Geosci. Model Dev., 6, 583–590, 2013
www.geosci-model-dev.net/6/583/2013/
doi:10.5194/gmd-6-583-2013

Geoscientific Model Development
Open Access

Improving computational efficiency in large linear inverse problems: an example from carbon dioxide flux estimation

V. Yadav and A. M. Michalak

Department of Global Ecology, Carnegie Institution for Science, Stanford, California, 94305, USA

Correspondence to: V. Yadav (vineety@stanford.edu)

Received: 7 September 2012 – Published in Geosci. Model Dev. Discuss.: 19 October 2012
Revised: 7 March 2013 – Accepted: 26 March 2013 – Published: 3 May 2013

Abstract. Addressing a variety of questions within Earth science disciplines entails the inference of the spatiotemporal distribution of parameters of interest based on observations of related quantities. Such estimation problems often represent inverse problems that are formulated as linear optimization problems. Computational limitations arise when the number of observations and/or the size of the discretized state space becomes large, especially if the inverse problem is formulated in a probabilistic framework and therefore aims to assess the uncertainty associated with the estimates. This work proposes two approaches to lower the computational costs and memory requirements for large linear space–time inverse problems, taking the Bayesian approach for estimating carbon dioxide (CO_2) emissions and uptake (a.k.a. fluxes) as a prototypical example. The first algorithm can be used to efficiently multiply two matrices, as long as one can be expressed as a Kronecker product of two smaller matrices, a condition that is typical when multiplying a sensitivity matrix by a covariance matrix in the solution of inverse problems. The second algorithm can be used to compute a posteriori uncertainties directly at aggregated spatiotemporal scales, which are the scales of most interest in many inverse problems. Both algorithms have significantly lower memory requirements and computational complexity relative to direct computation of the same quantities ($O(n^{2.5})$ vs. $O(n^3)$). For an examined benchmark problem, the two algorithms yielded massive savings in floating point operations relative to direct computation of the same quantities. Sample computer codes are provided for assessing the computational and memory efficiency of the proposed algorithms for matrices of different dimensions.

1 Introduction

Addressing a variety of questions within Earth science disciplines including environmental science, hydrology, geology, geophysics, and biogeochemistry entails the inference of the spatiotemporal distribution of parameters of interest based on observations of related quantities. Such estimation problems often represent inverse problems, with examples including the estimation of hydraulic conductivity or other aspects of subsurface structure using geophysical (e.g., Aster et al., 2013) or hydrologic (e.g., Hyndman et al., 2007) observations, the identification of environmental contaminant sources using downstream concentrations (e.g., Atmadja and Bagtzoglou, 2001; Liu and Zhai, 2007; Michalak and Kitanidis, 2003; Zhang and Chen, 2007), the characterization of atmospheric and oceanic processes (Bennett, 2002), and the quantification of budgets of atmospheric trace gases using atmospheric observations (e.g., Ciais et al., 2011; Enting, 2002). Such inverse problems are often formulated as linear optimization problems. Even when the physics and/or chemistry relating the unobserved field to the measurements yields a nonlinear problem, the inverse problem is often solved through iterative application of a linear approximation (e.g., Kitanidis, 1995). Computational limitations arise when the number of observations n and/or the size of the discretized state space m becomes large, especially if the inverse problem is formulated in a probabilistic framework and therefore aims to assess the uncertainty associated with the estimates.

This work proposes approaches for addressing these computational limitations. We take the Bayesian approach for estimating carbon dioxide (CO_2) emissions and uptake (a.k.a. fluxes) as a prototypical example of a spatiotemporal

inverse problem, and use it to illustrate the proposed tools. We use the study of Gourdji et al. (2012) as a computational benchmark.

1.1 A prototypical spatiotemporal inverse problem

Gourdji et al. (2012) used atmospheric concentration measurements of CO_2 to constrain CO_2 fluxes in North America at a 1° longitude by 1° latitude spatial resolution ($m_s = 2635$ land regions for 50° W to 170° W and 10° N to 70° N) and a 3 hourly temporal frequency over the period of 24 December 2003 to 31 December 2004 ($m_\tau = 2992$ periods over 374 days). The implemented setup resulted in $n = 8503$ observations and $m = m_s \times m_\tau = 7\,883\,920$ parameters to be estimated. This high spatiotemporal resolution was primarily motivated by the desire to avoid "aggregation errors" (Engelen et al., 2002; Meirink et al., 2008, Thompson et al., 2011), i.e., biases in the estimated fluxes caused by prescribing spatial and temporal patterns within estimation regions and not allowing these patterns to be adjusted through the estimation. The resolutions that can be resolved by observations are often coarser, however, as are the scales that are of most scientific interest. In the case of Gourdji et al. (2012), the estimates were therefore aggregated a posteriori to monthly and annual temporal resolution for interpretation. The a priori spatiotemporal error covariance was assumed separable, with exponential decay in correlation in both space and time. As a result, the prior covariance matrix could be expressed as a Kronecker product of matrices describing the spatial and temporal covariances, respectively.

1.2 Bayesian framework for linear inverse problems

Stochastic linear inverse problems are often formulated in a Bayesian framework by requiring the minimization of an objective function that can be written as

$$L_s = \frac{1}{2}(\mathbf{z}-\mathbf{H}\mathbf{s})^T \mathbf{R}^{-1}(\mathbf{z}-\mathbf{H}\mathbf{s}) + \frac{1}{2}(\mathbf{s}-\mathbf{s}_p)^T \mathbf{Q}^{-1}(\mathbf{s}-\mathbf{s}_p) \quad (1)$$

where $z_{(n\times 1)}$ is a known vector of measurements, $\mathbf{H}_{(n\times m)}$ is a matrix that describes the relationship between measurements and the unknown field $\mathbf{s}_{(m\times 1)}$, $\mathbf{R}_{(n\times n)}$ is the covariance matrix of the model–data mismatch errors, $\mathbf{s}_{p(m\times 1)}$ is the prior estimate of $\mathbf{s}$, and $\mathbf{Q}_{(m\times m)}$ is a (square and symmetric) prior error covariance matrix, describing deviations between the true field $\mathbf{s}$ and the prior $\mathbf{s}_p$. The first term in Eq. (1) penalizes differences between available observations and those that would result from an estimated underlying field, while the second is a regularization term that penalizes departures from the prior, or more generally any type of desired structure.

The solution to the Bayesian linear inverse problem, defined as the estimate of $\boldsymbol{s}$ that minimizes the objective function in Eq. (1), can be expressed as

$$\hat{\mathbf{s}} = \mathbf{s}_p + (\mathbf{H}\mathbf{Q})^T (\mathbf{H}\mathbf{Q}\mathbf{H}^T + \mathbf{R})^{-1}(\mathbf{z}-\mathbf{H}\mathbf{s}_p) \quad (2)$$

and the a posteriori uncertainty covariance of the estimated $\hat{\mathbf{s}}$ can be written as

$$\mathbf{V}_{\hat{\mathbf{s}}} = \mathbf{Q} - (\mathbf{H}\mathbf{Q})^T (\mathbf{H}\mathbf{Q}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\mathbf{Q} \quad (3)$$

For small n and m, implementing Eqs. (2) and (3) is straightforward. As inverse problems are solved using increasingly more observations and are used to estimate parameters at increasingly high spatiotemporal resolutions, as in the prototypical Gourdji et al. (2012) example, the number of floating point operations required to implement these equations becomes prohibitive.

A closer look at Eqs. (2) and (3) shows that the first computational bottleneck occurs due to the cost of multiplying matrices $\mathbf{H}$ and $\mathbf{Q}$. The second is the cost of computing and storing a dense $\mathbf{V}_{\hat{\mathbf{s}}}$ with dimensions $m \times m$. Paradoxically, as noted previously, the scales of ultimate interest are often coarser than the native resolution of $\mathbf{V}_{\hat{\mathbf{s}}}$, and these covariances are frequently aggregated a posteriori in space and/or time by summing or averaging the corresponding entries in $\mathbf{V}_{\hat{\mathbf{s}}}$.

In this work, we propose a computational approach for evaluating $\mathbf{HQ}$, and by extension $\mathbf{HQH}^T$ for very large inverse problems, for the case where the covariance matrix $\mathbf{Q}$ can be expressed as a Kronecker product of two smaller matrices. This is typical of spatiotemporal inverse problems where the space–time covariance is assumed separable, or simpler problems that only consider covariance in space or in time, rather than both. While the fact that the covariance matrix $\mathbf{Q}$ can be expressed as a Kronecker product has been recognized in previous work (Meirink et al., 2008; Singh et al., 2011; Thompson et al., 2011), the novelty here is the use of this property in developing an algorithm for increasing the computational efficiency of the matrix product $\mathbf{HQ}$. We further present an approach for directly calculating the a posteriori error covariance at aggregated scales, without the intermediary step of first computing the full $\mathbf{V}_{\hat{\mathbf{s}}}$. Note that both of the presented algorithms rely on the availability of a pre-computed matrix $\mathbf{H}$, which can itself entail considerable computational cost that is not addressed in the work presented here. We use the Gourdji et al. (2012) problem as a computational benchmark for evaluating the performance of the proposed approaches relative to a direct implementation of Eqs. (2) and (3). Code demonstrating the implementation of both methods for a toy example is available as supplementary material.

2 Efficient method for the multiplication of any matrix with a matrix expressed as a Kronecker product

Reducing the computational cost of matrix multiplication and determining the minimum number of operations required to perform this operation has been an area of active research for several decades. The computational efficiency of matrix multiplication was generally assumed to be cubic until 1969

when Strassen (1969) showed that by performing more additions and fewer multiplications this cost could be reduced to $O(n^{2.82})$ (see Chapt. 47, p. 3 in Bini, 2010). This lower bound has since been further reduced, with the last minimum reported bound of $O(n^{2.37})$ obtained by Coppersmith and Winograd (1990). The majority of the existing algorithms (see Chapt. 47, p. 9 by Bini, 2010) rely on the property that matrix multiplication can be represented as multilinear maps (for details, see Bini and Lotti (1980) and Horn and Johnson (1990)), which helps in formulating a recursive non-commutative block decomposition based algorithms for performing matrix multiplication (Bini, 2010; Laderman, 1976). However the stability of these algorithms remains questionable, and other than Strassen's algorithm the proposed algorithms have not been found to be practically feasible (Bini and Lotti, 1980; Eve, 2009; Li et al., 2011).

The computational complexity of performing matrix multiplication of sparse and specially structured matrices such as Toeplitz, triangular Toeplitz, Hankel, Cauchy, Vandermonde, and Vandermonde transposed matrices (Mouilleron, 2011), circulant matrices (Nowak et al., 2003), banded matrices (Gohberg and Olshevsky, 1994), and hierarchical matrices (Saibaba and Kitanidis, 2012) with an arbitrary matrix can be made to be significantly lower (for comparison of computational cost, see page 15 in Mouilleron, 2011). As such, in the case of inverse problems, the structure of the covariance matrices can be used to reduce the cost of matrix multiplication. For the special case of a regular estimation grid and Toeplitz covariances, for example, the fast Fourier (FF) method of matrix multiplication gives an exact answer and has a computational complexity of $O(n^2 \log n)$ (for details, see Nowak et al., 2003); however the algorithm has high computational cost and memory requirements for sparse matrices. For an irregular estimation grid, an approximate method based on a hierarchical framework has been proposed by Saibaba and Kitanidis (2012). Like the FF method, this method also has a computational complexity of $O(n^2 \log n)$, and can only be used for very specific generalized covariance functions (for details, see Saibaba and Kitanidis, 2012).

Another type of structured covariance matrix is the one that can be expressed as the Kronecker product of two smaller matrices, such as in the case of **Q**. This property has been used in the past for decomposing square symmetric covariance matrices (Meirink et al., 2008; Thompson et al., 2011) and in some cases in increasing the efficiency of the matrix multiplication of separable covariance matrices with diagonal matrices (Singh et al., 2011). Here we build upon this work and propose an algorithm for matrix multiplication of two arbitrary matrices, one of which can be expressed as a Kronecker product, and show that its computational complexity is $O(n^{2.5})$. The algorithm has the same stability properties as those of cubic matrix multiplication algorithm.

2.1 Algorithm

Any matrix $\mathbf{B}_{(pr\times qt)}$ that can be expressed as a Kronecker product can be defined based on matrices $\mathbf{D}_{(p\times q)}$ and $\mathbf{E}_{(r\times t)}$ and denoted as $\mathbf{D}\otimes\mathbf{E}$, where

$$\mathbf{D}_{(p\times q)}\otimes\mathbf{E}_{(r\times t)} = \begin{pmatrix} d_{(1,1)}\mathbf{E} & \cdots & d_{(1,q)}\mathbf{E} \\ \vdots & \ddots & \vdots \\ d_{(p,1)}\mathbf{E} & \cdots & d_{(p,q)}\mathbf{E} \end{pmatrix}. \tag{4}$$

For a square covariance matrix **Q**, both **D** and **E** are also square. For the prototypical case examined here, **Q** is expressed as the Kronecker product of the temporal covariance and the spatial covariance, both of which decay exponentially with separation distance or lag:

$$\mathbf{Q} = \sigma_s^2 \overbrace{\left[\exp\left(-\frac{\mathbf{X}_\tau}{l_\tau}\right)\right]}^{\substack{\text{temporal}\\ \text{covariance}(\mathbf{D})}} \otimes \overbrace{\left[\exp\left(-\frac{\mathbf{X}_s}{l_s}\right)\right]}^{\substack{\text{spatial}\\ \text{covariance}(\mathbf{E})}}, \tag{5}$$

where σ_s^2 is the variance in space and time, $\mathbf{X}_s$ and $\mathbf{X}_\tau$ represent the separation distances/lags between estimation locations in space and time, respectively, and l_s and l_τ are the spatial and temporal correlation range parameters, respectively. In this case, $p = q = m_\tau$ and $r = t = m_s$. This defines a block matrix **Q** with m_τ^2 blocks, each defined as a square matrix $d_{(i,j)}\mathbf{E}$ with m_s^2 elements. As the Kronecker product is not commutative, the arrangement of the temporal and spatial covariance in Eq. (5) determines the design of **Q**.

Returning to the more generic case, the multiplication of any matrix $\mathbf{A}_{(n\times pr)}$ by $\mathbf{B}_{(pr\times qt)}$ proceeds as follows:

1. Divide **A** into p column blocks each with dimension $(n\times r)$

$$\mathbf{A}_{(n\times pr)} = \begin{pmatrix} \underbrace{\mathbf{a}_1}_{(n\times r)} & \underbrace{\mathbf{a}_2}_{(n\times r)} & \cdots & \underbrace{\mathbf{a}_p}_{(n\times r)} \end{pmatrix}. \tag{6}$$

2. Multiply each block of **A** by the elements of the first column of **D** and add these blocks $(\sum_{i=1}^{p} \mathbf{a}_i d_{(i,1)})$. If an element of **D** is zero, then skip the multiplication; if it is one, then add the column block of **A** without performing scalar multiplication.

3. Multiply the resulting $n \times r$ matrix by $\mathbf{E}_{(r\times t)}$ to obtain the first $n \times t$ column block of **AB**.

4. Repeat steps 2 and 3 for the remaining $q-1$ columns of **D** and the corresponding blocks of **AB**. Overall,

$$\mathbf{AB}_{(n\times qt)} = \begin{pmatrix} \underbrace{\left(\sum_{i=1}^{p} \mathbf{a}_i d_{(i,1)}\right)\mathbf{E}}_{(n\times t)} & \underbrace{\left(\sum_{i=1}^{p} \mathbf{a}_i d_{(i,2)}\right)\mathbf{E}}_{(n\times t)} & \cdots & \underbrace{\left(\sum_{i=1}^{p} \mathbf{a}_i d_{(i,q)}\right)\mathbf{E}}_{(n\times t)} \end{pmatrix}. \tag{7}$$

This algorithm can also be used for the multiplication of matrices where the first matrix is a Kronecker product of two smaller matrices, through the cyclical permutation property of transposes.

For $\mathbf{H}_{(n\times m)=(n\times m_\tau m_s)}$ and $\mathbf{Q}_{(m\times m)=(m_\tau m_s\times m_\tau m_s)}$, Eqs. (6) and (7) become

$$\mathbf{H}_{(n\times m_\tau m_s)} = \left(\underbrace{\mathbf{h}_1}_{(n\times m_s)} \underbrace{\mathbf{h}_2}_{(n\times m_s)} \cdots \underbrace{\mathbf{h}m_\tau}_{(n\times m_s)} \right) \tag{8}$$

$$\mathbf{HQ}_{(n\times m_\tau m_s)} = \left(\underbrace{\left(\sum_{i=1}^{m_\tau} \mathbf{h}_i d_{(i,1)}\right)\mathbf{E}}_{(n\times m_s)} \underbrace{\left(\sum_{i=1}^{m_\tau} \mathbf{h}_i d_{(i,2)}\right)\mathbf{E}}_{(n\times m_s)} \cdots \underbrace{\left(\sum_{i=1}^{m_\tau} \mathbf{h}_i d_{(i,m_\tau)}\right)\mathbf{E}}_{(n\times m_s)} \right) \tag{9}$$

The multiplication of **H** and **Q** where **Q** is a block diagonal (e.g., there is correlation in space but not in time) is a special case of the algorithm where **D** is an identity matrix.

2.2 Floating point operations

The number of floating point operations required for a direct multiplication of a matrix **A** by a matrix **B** can be expressed as a function of the dimensions of these matrices (for details, see Golub and Van Loan, 1996):

$$\mathrm{AB}_{\mathrm{direct}} = nqt(2pr-1). \tag{10}$$

Similarly, the cost of the "indirect" multiplication algorithm presented in the last section is

$$\mathrm{AB}_{\mathrm{indirect}} = \overbrace{n\{prq\}}^{\substack{\text{Scalar multiplication}\\ \text{(step 2)}}} + \overbrace{q\{(p-1)nr\}}^{\substack{\text{Addition of column blocks}\\ \text{(step 2)}}} + \overbrace{q\{(2r-1)nt\}}^{\substack{\text{Matrix multiplication}\\ \text{(step 3)}}}. \tag{11}$$

Equation (11) is based on the fact that steps 2 and 3 are each repeated q times. The relative computational performance of the indirect method can therefore be expressed as

$$\frac{\mathrm{AB}_{\mathrm{indirect}}}{\mathrm{AB}_{\mathrm{direct}}} = \frac{2pr+2rt-r-t}{t(2pr-1)}. \tag{12}$$

For H and Q, this simplifies to

$$\frac{\mathrm{HQ}_{\mathrm{indirect}}}{\mathrm{HQ}_{\mathrm{direct}}} = \frac{2(m_\tau+m_s-1)}{2\,m_\tau m_s-1}. \tag{13}$$

Note that the number of observations n does not affect the relative performance of the algorithm. Asymptotically, Eq. (13) approaches zero with increasing m_τ and m_s. For the Gourdji et al. (2012) problem, this ratio is 7.14×10^{-4}, a savings of over 99.9 % floating point operations relative to the direct approach.

For the more generic case of multiplying **A** and **B**, consider the special simplifying case where $n=m$ and $p=q=r=t=\sqrt{n}$; the floating point operations required by the direct and indirect methods become

$$\mathrm{AB}_{\mathrm{direct}} = 2n^3-n^2 \text{ and} \tag{14}$$

$$\mathrm{AB}_{\mathrm{indirect}} = 4n^2\sqrt{n}-2n^2. \tag{15}$$

This results in an asymptotic complexity of $\mathrm{O}(n^3)$ for the direct method, and $\mathrm{O}(n^{2.5})$ for the indirect method. The direct method is thus more economical only for $n<3$, and the relative cost of the indirect method decreases exponentially thereafter. The overall computational cost could be reduced further if the matrix multiplications in step 3 of the algorithm were computed through the Strassen or Coppersmith and Winograd algorithm. For a special case where **D** is composed of zeros and ones, the computational cost can also be further reduced by avoiding scalar multiplications, as listed in step 2 of the algorithm.

2.3 Other computational aspects of the indirect approach

Beyond the economies in floating point operations, the indirect method also dramatically decreases the random access memory requirements for matrix multiplication, because the proposed approach eliminates the need to create or store the full matrix **B** (or **Q**). Thus, again taking the special case of $n=m$ and $p=q=r=t=\sqrt{n}$ as an example, the memory requirement for storing **D** and **E** is $\mathrm{O}(n^2)$, whereas it is $\mathrm{O}(n^4)$ if **B** is explicitly stored in memory.

To maximize the computational gains from the algorithm, it is also necessary to take into account the method adopted by different computer languages for storing arrays in computer memory. For example, C and C++ are far more efficient at multiplying **Q** by $\mathbf{H}^T$ than **H** by **Q**. This results from the fact that C and C++ store matrices in row-major order, whereas Fortran and MATLAB store them in column-major order. The Fortran and C++ codes submitted with this manuscript show how these operations are performed for matrices stored in column (Fortran and MATLAB) or row-major order (C and C++) for square symmetric **Q**, **D** and **E** matrices. In addition, whether the matrices are stored as single-dimension vectors or as two-dimensional matrices also affects the computational performance of the proposed algorithm, especially in C and C++, which are not array-based programming languages.

In comparison to the direct method, the indirect approach for matrix multiplication is also fault tolerant and amenable to distributed parallel programming or "out of core" matrix multiplication, as each column block of **AB** (or **HQ**) can be obtained separately without any communication between processors populating the individual blocks.

In the case of the solution of an inverse problem, the multiplication of $\mathbf{HQH}^T$ can also be completed block by block:

$$\mathbf{HQH}^T = \sum_{j=1}^{m_\tau} \left(\left(\sum_{i=1}^{m_\tau} \mathbf{h}_i \, d_{(i,j)} \right) \mathbf{E} \right) \mathbf{h}_j^T, \quad (16)$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ represent column blocks of the $\mathbf{H}$ matrix as defined earlier. The computational efficiency of the matrix multiplication of $\mathbf{H}$, $\mathbf{Q}$ and $\mathbf{H}^T$ can be further improved if the symmetry of $\mathbf{HQH}^T$ is taken into account (see details on quadratic forms (Harville, 2008)). However there are no "Basic Linear Algebra Subroutines" (Blackford et al., 2002; Dongarra et al., 1988; Lawson et al., 1979) that take this property into account, and additional work would be required to develop these methods for application in inverse problems and statistics.

3 Computation of aggregated a posteriori covariance in large linear space–time inverse problems

The a posteriori covariance matrix ($\mathbf{V}_{\hat{\mathbf{s}}}$; Eq. 3) is typically dense, and calculating $\mathbf{V}_{\hat{\mathbf{s}}}$ is a computational bottleneck for large inverse problems. For example, computing $\mathbf{V}_{\hat{\mathbf{s}}}$ explicitly for the Gourdji et al. (2012) problem would require approximately 1.06×10^{18} floating point operations, and over 56 TB of RAM. We propose an algorithm for computing the a posteriori covariance directly at aggregated scales, which are typically the scales of most interest as described in Sect. 1, without explicitly calculating $\mathbf{V}_{\hat{\mathbf{s}}}$. We use the estimation of a posteriori uncertainties at the native spatial scales ($1^\circ \times 1^\circ$ grid scale in the prototypical example) but for estimates averaged across larger temporal scales as an example.

3.1 Algorithm

The algorithm is presented for a $\mathbf{V}_{\hat{\mathbf{s}}}$ design consistent with Eqs. (4) and (5), i.e., where the diagonal blocks describe the spatial covariance, and the off-diagonal blocks describe the decay of this covariance with time. The particular design framework of $\mathbf{V}_{\hat{\mathbf{s}}}$ used in this study does not hinder the application of the proposed algorithm for obtaining a posteriori covariances and cross-covariances in inverse problems where $\mathbf{V}_{\hat{\mathbf{s}}}$ has a different design, or where the aggregation is to be conducted over a different desired dimension.

The calculation of the a posteriori covariance at the native spatial resolution aggregated temporally over a desired time period proceeds as follows:

1. Sum all blocks of the matrix corresponding to the $k = t_u - t_l + 1$ time periods between periods t_l and t_u over which the a posteriori uncertainties are to be aggregated, where $1 \le t_l < m_\tau$, $t_l \le t_u < m_\tau$. For $\mathbf{Q}$ expressed as a Kronecker product as given in Sect. 2.1, this is the sum of all entries between t_l and t_u in $\mathbf{D}$, multiplied by $\mathbf{E}$ (spatial covariance):

$$\mathbf{Q}_{\text{sum}}(m_s \times m_\tau) = \left(\left(\sum_{j=t_l}^{t_u} \sum_{i=t_l}^{t_u} d_{(i,j)} \right) \mathbf{E} \right), \quad (17)$$

where $\mathbf{Q}_{\text{sum}}$ represents the sum of all $\mathbf{Q}$ blocks between t_l and t_u.

2. Sum all column blocks of the $\mathbf{HQ}$ (see Eq. (9)):

$$(\mathbf{HQ})_{\text{sum}} = \left(\sum_{j=t_l}^{t_u} \left(\sum_{i=1}^{m_\tau} \boldsymbol{h}_i \, d_{(i,j)} \right) \mathbf{E} \right)_{(n \times m_s)}, \quad (18)$$

where $(\mathbf{HQ})_{\text{sum}}$ represents the sum of all $\mathbf{HQ}$ column blocks as shown in Eq. (9) between t_l and t_u.

3. Compute the aggregated grid-scale a posteriori covariance $\bar{\mathbf{V}}_{\hat{\mathbf{s}}}$ for the estimates averaged over the desired time periods:

$$\bar{\mathbf{V}}_{\hat{\mathbf{s}}} = \frac{\left(\mathbf{Q}_{\text{sum}} - (\mathbf{HQ})_{\text{sum}}^T \left(\mathbf{HQH}^T + \mathbf{R} \right)^{-1} (\mathbf{HQ})_{\text{sum}} \right)}{k^2}, \quad (19)$$

where $\bar{\mathbf{V}}_{\hat{\mathbf{s}}}$ is the covariance of $\hat{\mathbf{s}}$ temporally averaged for time periods t_l to t_u .

3.2 Floating point operations

The number of floating point operations required for the direct calculation of $\mathbf{V}_{\hat{\mathbf{s}}}$ (Eq. 3) and its aggregation over k time periods is compared to the calculation of the aggregated $\mathbf{V}_{\hat{\mathbf{s}}}$ using the algorithm described above. The floating point operations required for multiplying $\mathbf{H}$ by $\mathbf{Q}$ and $\mathbf{HQ}$ by $\mathbf{H}^T$, for adding $\mathbf{R}$ to $\mathbf{HQH}^T$, for taking the inverse of $\left(\mathbf{HQH}^T + \mathbf{R}\right)$, and for dividing the aggregated covariance by k^2 are excluded in the floating point operation count, because the cost of these operations is the same for both approaches. Of course, computational savings can be achieved for both by following the algorithm outlined in Sect. 2 for the matrix multiplications.

The number of floating point operations required for obtaining grid-scale a posteriori covariance from the direct method can be divided into four sequential operations: (1) matrix multiplication of $(\mathbf{HQ})^T$ and $(\mathbf{HQH}^T + \mathbf{R})^{-1}$, (2) matrix multiplication of $(\mathbf{HQ})^T (\mathbf{HQH}^T + \mathbf{R})^{-1}$ and $\mathbf{HQ}$, (3) subtraction of $(\mathbf{HQ})^T (\mathbf{HQH}^T + \mathbf{R})^{-1}\mathbf{HQ}$ from $\mathbf{Q}$, and (4) summation of all k^2 and m_s^2 (spatial covariance) blocks of $\mathbf{V}_{\hat{\mathbf{s}}}$. The floating point operations for these four calculations are

$$\bar{V}_{\hat{s}_{\text{direct}}} = [n m_\tau m_s (2n-1)] + \left[m_\tau^2 m_s^2 (2n-1) \right] + \left[m_\tau^2 m_s^2 \right] + \left[m_s^2 \left(k^2 - 1 \right) \right]. \quad (20)$$

For the algorithm proposed in Sect. 3.1, five operations are required to obtain aggregated a posteriori covariance for the desired time period: (1) summation of k and m_s^2 (spatial covariance) blocks of $\mathbf{Q}$, (2) summation of k and $n \times m_s$ blocks of $\mathbf{HQ}$, (3) multiplication of $(\mathbf{HQ})^T_{\text{sum}}$ and $\left(\mathbf{HQH}^T + \mathbf{R}\right)^{-1}$, (4) multiplication of $(\mathbf{HQ})^T_{\text{sum}}\left(\mathbf{HQH}^T + \mathbf{R}\right)^{-1}$ and $(\mathbf{HQ})_{\text{sum}}$, and (5) subtraction of $(\mathbf{HQ})^T_{\text{sum}}\left(\mathbf{HQH}^T + \mathbf{R}\right)^{-1}(\mathbf{HQ})_{\text{sum}}$ from $\mathbf{Q}_{\text{sum}}$. The last three of these are all part of step 3 of the algorithm. The floating point operations for these five calculations are

$$\bar{V}_{\hat{\mathbf{s}}_{\text{indirect}}} = \qquad (21)$$

$$\overbrace{k^2-1+m_s^2}^{\substack{\text{compute}\\ \mathbf{Q}_{\text{sum}}\\ \text{(step 1)}}} + \overbrace{[nm_s(k-1)]}^{\substack{\text{compute}\\ (\mathbf{HQ})_{\text{sum}}\\ \text{(step 2)}}} + \overbrace{[nm_s(2n-1)] + \left[m_s^2(2n-1)\right] + \left[m_s^2\right]}^{\substack{\text{Compute}\\ \bar{V}\hat{s}_{\text{indirect}}\\ \text{(step 3)}}}.$$

Asymptotically, $\frac{\bar{\mathbf{V}}\hat{\mathbf{s}}_{\text{indirect}}}{\bar{\mathbf{V}}\hat{\mathbf{s}}_{\text{direct}}}$ approaches zero with increasing n, m_τ and m_s. For the Gourdji et al. (2012) problem, this ratio is 5.34×10^{-7} when evaluating the a posteriori covariance aggregated over the full year ($k = m_\tau$), a savings of over 99.9999 % in floating point operations relative to the direct approach.

For the special simplifying case where $n = m$ (i.e., $n = m_\tau m_s$) and $m_\tau = m_s$, the ratio of floating point operations required by the direct and the indirect methods becomes

$$\frac{\bar{V}\hat{s}_{\text{indirect}}}{\bar{V}\hat{s}_{\text{direct}}} = \frac{2n^2\sqrt{n} + 2n^2 + n\sqrt{n}\,(k-2) + n + k^2 - 1}{4n^3 - n^2 + n(k^2-1)}. \qquad (22)$$

This results in an asymptotic complexity of $O(n^3)$ for the direct method, and $O(n^{2.5})$ for the indirect method. The reduced memory requirements are arguably even more important, however, as the proposed algorithm makes it possible to compute a posteriori covariances at any temporal resolution without explicitly creating $\mathbf{V}_{\hat{\mathbf{s}}}$.

4 Conclusions

We propose two algorithms to lower the computational cost of performing large linear space–time inverse problems. The proposed matrix multiplication algorithm can be implemented with any matrices, as long as one of them can be expressed as a Kronecker product of smaller matrices, making it broadly applicable in other areas of statistics and signal processing, among others (Van Loan, 2000). Furthermore, although not explicitly discussed in this work, the Kronecker product representation of covariance matrices can be used for reducing the memory allocation complexity of storing error covariance matrices in both batch inverse problems and data assimilation (for application, see Singh et al., 2011). The presented a posteriori covariance computation algorithm can provide aggregated uncertainty covariances even for extremely large space–time inverse problems with dramatically decreased computational and memory requirements. The mounting availability of massive volumes of data (e.g., satellite observations) will further increase the computational cost associated with the solution of inverse problems in the Earth sciences, making algorithms such as the ones presented here, as well as further improvements to the computational efficiency associated with the solution of large inverse problems, increasingly important.

Appendix A

Description of the submitted code

Two MATLAB code files demonstrating the application of the methods proposed in this manuscript are included as supplementary material. The MATLAB script file "HQ_HQHt.m" allows users to experiment with different sizes of random covariance matrices in a Kronecker product form and computes $\mathbf{HQ}$ and $\mathbf{HQH}^T$ using the direct method as well as the method presented in Sect. 2.1. The second MATLAB script file "Uncertainty_Computations.m" allows users to experiment with random matrices for computing a posteriori covariances aggregated either over all time periods or for specified time periods. A detailed description of the codes is also given at the beginning of the script files. Note that these codes are provided to illustrate the two algorithms proposed in this research, but these codes should not be used to assess the computational performance of these algorithms. This is because MATLAB uses (1) highly optimized multithreaded external libraries (Basic Algebra Subroutines) for performing matrix multiplication, and (2) automatic memory management (e.g., allocation and reallocation of memory).

To supplement the MATLAB routines, we also include completely serial Fortran (filename: HQ.f90) and C++ (filename: HQ.cpp) code for performing the matrix multiplication of $\mathbf{H}$ and $\mathbf{Q}$ matrix. Although the performance of these codes may vary depending on computer architecture, the performance will approximately reflect the computational savings described in the manuscript. For example, for the Fortran code compiled using the GFortran compiler on a Intel Xeon X5660 2.80 GHz machine with 96 GB RAM, a matrix multiplication (1) with $n = 8503$, $r = t = 10$, and $p = q = 100$ took approximately 12 s using the direct method and approximately 2.2 s using the indirect method; (2) with $n = 8503$, $r = t = 100$, and $p = q = 10$, the direct method took approximately 9.4 s whereas the indirect method took approximately 1.0 s; and (3) with $n = 15000$, $r = t = 150$ and $p = q = 150$, the direct method took approximately 3 h whereas the indirect method took approximately 3.4 min.

Supplementary material related to this article is available online at: http://www.geosci-model-dev.net/6/583/2013/gmd-6-583-2013-supplement.zip.

Acknowledgements. This material is based upon work supported by the National Aeronautics and Space Administration under Grant No. NNX12AB90G and the National Science Foundation under Grant No. 1047871.

Edited by: L. Gross

References

Aster, R. C., Borchers, B., and Thurber, C. H.: Parameter estimation and inverse problems, Academic Press, 376 pp., 2013.

Atmadja, J. and Bagtzoglou, A. C.: State of the art report on mathematical methods for groundwater pollution source identification, Environ. Forensics, 2, 205–214, 2001.

Bennett, A. F.: Inverse modeling of the ocean and atmosphere, Cambridge University Press, Cambridge, UK, 2002.

Bini, D. and Lotti, G.: Stability of Fast Algorithms for Matrix Multiplication, Numer. Math., 36, 63–72, doi:10.1007/bf01395989, 1980.

Bini, D.: Fast Matrix Multiplication, in: Handbook of Linear Algebra, edited by: Hogben, L., Discrete Mathematics and its Applications, Chapman & Hall/CRC, Boca Raton, 47–41 to 47–12, 2010.

Blackford, L. S., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., Heroux, M., Kaufman, L., Lumsdaine, A., Petitet, A., Pozo, R., Remington, K., and Whaley, R. C.: An updated set of Basic Linear Algebra Subprograms (BLAS), ACM Trans. Math. Softw., 28, 135–151, doi:10.1145/567806.567807, 2002.

Ciais, P., Rayner, P., Chevallier, F., Bousquet, P., Logan, M., Peylin, P., and Ramonet, M.: Atmospheric inversions for estimating CO_2 fluxes: methods and perspectives, Greenhouse Gas Inventories, 69–92, 2011.

Coppersmith, D. and Winograd, S.: Matrix Multiplication via Arithmetic Progressions, J. Symb. Comput., 9, 251–280, doi:10.1016/s0747-7171(08)80013-2, 1990.

Dongarra, J. J., Ducroz, J., Hammarling, S., and Hanson, R. J.: An Extended Set of Fortran Basic Linear Algebra Subprograms, ACM Trans. Math. Softw., 14, 1–17, doi:10.1145/42288.42291, 1988.

Engelen, R. J., Denning, A. S., and Gurney, K. R.: On error estimation in atmospheric CO_2 inversions, J. Geophys. Res., 107, 4635, doi:10.1029/2002JD002195, 2002.

Enting, I. G.: Inverse problems in atmospheric constituent transport, Cambridge Univ Press, 408 pp., 2002.

Eve, J.: On O(n^2 log n) algorithms for $n \times n$ matrix operations, University of Newcastle upon Tyne, Technical Report Series, Newcastle upon Tyne, No. CS-TR1169, 2009.

Gohberg, I. and Olshevsky, V.: Fast algorithms with preprocessing for matrix-vector multiplication problems, J. Complex., 10, 411–427, 1994.

Golub, G. H. and Van Loan, C. F.: Matrix Computations, 3rd ed., The Johns Hopkins University Press, London, 1996.

Gourdji, S. M., Mueller, K. L., Yadav, V., Huntzinger, D. N., Andrews, A. E., Trudeau, M., Petron, G., Nehrkorn, T., Eluszkiewicz, J., Henderson, J., Wen, D., Lin, J., Fischer, M., Sweeney, C., and Michalak, A. M.: North American CO_2 exchange: inter-comparison of modeled estimates with results from a fine-scale atmospheric inversion, Biogeosciences, 9, 457–475, doi:10.5194/bg-9-457-2012, 2012.

Harville, D. A.: Matrix Algebra From a Statistician's Perspective, 2nd ed., Springer, Dodrecht, Netherlands, 2008.

Horn, R. A. and Johnson, C. R.: Matrix analysis, Cambridge University Press, 1990.

Hyndman, D. W., Day-Lewis, F. D., Singha, K., and Union, A. G.: Subsurface Hydrology: Data Integration for Properties and Processes, American Geophysical Union, 253 pp., 2007.

Kitanidis, P. K.: Quasi-linear geostatistical theory for inversing, Water Resour. Res., 31, 2411–2419, 1995.

Laderman, J. D.: A noncommutative algorithm for multiplying 33 matrices using 23 multiplications, Bull. Amer. Math. Soc, 82, 126–128, 1976.

Lawson, C. L., Hanson, R. J., Kincaid, D. R., and Krogh, F. T.: Basic linear algebra subprograms for Fortran usage, ACM Trans. Math. Softw., 5, 308–325, doi:10.1145/355841.355847, 1979.

Li, J., Ranka, S., and Sahni, S.: Strassen's Matrix Multiplication on GPUs, Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on, 2011, 157–164, 2011.

Liu, X. and Zhai, Z.: Inverse modeling methods for indoor airborne pollutant tracking: literature review and fundamentals, Indoor air, 17, 419–438, 2007.

Meirink, J. F., Bergamaschi, P., and Krol, M. C.: Four-dimensional variational data assimilation for inverse modelling of atmospheric methane emissions: method and comparison with synthesis inversion, Atmos. Chem. Phys., 8, 6341–6353, doi:10.5194/acp-8-6341-2008, 2008.

Michalak, A. M. and Kitanidis, P. K.: A method for enforcing parameter nonnegativity in Bayesian inverse problems with an application to contaminant source identification, Water Resour. Res., 39, 1033, doi:10.1029/2002WR001480, 2003.

Mouilleron, C.: Efficient computation with structured matrices and arithmetic expressions, Ecole normale supérieure de lyon-ENS LYON, 2011.

Nowak, W., Tenkleve, S., and Cirpka, O. A.: Efficient computation of linearized cross-covariance and auto-covariance matrices of interdependent quantities, Math. Geol., 35, 53–66, 2003.

Saibaba, A. K. and Kitanidis, P. K.: Efficient methods for large-scale linear inversion using a geostatistical approach, Water Resour. Res., 48, W05522, doi:10.1029/2011WR011778, 2012.

Singh, K., Jardak, M., Sandu, A., Bowman, K., Lee, M., and Jones, D.: Construction of non-diagonal background error covariance matrices for global chemical data assimilation, Geosci. Model Dev., 4, 299–316, doi:10.5194/gmd-4-299-2011, 2011.

Strassen, V.: Gaussian Elimination is not Optimal, Numer. Math., 13, 354–356, doi:10.1007/bf02165411, 1969.

Thompson, R. L., Gerbig, C., and Rödenbeck, C.: A Bayesian inversion estimate of N_2O emissions for western and central Europe and the assessment of aggregation errors, Atmos. Chem. Phys., 11, 3443–3458, doi:10.5194/acp-11-3443-2011, 2011.

Van Loan, C. F.: The ubiquitous Kronecker product, Journal of Computational and Applied Mathematics, 123, 85–100, doi:10.1016/s0377-0427(00)00393-9, 2000.

Zhang, T. and Chen, Q.: Identification of contaminant sources in enclosed environments by inverse CFD modeling, Indoor air, 17, 167–177, 2007.