilde{y}_k V_{12}^{(k)}\}$
    $Y_{21} = \{\tilde{y}_1 V_{11}^{(1)}, \ \ldots, \ \tilde{y}_k V_{21}^{(k)}\}$,      $Y_{22} = \{\tilde{y}_1 V_{22}^{(1)}, \ \ldots, \ \tilde{y}_k V_{22}^{(k)}\}$
    $Y_{31} = \{\tilde{y}_1 V_{31}^{(1)}, \ \ldots, \ \tilde{y}_k V_{31}^{(k)}\}$,      $Y_{32} = \{\tilde{y}_1 V_{32}^{(1)}, \ \ldots, \ \tilde{y}_k V_{32}^{(k)}\}$
    $\{\tilde{Y}_{11}, \tilde{Y}_{12}, \tilde{Y}_{21}, \tilde{Y}_{22}, \tilde{Y}_{31}, \tilde{Y}_{32}\} = \texttt{Aprec}(Y_{11}, Y_{12}, Y_{21}, Y_{22}, Y_{31}, Y_{32})$
    $X_{11} = \{X_{11}^{(0)}, \ \tilde{Y}_{11}\}$,      $X_{12} = \{X_{12}^{(0)}, \ \tilde{Y}_{12}\}$
    $X_{21} = \{X_{21}^{(0)}, \ \tilde{Y}_{21}\}$,      $X_{22} = \{X_{22}^{(0)}, \ \tilde{Y}_{22}\}$
    $X_{31} = \{X_{31}^{(0)}, \ \tilde{Y}_{31}\}$,      $X_{32} = \{X_{32}^{(0)}, \ \tilde{Y}_{32}\}$
    break
  end if
end for
```

Algorithm 2 Matrix multiplication (`Amult`).

```
Input: $W_{11}, W_{12}, W_{21}, W_{22}, W_{31}, W_{32}$
Output: $Z_{11}, Z_{12}, Z_{21}, Z_{22}, Z_{31}, Z_{32}$
  $Z_{11} = [BW_{11}, \quad QW_{11}, \quad W_{31}, \quad MW_{31}]$,      $Z_{12} = [E_1 W_{12}, \quad E_2 W_{12}, \quad W_{32}, \quad CW_{32}]$,
  $Z_{21} = [RW_{21}, \quad HW_{31}]$,                                  $Z_{22} = [W_{22}, \quad W_{32}]$,
  $Z_{31} = [W_{11}, \quad M^T W_{11}, \quad H^T W_{21}]$,               $Z_{32} = [W_{12}, \quad C^T W_{12}, \quad W_{22}]$
```

3.4.1. *Inexact constraint preconditioner*

As mentioned above, the inexact constraint preconditioner [7] has been seen to be an effective preconditioner for the saddle point formulation of weak constraint 4D-Var [18], provided a suitable choice of approximation of L is taken.

The inverse of the inexact constraint preconditioner (3.17) is given by

$$\mathcal{P}^{-1} = \begin{bmatrix} 0 & 0 & \tilde{L}^{-T} \\ 0 & \mathcal{R}^{-1} & 0 \\ \tilde{L}^{-1} & 0 & -\tilde{L}^{-1}D\tilde{L}^{-T} \end{bmatrix}, \tag{3.18}$$

which includes the term $\tilde{L}^{-1}$. In order to implement this in LR-GMRES, we write $\tilde{L}^{-1}$ in Kronecker form. This restricts the choice of $\tilde{L}$, however taking an approximation $\tilde{L}$ of the form $I_{N+1} \otimes I_n + C \otimes \tilde{M}$, where $\tilde{M}$ is an approximation to M, the structure of L is maintained. Additionally, we can write the inverse in Kronecker form as

$$\begin{aligned} \tilde{L}^{-1} &= I_{N+1} \otimes I_n - C \otimes \tilde{M} + C^2 \otimes \tilde{M}^2 - \ldots + C^N \otimes \tilde{M}^N \\ &= I_{N+1} \otimes I_n + \sum_{k=1}^{N} (-1)^k C^k \otimes \tilde{M}^k. \end{aligned} \tag{3.19}$$

Despite being able to write this in Kronecker form, this results in an unfeasible number of terms for large N, furthermore for close approximations $\tilde{M}$ to the model matrix M, the computations are expensive. A possibility is therefore to approximate $\tilde{L}^{-1}$ by truncating (3.19) after a few terms.

Truncating after one term we obtain the approximation $\tilde{L}^{-1} = I_{n(N+1)}$. Hence in Kronecker form we can then write the resulting inverse of the preconditioner as:

$$\mathcal{P}_I^{-1} = \begin{bmatrix} 0 & 0 & I_{N+1} \otimes I_n \\ 0 & I_{N+1} \otimes R^{-1} & 0 \\ I_{N+1} \otimes I_n & 0 & -E_1 \otimes B - E_2 \otimes Q \end{bmatrix}. \tag{3.20}$$

To illustrate a possible choice of the `Aprec` function, we present the application of (3.20) as Algorithm 3.

Algorithm 3 Inexact constraint preconditioner $\tilde{L}^{-1} = I_{n(N+1)}$ (`Aprec`).

```
Input: $W_{11}, W_{12}, W_{21}, W_{22}, W_{31}, W_{32}$
Output: $Z_{11}, Z_{12}, Z_{21}, Z_{22}, Z_{31}, Z_{32}$
    $Z_{11} = W_{31}$,                              $Z_{12} = W_{32}$,
    $Z_{21} = R^{-1} W_{21}$,                       $Z_{22} = W_{22}$,
    $Z_{31} = [W_{11}, \ -BW_{31}, \ -QW_{31}]$,    $Z_{32} = [W_{12}, \ E_1 W_{32}, \ E_2 W_{32}]$
```

If we take $\tilde{M} = I_n$ we may consider the approximation $\hat{L} = I_{N+1} \otimes I_n + C \otimes I_n$. Truncating the resulting inverse after two terms we compute that the Kronecker inverse of the preconditioner is

$$\hat{\mathcal{P}}_{\hat{L}}^{-1} = \begin{bmatrix} 0 & 0 & I \otimes I - C \otimes I \\ 0 & I \otimes R^{-1} & 0 \\ I \otimes I - C \otimes I & 0 & J \end{bmatrix}, \tag{3.21}$$

where $J = -(I \otimes I - C \otimes I)(E_1 \otimes B)(I \otimes I - C^T \otimes I) - (I \otimes I - C \otimes I)(E_2 \otimes Q)(I \otimes I - C^T \otimes I)$, and we drop the subscripts for the identities.

An alternative approach is to consider an inexact constraint preconditioner where we approximate $\mathcal{H}$ in (2.16) in addition to L. In this example we approximate L by $\tilde{L} = I$, and using the exact $\mathcal{H}$, we obtain

$$\mathcal{P}_{I\mathcal{H}} = \begin{bmatrix} D & 0 & I \\ 0 & \mathcal{R} & \mathcal{H} \\ I & \mathcal{H}^T & 0 \end{bmatrix}. \tag{3.22}$$

The inverse of which is

$$\mathcal{P}_{I\mathcal{H}}^{-1} = \begin{bmatrix} \mathcal{H}^T \mathcal{F} \mathcal{H} & -\mathcal{H}^T \mathcal{F} & I - \mathcal{H}^T \mathcal{F} \mathcal{H} D \\ -\mathcal{F} \mathcal{H} & \mathcal{F} & \mathcal{F} \mathcal{H} D \\ I - D\mathcal{H}^T \mathcal{F} \mathcal{H} & D\mathcal{H}^T \mathcal{F} & D\mathcal{H}^T \mathcal{F} \mathcal{H} D - D \end{bmatrix}, \tag{3.23}$$

where $\mathcal{F} = (\mathcal{H} D \mathcal{H}^T + \mathcal{R})^{-1} = (E_1 \otimes (HBH^T + R)^{-1}) + (E_2 \otimes (HQH^T + R)^{-1})$. If H is computationally expensive (such as if H is not a simple interpolatory observation operator), this choice of preconditioner may prove unfeasible.

3.4.2. Schur complement preconditioners

An alternative choice of preconditioner is a Schur complement preconditioner, such as the block diagonal preconditioner

$$\mathcal{P}_D = \begin{bmatrix} D & 0 & 0 \\ 0 & \mathcal{R} & 0 \\ 0 & 0 & \tilde{\mathcal{S}} \end{bmatrix}, \tag{3.24}$$

where $\tilde{\mathcal{S}}$ is an approximation to the Schur-complement

$$\mathcal{S} = -L^T D^{-1} L - \mathcal{H}^T \mathcal{R}^{-1} \mathcal{H}.$$

This choice of preconditioner is used in [38], and allows the use of LR-MINRES, though in Section 4.2 we use LR-GMRES to compare the different choices as in the full-rank case, GMRES and MINRES are theoretically equivalent for symmetric systems.

As an approximation to the Schur complement we consider

$$\tilde{\mathcal{S}} = -\tilde{L}^T D^{-1} \tilde{L}, \tag{3.25}$$

the inverse of which, $\tilde{\mathcal{S}}^{-1} = -\tilde{L}^{-1} D \tilde{L}^{-T}$ is familiar as the $(3, 3)$ term in the inexact constraint preconditioner inverse (3.18). As such we must approximate this by truncating the expansion of $\tilde{L}^{-1}$ (3.19) as before. Considering the approximation

$\hat{L} = I_{N+1} \otimes I_n + C \otimes I_n$ and truncating after two terms as before, the block diagonal Schur complement preconditioner may be implemented in the same way as the inexact constraint preconditioner (3.21) above. This results in

$$\mathcal{P}_{D\hat{L}}^{-1} = \begin{bmatrix} E_1 \otimes B^{-1} + E_2 \otimes Q^{-1} & 0 & 0 \\ 0 & I \otimes R^{-1} & 0 \\ 0 & 0 & J \end{bmatrix}, \tag{3.26}$$

where $J = -(I \otimes I - C \otimes I)(E_1 \otimes B)(I \otimes I - C^T \otimes I) - (I \otimes I - C \otimes I)(E_2 \otimes Q)(I \otimes I - C^T \otimes I)$ as before.

An alternative method for implementing the Schur complement approximation (3.25) in a low-rank form is detailed in [38]. Instead of truncating the resulting inverse, and applying the technique used in Algorithm 3, the relationship between the Kronecker product and Sylvester equations is exploited. In order to solve $\tilde{\mathcal{S}} Z_{31} Z_{32}^T = W_{31} W_{32}^T$, the Kronecker form

$$-(I \otimes I + C^T \otimes \tilde{M}^T)(E_1 \otimes B^{-1} + E_2 \otimes Q^{-1})(I \otimes I + C \otimes \tilde{M})\text{vec}\left(Z_{31} Z_{32}^T\right) = \text{vec}\left(W_{31} W_{32}^T\right),$$

is written as two consecutive Sylvester equations. These resulting Sylvester equations are solved one after the other using a low-rank solver such as an ADI [2,4] or Krylov [36] method to generate a low-rank approximation $X_{31} X_{32}^T$. It is this approach which we employ in our numerical implementations in Section 4.2.

An alternative Schur complement preconditioner is the block triangular Schur complement preconditioner, which requires the use of LR-GMRES unlike the block diagonal one above. This choice uses approximations to L, $\mathcal{H}$, and the Schur complement $\mathcal{S}$,

$$\mathcal{P}_T = \begin{bmatrix} D & 0 & \tilde{L} \\ 0 & \mathcal{R} & \tilde{\mathcal{H}} \\ 0 & 0 & \tilde{\mathcal{S}} \end{bmatrix}. \tag{3.27}$$

When inverted, unlike the other preconditioners we have considered, this maintains a term containing $\tilde{L}$, in addition to the $\tilde{L}^{-1}$ in the Schur complement approximation inverse. Taking the same approximation to $\mathcal{S}$ as above, we obtain the inverse

$$\mathcal{P}_T^{-1} = \begin{bmatrix} D^{-1} & 0 & -D^{-1}\tilde{L}\tilde{\mathcal{S}}^{-1} \\ 0 & \mathcal{R}^{-1} & -\mathcal{R}^{-1}\tilde{\mathcal{H}}\tilde{\mathcal{S}}^{-1} \\ 0 & 0 & \tilde{\mathcal{S}}^{-1} \end{bmatrix}. \tag{3.28}$$

In order to implement this preconditioner, (3.28) must be described in Kronecker form, approximating $\tilde{\mathcal{S}}^{-1}$ by truncation or as we use in Section 4.2, the Sylvester equation approach above.

3.4.3. *Analysis of preconditioners*

As mentioned above, whilst there has been investigation into preconditioning saddle point problems such as [5,6,8], most of these choices assume that the $(1, 1)$ block is the computationally expensive one.

Schur complement preconditioners such as the block diagonal and block triangular examples we consider here are detailed in [5,6]. Using exact matrices for the approximations $\tilde{\mathcal{S}}$, $\tilde{L}$ and $\tilde{\mathcal{H}}$, in (3.24) and (3.27) results in the preconditioned system having two or three eigenvalues; therefore methods such as MINRES or GMRES converge in at most three steps. However in general, we must consider approximations which reduces the efficacy of the preconditioner. Furthermore, for the data assimilation saddle point problem, these are not necessarily the most appropriate from a computational point of view.

The use of the inexact constraint preconditioner [8] in the data assimilation setting is considered in [15,16,18], and experimentally has proved effective. Here as the covariance matrices are less computationally expensive, the exact $(1, 1)$ block is typically used. Thus using the result in [8], the eigenvalues τ of the matrix

$$\begin{bmatrix} D & 0 & \tilde{L} \\ 0 & \mathcal{R} & \tilde{H} \\ \tilde{L}^T & \tilde{H}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} D & 0 & L \\ 0 & \mathcal{R} & \mathcal{H} \\ L^T & \mathcal{H}^T & 0 \end{bmatrix} \tag{3.29}$$

are either one (with multiplicity at least $(N+1)(2n+p) - 2\,\text{rank}([L^T, \quad \mathcal{H}^T] - [\tilde{L}^T, \quad \tilde{\mathcal{H}}^T])$) or bounded by

$$|\tau - 1| \leq \frac{\|[L^T, \quad \mathcal{H}^T] - [\tilde{L}^T, \quad \tilde{\mathcal{H}}^T]\|}{\tilde{\sigma}_1},$$

where $\tilde{\sigma}_1$ is the smallest singular value of $[\tilde{L}^T, \quad \tilde{\mathcal{H}}^T]$.

When considering the exact approximation $\tilde{L} = L$, and taking $\tilde{\mathcal{H}} = 0$, the resulting preconditioned system has eigenvalues

$$\tau = 1 \pm \sqrt{\frac{\mu^T \mathcal{H} L^{-1} D L^{-T} \mathcal{H}^T \mu}{\mu^T \mathcal{R} \mu}}\, i$$

where $\mu \in \mathbb{R}^{(N+1)p}$. Using the properties of the Rayleigh quotient, we know that the eigenvalues are on a line parallel to the imaginary axis through 1, where the maximum distance from the real axis is given by

$$\sqrt{\frac{\lambda_{\max}(\mathcal{H}L^{-1}DL^{-T}\mathcal{H}^T)}{\lambda_{\min}(\mathcal{R})}}.$$

Experimental results in [18] demonstrate that when an approximation is taken for $\tilde{L}$, the eigenvalues are clustered in a cloud surrounding $\tau = 1$ with the size of this cloud likely depending on the accuracy of the chosen approximation.

4. Numerical results

In this section we present numerical results using LR-GMRES. (For preconditioning strategies see Section 4.2). We use 20 iterations of LR-GMRES with a tolerance of 10^{-6}. During the algorithm where we truncate the matrices after concatenation and applying `Amult`, we use a truncation tolerance of 10^{-8}. We present examples with different choices of reduced rank r.

4.1. One-dimensional advection–diffusion system

As a first example, let us consider the one-dimensional (linear) advection–diffusion problem, defined as:

$$\frac{\partial}{\partial t}u(x,t) = c_d\frac{\partial^2}{\partial x^2}u(x,t) + c_a\frac{\partial}{\partial x}u(x,t) \tag{4.1}$$

for $x \in [0,1]$, $t \in (0,T)$, subject to the boundary and initial conditions

$$\begin{aligned} u(0,t) &= 0, && t \in (0,T) \\ u(1,t) &= 0, && t \in (0,T) \\ u(x,0) &= u_0(x), && x \in [0,1]. \end{aligned}$$

We solve this system with a centered difference scheme for u_x and u_t, and a Crank–Nicolson scheme [12] for u_{xx}, discretising x uniformly with $n = 100$, and taking timesteps of size $\Delta t = 10^{-3}$. For this example, we set the underlying system to have $c_d = 0.1$, $c_a = 1.4$ and $u_0(x) = \sin(\pi x)$.

We now consider this example as a data assimilation problem, and compare the solutions obtained both by the saddle point formulation (2.16), and the low-rank approximation using LR-GMRES. We take an assimilation window of 200 timesteps (giving $N = 199$), followed by a forecast of 800 timesteps. Thus the resulting linear system (2.16) we solve here is of size $(40{,}000 + 200p)$, where p is the number of observations we take at each timestep. Independent of p, the full-rank update $\delta x \in \mathbb{R}^{20{,}000}$. In contrast the low-rank update is WV^T, where $W \in \mathbb{R}^{100\times r}$, $V \in \mathbb{R}^{200\times r}$. For $r = 20$, this requires only 30% of the storage.

In the examples to follow, we compare the full- and low-rank solutions to the data assimilation problem with the background estimate.

Perfect observations First let us suppose we have perfect observations of every state in the assimilation window. Hence $p = 100$, and the size of the saddle point system we consider is 60,000. We take as the background estimate u_0^b, a perturbed initial condition with background covariance $B = 0.1I_{100}$, and for this, and the following examples, we consider a model error with covariance $Q = 10^{-4}I_{100}$.

Fig. 4.1 shows the state $u(x,t_a)$ and absolute error $|u^*(x,t_a) - u(x,t_a)|$ for the time t_a immediately after assimilation. We consider the three approaches, denoting the true solution by u^*. In Fig. 4.2 we consider the root mean squared error of the approaches, presenting the errors in both the assimilation window, and the forecast. The results show that the low-rank solution matches the full-rank solution very closely, in both the observation window and the forecast. In Fig. 4.1, the low, and full-rank approximations are indistinguishable, with both displaying the same characteristics in the state error plot. Both methods for solving the data assimilation problem result in a superior forecast to the initial guess (without assimilation).

It is worth noting that here the low-rank solution to the data assimilation problem achieves a lower root mean squared error than the full-rank solution for half of the forecast window. Investigating different random seeds, we saw that this was not always the case, though in majority of experiments the two solutions were close. In this example, the full- and low-rank solutions both outperformed the background estimate for all random seeds considered.

Partial, noisy observations Next, we introduce partial noisy observations, taking observations in every fifth component of u. These are generated from the truth with covariance $R = 0.01I_p$, for $p = 20$, and as such the linear system we consider for this example is of size 44,000. In this example we take for the background error covariance $B_{i,j} = 0.1\exp(\frac{-|i-j|}{50})$, keeping $Q = 10^{-4}I_{100}$ and $r = 20$. The resulting errors for three approaches, and the root mean squared errors are shown in Fig. 4.3.

As with the previous example, the state errors of both the full- and low-rank solutions are similar, though here we notice a greater variation between the two than in the previous example. Unlike above, when we compare the root mean squared

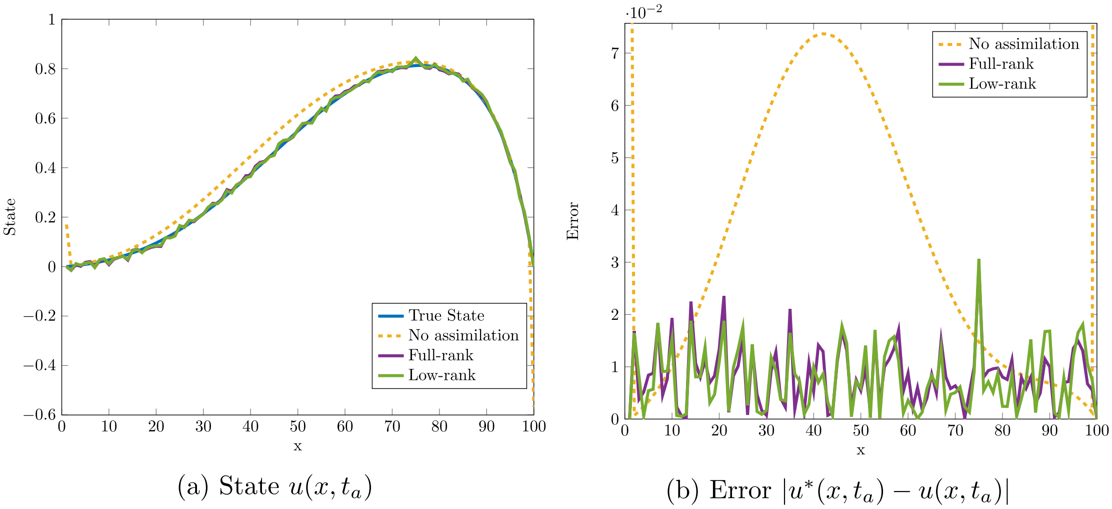


Fig. 4.1. State and error for time t_a after the assimilation window for 1D advection–diffusion problem with perfect observations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

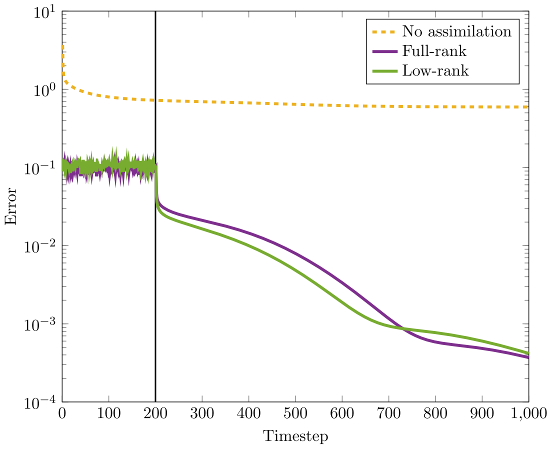


Fig. 4.2. Root mean squared errors for 1D advection–diffusion data assimilation problem with perfect observations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

errors of the full- and low-rank approaches, there is a greater disparity between the two, with the full-rank performing significantly better except at the very start of the forecast. Nonetheless the low-rank approximation is superior to using no assimilation.

Different choices of rank We now consider the effect of the chosen rank on the assimilation result. In the previous examples we have considered $r = 20$, which resulted in the low-rank approximation to δx requiring only 30% of the storage needed for the full-rank solution. Here we consider $r = 5$ (requiring 7.5% of the storage), and $r = 1$ (needing just 1.5%), and otherwise keep the setup of the example used in Fig. 4.3, with partial, noisy observations unchanged. In Fig. 4.4 we obtain a very close forecast from taking $r = 5$ to that which we saw from $r = 20$, though the assimilation window has greater variation for $r = 5$ whilst remaining close to the full-rank solution. In contrast, the behaviour of the root mean squared error for $r = 1$ is considerably different to that of the full-rank solution. Despite this, the forecasts for both $r = 5$ and $r = 1$ are close to the full-rank solution and are comfortably more accurate than using no assimilation. The closeness to the full-rank may be caused by the smoothing properties of this model operator, and the particular random seed, as noted above. Though taking different random seeds results in similar behaviour in majority of cases.

Table 1 presents the storage requirements for the examples considered in this section. As Figs. 4.1–4.4 demonstrate, despite the large reduction in the necessary storage for the low-rank approach, it results in close approximations to the full-rank method.

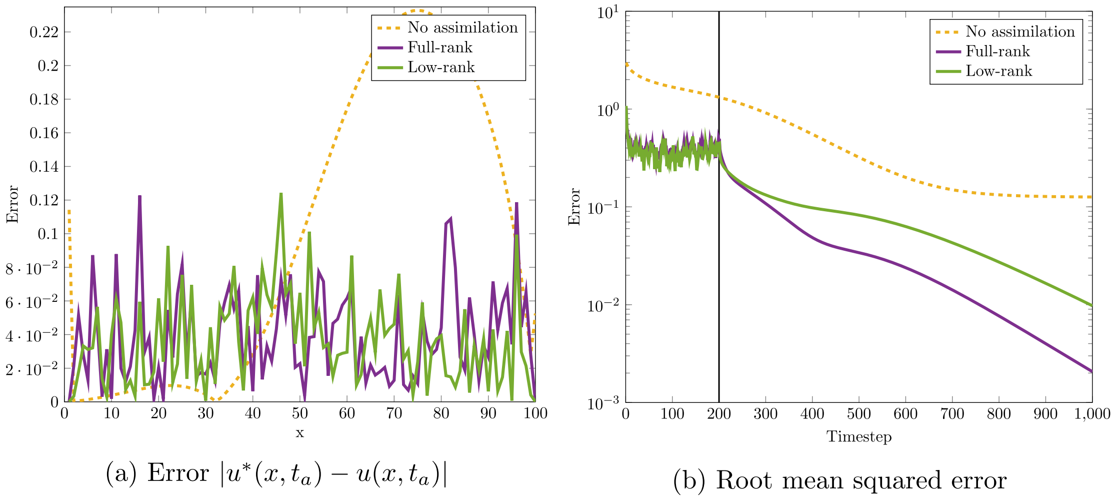


Fig. 4.3. Error for time t_a after the assimilation window, and root mean squared error for 1D advection–diffusion problem with partial, noisy observations ($r = 20$). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

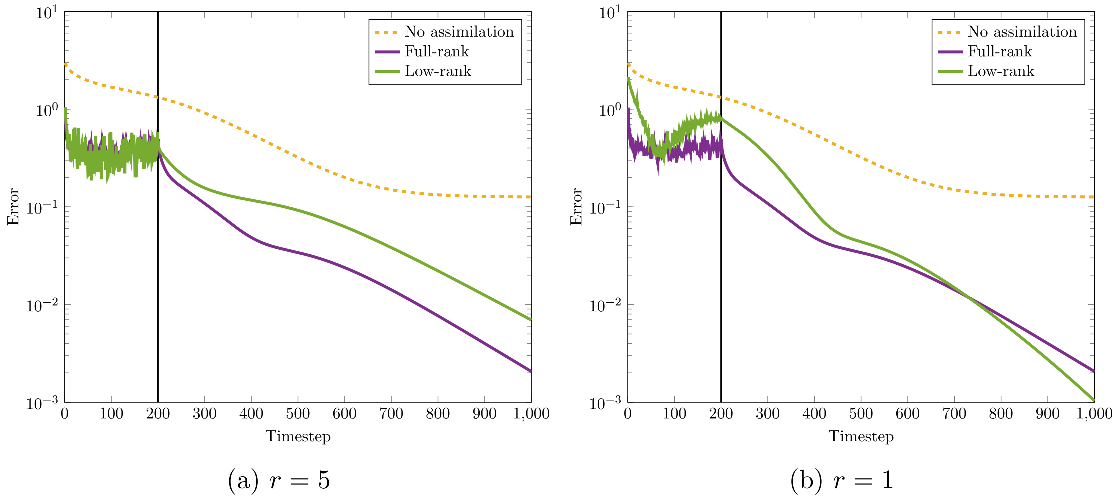


Fig. 4.4. Root mean squared errors for 1D advection–diffusion data assimilation problem with partial, noisy observations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Storage requirements for full- and low-rank methods in the 1D advection–diffusion equation examples.

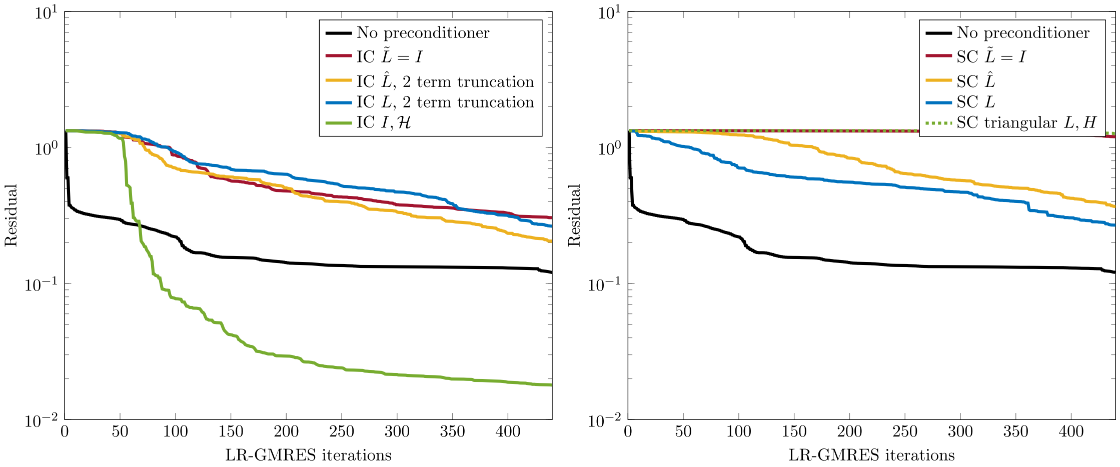
n	N	p	rank	# of matrix elements in solution		storage reduction
				full-rank	low-rank	
100	199	100	20	20,000	6,000	70%
100	199	20	20	20,000	6,000	70%
100	199	20	5	20,000	1,500	92.5%
100	199	20	1	20,000	300	98.5%

Computation time In Table 2, we present a comparison of the computation time for different choices of rank in the advection–diffusion example using LR-GMRES. As above, we perform twenty iterations of LR-GMRES, and average over one hundred runs. These computations were done on an Intel i5-4460 processor operating at 3.2GHz.

We note that due to the truncation steps in the LR-GMRES algorithm, which are currently performed using a (sparse) `svd`, we do not see significant savings in time compared to solving the saddle point system using Matlab's backslash function because of this expense. However it is possible that in larger problem sizes, with a low choice of rank, we may see superior time savings.

Table 2
Comparison of computation time for low-rank GMRES in the 1D advection–diffusion equation examples.

n	N	p	rank	saddle point size	runtime (s)
100	199	20	99	44,000	21.6881
100	199	20	50	44,000	9.4815
100	199	20	20	44,000	2.7177
100	199	20	5	44,000	0.7075
100	199	20	1	44,000	0.4440



(a) Inexact constraint (b) Schur complement

Fig. 4.5. Residual using different preconditioners for the 440 × 440 advection–diffusion example. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.2. Comparison of preconditioners

We present here a comparison between different choices of preconditioner for the 1D advection–diffusion equation system in Section 4.1. We consider a small example taking $n = 10$, $N = 19$, $p = 4$ with $B_{i,j} = 0.1\exp(\frac{-|i-j|}{50})$, $Q = 10^{-4} I_{10}$, $R = 0.01 I_4$. The resulting saddle point matrix is 440 × 440. In all the following cases a reduced rank size of $r = 5$ is considered, though similar results are obtained when we vary this choice.

The preconditioners considered in Fig. 4.5a are inexact constraint preconditioners (3.17), which we compare to using no preconditioner. We use $\tilde{L} = I$, $\hat{L} = I_{N+1} \otimes I_n + C \otimes I_n$, and also consider $\mathcal{P}_{I\mathcal{H}}$ from (3.22) where $\tilde{L} = I$, and use the exact $\mathcal{H}$.

We see that none of the preconditioners achieve a residual smaller than 10^{-2} even after 440 iterations due to the additional restrictions of the low-rank solver (e.g. the truncation during the algorithm). The three inexact constraint preconditioners where we take $\tilde{\mathcal{H}} = 0$ exhibit very similar behaviour with the approximation $\hat{L}$ performing slightly better than the other two on the whole. The only preconditioner which achieved superior results to taking the identity, was $\mathcal{P}_{I\mathcal{H}}$ from (3.22), incorporating the true $\mathcal{H}$ and taking $L = I$. Despite this, the improvement occurs only after 70 iterations which for GMRES is not ideal since we must store all iterates. Even using the low-rank representation here, this becomes problematic.

For Fig. 4.5b, we experimented with a selection of Schur complement preconditioners, all of which approximate the Schur complement using the approximation (3.25). For the block triangular preconditioner, we use the exact L and $\mathcal{H}$ in the inverted matrix in addition to (3.25).

Unlike the inexact constraint preconditioners, none of the Schur complement preconditioners we consider here showed better results than using no preconditioner. Comparison with the inexact constraint preconditioners shows the block diagonal Schur complement preconditioners using $\hat{L}$ and L to be comparable. Despite the block triangular preconditioner containing the true $\mathcal{H}$ it results in an ineffective choice, performing worse than all others considered.

To illustrate a larger problem size than those above, we conduct a further test using $n = 20$ with the remaining setup unchanged from above. Thus the saddle point matrix is now of size 880. In Fig. 4.6 we compare the best performing of the above preconditioners, the inexact constraint preconditioner $\mathcal{P}_{I\mathcal{H}}$ from (3.22) using $\tilde{L} = I$ and $\tilde{\mathcal{H}} = \mathcal{H}$. We see that as before, the inexact constraint preconditioner eventually results in a lower residual, though here it takes over 250 iterations, nearly four times as many as in the 440 system which was merely half the size. As mentioned above this is infeasible for this implementation of LR-GMRES, and hence we used no preconditioner in the numerical examples presented in Sections 4.1 and 5.1.

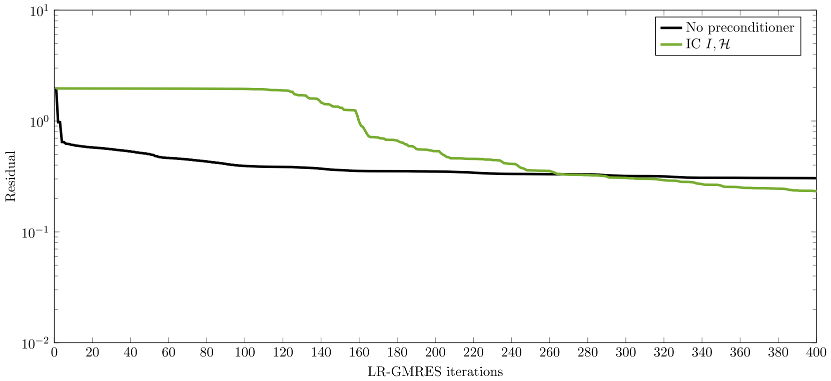


Fig. 4.6. Residual using the inexact constraint preconditioner for the 880 × 880 advection–diffusion example. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

A possible explanation for why preconditioning is not effective here is the following. During LR-GMRES, the truncation process selects only the most important modes, e.g. the ones belonging to larger eigenvalues, ignoring the smaller ones. Therefore, the low-rank approach acts like a regularisation, and hence in some sense like a projected preconditioner.

5. Time-dependent systems

Next we consider an extension of the Kronecker formulation (3.2) to the time-dependent case, allowing for time-dependent model, and observation operators, and the respective covariance matrices. The remaining assumption we must make is that the number of observations in the i-th timestep, p_i is constant, i.e. $p_i = p$ for each i. With these assumptions, the linear system in (3.2) becomes

$$\begin{bmatrix} F_1 \otimes B + \sum_{i=1}^{N} F_{i+1} \otimes Q_i & 0 & I \otimes I_x + \sum_{i=1}^{N} C_i \otimes M_i \\ 0 & \sum_{i=0}^{N} F_{i+1} \otimes R_i & \sum_{i=0}^{N} F_{i+1} \otimes H_i \\ I \otimes I_x + \sum_{i=1}^{N} C_i^T \otimes M_i^T & \sum_{i=0}^{N} F_{i+1} \otimes H_i^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \\ \delta x \end{bmatrix} = \begin{bmatrix} b \\ d \\ 0 \end{bmatrix}, \tag{5.1}$$

where F_i denotes the matrix with 1 on the ith entry of the diagonal, and zeros elsewhere, and C_i is the matrix with -1 on the ith column of the subdiagonal, and zeros elsewhere. Here M_i and H_i are linearisations of the model and observation operators $\mathcal{M}_i$ and $\mathcal{H}_i$ respectively about x_i.

As in Section 3.1, we may use (3.1) to rewrite this as the (now more general) matrix equations

$$\begin{aligned} B\Lambda F_1 + \sum_{i=1}^{N} Q_i \Lambda F_{i+1} + X + \sum_{i=1}^{N} M_i X C_i^T &= \mathbb{b} \\ \sum_{i=0}^{N} R_i U F_{i+1} + \sum_{i=0}^{N} H_i X F_{i+1} &= \mathbb{d} \\ \Lambda + \sum_{i=1}^{N} M_i^T \Lambda C_i + \sum_{i=0}^{N} H_i^T U F_{i+1} &= 0. \end{aligned} \tag{5.2}$$

Where as before $\lambda, \delta x, b, \mu$ and d are vectorised forms of the matrices $\Lambda, X, \mathbb{b} \in \mathbb{R}^{n \times N+1}$ and $U, \mathbb{d} \in \mathbb{R}^{p \times N+1}$ respectively. These matrix equations must again be solved for Λ, U and X, where X is the matrix of interest.

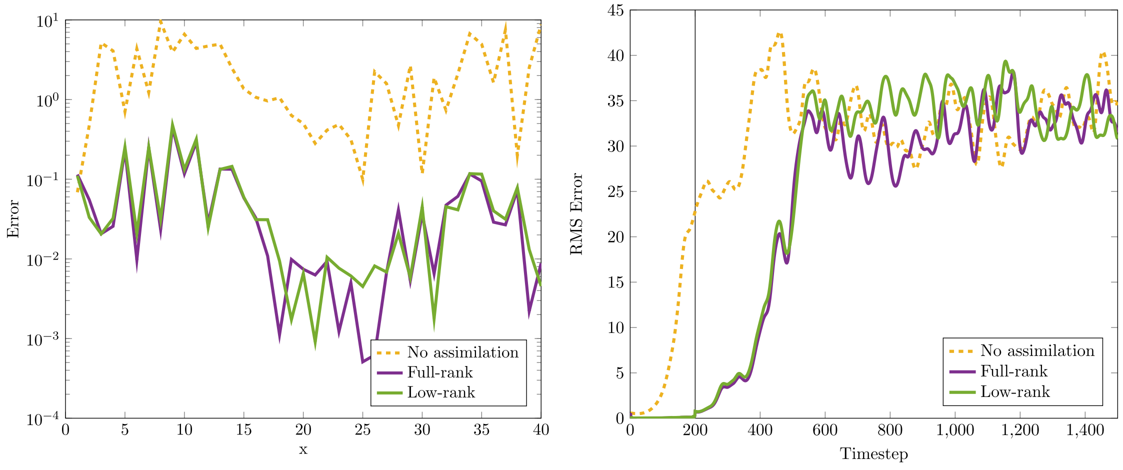
Algorithm 4 is an implementation of `Amult` for the time-dependent case, explicitly writing the concatenation defined by (5.2) in the form required for LR-GMRES. This requires linearisations of the model and observation operators at all timesteps in order to be applied.

We note that further to the truncation expense highlighted in Section 4, the significantly increased number of matrices being concatenated prior to truncation results in longer runtimes, particularly if new linearised matrices must be computed.

As an example, we consider the Lorenz-95 system [28] which is both non-linear, and also chaotic rather than smoothing such as the previous example (Section 4.1), so as to better represent real world data assimilation problems such as weather forecasting.

Algorithm 4 Matrix multiplication (time-dependent) (`Amult`).

```
Input: $W_{11}, W_{12}, W_{21}, W_{22}, W_{31}, W_{32}$
Output: $Z_{11}, Z_{12}, Z_{21}, Z_{22}, Z_{31}, Z_{32}$
  $Z_{11} = [BW_{11}, \quad Q_1 W_{11}, \quad \dots, \quad Q_N W_{11}, \quad W_{31}, \quad M_1 W_{31}, \quad \dots, \quad M_N W_{31}],$
  $Z_{12} = [F_1 W_{12}, \quad F_2 W_{12}, \quad \dots, \quad F_{N+1} W_{12}, \quad W_{32}, \quad C_1 W_{32}, \quad \dots, \quad C_N W_{32}],$
  $Z_{21} = [R_0 W_{21}, \quad \dots, \quad R_N W_{21}, \quad H_0 W_{31}, \quad \dots, \quad H_N W_{31}],$
  $Z_{22} = [F_1 W_{22}, \quad \dots, \quad F_{N+1} W_{22}, \quad F_1 W_{32}, \quad \dots, \quad F_{N+1} W_{32}],$
  $Z_{31} = [W_{11}, \quad M_1^T W_{11}, \quad \dots, \quad M_N^T W_{11}, \quad H_0^T W_{21}, \quad \dots, \quad H_N^T W_{21}],$
  $Z_{32} = [W_{12}, \quad C_1^T W_{12}, \quad \dots, \quad C_N^T W_{12}, \quad F_1 W_{22}, \quad \dots, \quad F_{N+1} W_{22}]$
```



(a) Error $|x^* - x|$

(b) Root mean squared error

Fig. 5.1. Error $|x^* - x|$ for the time after the assimilation window, and root mean squared error for Lorenz-95 system with perfect observations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.1. Lorenz-95 system

We consider the Lorenz-95 system [28], this is a generalisation of the three-dimensional Lorenz system [27] to n dimensions. The model is defined by a system of n non-linear ordinary differential equations

$$\frac{\mathrm{d}x^i}{\mathrm{d}t} = -x^{i-2}x^{i-1} + x^{i-1}x^{i+1} - x^i + f, \tag{5.3}$$

where $x = [x^1, x^2, \dots, x^n]^T$ is the state of the system, and f is a forcing term. It is known that for $f = 8$, the Lorenz system exhibits chaotic behaviour [20,28]. Also noted is that for reasonably large values of n (here we take $n = 40$), this choice of f leads to a model which is comparable to weather forecasting models.

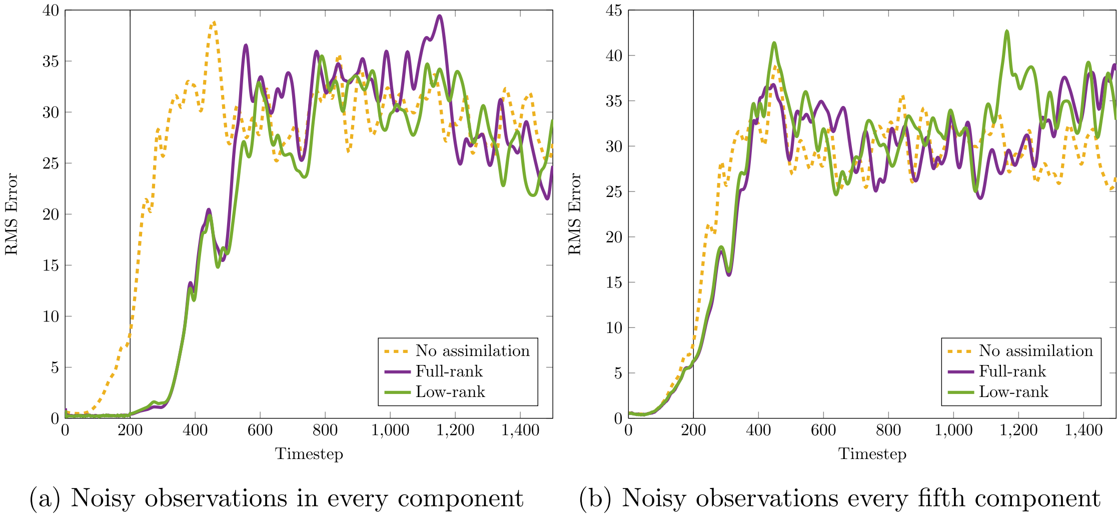
We solve (5.3) using a 4th order Runge–Kutta method in order to obtain

$$x_{k+1} = \mathcal{M}_k(x_k), \quad \text{where } x_k = [x_k^1, x_k^2, \dots, x_k^n]^T, \tag{5.4}$$

where $\mathcal{M}_k$ is the non-linear model operator which evolves the state x_k to x_{k+1}. As before $\mathcal{H}_k$ denotes the potentially non-linear observation operator for the state x_k. To formulate the data assimilation problem as a saddle point problem, we generate the tangent linear model, and observation operators M_k and H_k by linearising $\mathcal{M}_k$ and $\mathcal{H}_k$ about x_k.

As in Section 4.1, we compare the low-rank approximation computed using LR-GMRES, to the full-rank solution of the saddle point formulation (2.16), and the background estimate (e.g. no assimilation). We perform the data assimilation using an assimilation window of 200 timesteps, followed by a forecast of 1300 timesteps, where the timesteps are of size $\Delta t = 5 \cdot 10^{-3}$. The full-rank update is therefore $\delta x \in \mathbb{R}^{8,000}$, whilst in contrast the low-rank update WV^T, is such that $W \in \mathbb{R}^{40\times r}$, $V \in \mathbb{R}^{200\times r}$. Here we consider $r = 20$ once more, which here requires 60% of the storage, still demonstrating a significant reduction.

Perfect observations As with the advection–diffusion equation, let us first suppose we have perfect observations of every state in the assimilation window, we take as the background estimate x_0^b, a perturbed initial condition with background covariance $B = 0.1I_{40}$, and as before, we consider a model error with covariance $Q = 10^{-4}I_{40}$. The error $|x^* - x|$ for the time after assimilation, and the root mean square errors for the three approaches in this example are presented in Fig. 5.1. The choice of $r = 20$ here results in a low-rank approximation which is very close to the full-rank solution. This is very



(a) Noisy observations in every component (b) Noisy observations every fifth component

Fig. 5.2. Root mean squared error for Lorenz-95 system with noisy, and partial observations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

good given that the low-rank approximation requires 40% less storage. In the state error plot we observe small differences between solutions for the middle states, though this is still substantially smaller than the error with no assimilation. In the forecast the low-rank approximation matches the full-rank until both reach the error with no assimilation, with only small variation.

Noisy observations We next introduce noisy observations, taking $R = 0.01I_p$ for the observation error covariance, furthermore we take as the background error covariance $B_{i,j} = 0.1\exp(\frac{-|i-j|}{50})$. In Fig. 5.2 we consider the root mean squared errors for two different choices of observation operator: taking interpolatory observations in every component ($p = 40$) shown on the left, and in every fifth component ($p = 8$) on the right. In both cases, the low-rank approximation matches the full-rank very closely until the time at which both errors are comparable to the background estimate. In this example the assimilation of noisy observations in every fifth component is similarly difficult for both approaches. To achieve these very similar results using the low-rank approach, despite using just 60% of the storage is very promising.

150-dimensional Lorenz-95 Finally, we consider as a larger example, the 150-dimensional Lorenz-95 system with an assimilation window of 150 timesteps. This gives a full-rank update $\delta x \in \mathbb{R}^{22,500}$, and we consider two different choices of low-rank, $r = 20$ requiring 27% of the storage, and $r = 5$ needing 7%. In this example we take noisy observations in each state, with covariances $B_{i,j} = 0.1\exp(\frac{-|i-j|}{50})$, $R = 0.01I_{150}$ and $Q = 10^{-4}I_{150}$.

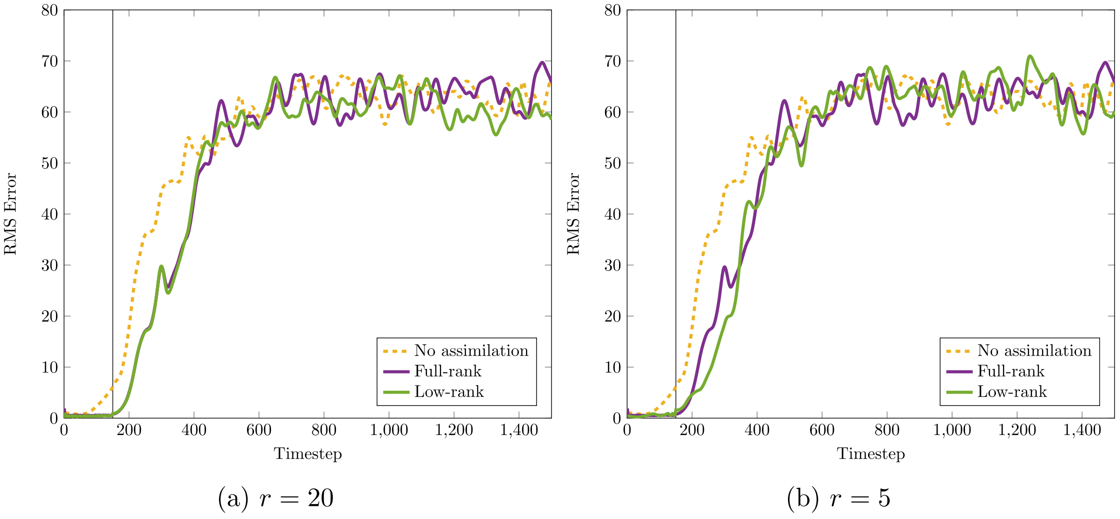
These examples, shown in Fig. 5.3 demonstrate further that a low-rank approximation performs very closely to that of the full-rank solution for small choices of r. Taking $r = 20$ we see that as in the previous examples, the resulting approximation is nearly indistinguishable until both solutions reach the same level of error as with no assimilation. As before, we see the low-rank performing better for $r = 5$, this is not always the case depending on the random seed as noted earlier, and is emphasised by the chaotic system sensitivity. However repeated experimentation shows that the full- and low-rank approximations are often close. Here the approximation using $r = 5$ gives similar results to the full-rank approximation, despite requiring just 7% of the storage.

Table 3 presents the storage requirements for the examples considered in this section. As with the advection–diffusion example, despite the large reduction in storage required, the experiments have shown that the low-rank approximations give similar results to the full-rank approach, which is a very good prospect.

6. Conclusions

The saddle point formulation of weak constraint four-dimensional variational data assimilation results in a large linear system which in the incremental approach is solved to determine the update δx at every step. In this paper we have proposed a low-rank approach which approximates the solution to the saddle point system, with significant reductions in the storage needed. This was achieved by considering the structure of this saddle point system and using techniques from the theory of matrix equations. Using the existence of low-rank solutions to Sylvester equations we showed that low-rank solutions to the data assimilation problem exist under certain assumptions, with numerical experimentation demonstrating that this may be the case even when these assumptions are relaxed.

We introduced a low-rank GMRES solver, considered the requirements for implementing this algorithm, and investigated several preconditioning approaches. For our examples we observed that no preconditioners were necessary, however further



(a) $r = 20$ (b) $r = 5$

Fig. 5.3. Root mean squared error for 150-dimensional Lorenz-95 system with $r = 20$ and $r = 5$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 3
Storage requirements for full- and low-rank methods in the Lorenz-95 examples.

n	N	p	rank	# of matrix elements in solution		storage reduction
				full-rank	low-rank	
40	199	40	20	8,000	4,800	40%
40	199	8	20	8,000	4,800	40%
150	149	150	20	22,500	6,000	73.3%
150	149	150	5	22,500	1,500	93.3%

investigation of this may lead to new choices of preconditioners for the data assimilation setting, and new low-rank solvers for weak constraint 4D-Var.

Numerical experiments have demonstrated that the low-rank approach introduced here is successful using both linear and non-linear models.

In these examples we achieved close approximations to the full-rank solutions with storage requirements as low as 10% of those needed by the full-rank approach. We see that reducing the rank additionally results in a larger time saving, however due to the superiority of Matlab's '\', we do not achieve faster results than a sophisticated direct solver for these problems. It is possible that with larger problem sizes, we may achieve greater time savings. These results are very promising, though some further investigation is needed, in particular for non-linear problems.

References

[1] P. Benner, T. Breiten, Low rank methods for a class of generalized Lyapunov equations and related issues, Numer. Math. 124 (2013) 441–470.
[2] P. Benner, P. Kürschner, Computing real low-rank solutions of Sylvester equations by the factored ADI method, Comput. Math. Appl. 67 (2014) 1656–1672.
[3] P. Benner, J.-R. Li, T. Penzl, Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems, Numer. Linear Algebra Appl. 15 (2008) 755–777.
[4] P. Benner, R.-C. Li, N. Truhar, On the ADI method for Sylvester equations, J. Comput. Appl. Math. 233 (2009) 1035–1045.
[5] M. Benzi, G.H. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numer. 14 (2005) 1–137.
[6] M. Benzi, A.J. Wathen, Some Preconditioning Techniques for Saddle Point Problems, Springer-Verlag, 2008, pp. 195–211.
[7] L. Bergamaschi, On eigenvalue distribution of constraint-preconditioned symmetric saddle point matrices, Numer. Linear Algebra Appl. 19 (2011) 754–772.
[8] L. Bergamaschi, J. Gondzio, M. Venturin, G. Zilli, Inexact constraint preconditioners for linear systems arising in interior point methods, Comput. Optim. Appl. 36 (2007) 137–147.
[9] L. Bergamaschi, J. Gondzio, M. Venturin, G. Zilli, Erratum to: Inexact constraint preconditioners for linear systems arising in interior point methods, Comput. Optim. Appl. 49 (2009) 401–406.
[10] Y. Cao, J. Zhu, I.M. Navon, Z. Luo, A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition, Int. J. Numer. Methods Fluids 53 (2007) 1571–1583.
[11] P. Courtier, J.-N. Thépaut, A. Hollingsworth, A strategy for operational implementation of 4D-var, using an incremental approach, Q. J. R. Meteorol. Soc. 120 (1994) 1367–1387.
[12] J. Crank, P. Nicolson, A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of the Heat-Conduction Type, Math. Proc. Camb. Philos. Soc., vol. 43, Cambridge Univ. Press, 1947, pp. 50–67.
[13] V. Druskin, V. Simoncini, Adaptive rational Krylov subspaces for large-scale dynamical systems, Syst. Control Lett. 60 (2011) 546–560.

[14] G. Evensen, Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, J. Geophys. Res. 99 (1994) 10143–10162.

[15] M. Fisher, S. Gratton, S. Gürol, Y. Trémolet, X. Vasseur, Low rank updates in preconditioning the saddle point systems arising from data assimilation problems, Optim. Methods Softw. 33 (2018) 45–69.

[16] M. Fisher, S. Gürol, Parallelisation in the time dimension of four-dimensional variational data assimilation, Q. J. R. Meteorol. Soc. (2017), https://doi.org/10.1002/qj.2997.

[17] M. Fisher, M. Leutbecher, G.A. Kelly, On the equivalence between Kalman smoothing and weak-constraint four-dimensional variational data assimilation, Q. J. R. Meteorol. Soc. 131 (2005) 3235–3246.

[18] M. Fisher, Y. Trémolet, H. Auvinen, D. Tan, P. Poli, Weak-Constraint and Long-Window 4D-Var, Tech. Rep. 655, ECMWF, 2011.

[19] G.M. Flagg, S. Gugercin, On the ADI method for the Sylvester equation and the optimal-$\mathcal{H}_2$ points, Appl. Numer. Math. 64 (2013) 50–58.

[20] M.A. Freitag, R. Potthast, Synergy of Inverse Problems and Data Assimilation Techniques, Radon Ser. Comput. Appl. Math., vol. 13, Walter de Gruyter, 2013, pp. 1–53.

[21] L. Grasedyck, Existence of a low rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation, Numer. Linear Algebra Appl. 11 (2004) 371–389.

[22] R.E. Kalman, A new approach to linear filtering and prediction problems, J. Basic Eng. 82 (1960) 35–45.

[23] D. Kressner, C. Tobler, Krylov subspace methods for linear systems with tensor product structure, SIAM J. Matrix Anal. Appl. 31 (2010) 1688–1714.

[24] A.S. Lawless, Variational Data Assimilation for Very Large Environmental Problems, Radon Ser. Comput. Appl. Math., vol. 13, Walter de Gruyter, 2013, pp. 55–90.

[25] A.S. Lawless, N.K. Nichols, C. Boess, A. Bunse-Gerstner, Using model reduction methods within incremental four-dimensional variational data assimilation, Mon. Weather Rev. 136 (2008) 1511–1522.

[26] J.-R. Li, J. White, Low-rank solution of Lyapunov equations, SIAM J. Matrix Anal. Appl. 24 (2002) 260–280.

[27] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (1963) 130–141.

[28] E.N. Lorenz, Predictability: a problem partly solved, in: Proc. Seminar on Predictability, vol. 1, 1996.

[29] B.C. Moore, Principal component analysis in linear systems: controllability, observability, and model reduction, IEEE Trans. Autom. Control 26 (1981) 17–32.

[30] T. Penzl, A cyclic low-rank Smith method for large sparse Lyapunov equations, SIAM J. Sci. Comput. 21 (1999) 1401–1418.

[31] D.T. Pham, J. Verron, M.C. Roubaud, A singular evolutive extended Kalman filter for data assimilation in oceanography, J. Mar. Syst. 16 (1998) 323–340.

[32] Y. Saad, Numerical solution of large Lyapunov equations, in: Signal Processing, Scattering and Operator Theory, and Numerical Methods, Proc. MTNS-89, Birkhäuser, 1990, pp. 503–511.

[33] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Comput. 7 (1986) 856–869.

[34] Y. Sasaki, An objective analysis based on the variational method, J. Meteorol. Soc. Jpn. 36 (1958) 77–88.

[35] Y. Sasaki, Some basic formalisms in numerical variational analysis, Mon. Weather Rev. 98 (1970) 875–883.

[36] V. Simoncini, A new iterative method for solving large-scale Lyapunov matrix equations, SIAM J. Sci. Comput. 29 (2007) 1268–1288.

[37] V. Simoncini, Computational methods for linear matrix equations, SIAM Rev. 58 (2016) 377–441.

[38] M. Stoll, T. Breiten, A low-rank in time approach to PDE-constrained optimization, SIAM J. Sci. Comput. 37 (2015) B1–B29.

[39] M. Verlaan, A.W. Heemink, Tidal flow forecasting using reduced rank square root filters, Stoch. Hydrol. Hydraul. 11 (1997) 349–368.