MONASH
BUSINESS
SCHOOL

ISSN 1440-771X

Department of Econometrics and Business Statistics

http://business.monash.edu/econometrics-and-business-statistics/research/publications

Optimal non-negative forecast reconciliation

Shanika L Wickramasuriya

Berwin A Turlach

Rob J Hyndman

March 2019

Working Paper ??/??

Optimal non-negative forecast reconciliation

Shanika L Wickramasuriya
Department of Statistics,
University of Auckland,
Auckland, New Zealand.
Email: s.wickramasuriya@auckland.ac.nz

Berwin A Turlach
Centre for Applied Statistics,
The University of Western Australia,
Crawley, Australia.
Email: berwin.turlach@uwa.edu.au

Rob J Hyndman
Department of Econometrics and Business Statistics,
Monash University,
Melbourne, Australia.
Email: rob.hyndman@monash.edu

13 March 2019

Abstract

To be written

Keywords: Aggregation, Australian tourism, Coherent forecasts, Contemporaneous error correlation, Forecast combinations, Least squares, Non-negative, Spatial correlations.

We don't guarantee that the reconciled forecasts are unbiased. So do you think the word "optimal" (in the title) suits for this paper?

1 Introduction

Forecast reconciliation is the problem of ensuring that disaggregated forecasts add up to the corresponding forecasts of the aggregated time series. This is a common problem in manufacturing, for example, where time series of sales are disaggregated in several ways — by region, product-type, and so on. There are often tens of thousands of forecasts at the most disaggregated level, and these are required to sum to give forecasts at higher levels of aggregation — a property known as "coherence".

A simple solution would be to forecast the most disaggregated series and sum the results. However, this tends to give poor aggregate forecasts due to the low signal-to-noise ratio in the disaggregated time series. Instead, a better solution is to forecast all the series at all levels of aggregation, and then reconcile them so they are coherent; that is, so that the forecasts of the disaggregated series add up to the forecasts of the aggregated series. Least squares reconciliation was proposed by Hyndman, Ahmed, et al. (2011), whereby the reconciled forecasts are as close as possible (in the L_2 sense) to the original forecasts subject to the aggregation constraint. This result was extended by Hyndman, Lee, and Wang (2016) to a larger class of reconciliation problems, and by Wickramasuriya, Athanasopoulos, and Hyndman (2018) who showed that the resulting reconciled forecasts are minimum variance unbiased estimators.

Most of the applications that we have found are inherently non-negative in nature, where the time series take only non-negative values such as revenue, demand, and counts of people. In such circumstances, it is important to ensure that the reconciled forecasts are non-negative, as forecasters and practitioners need to be able to make meaningful managerial decisions. Unfortunately, the MinT approach proposed by Wickramasuriya, Athanasopoulos, and Hyndman (2018) and its variants fail to guarantee this property even when the base forecasts are non-negative.

This can be overcome by explicitly imposing the non-negativity constraints on the reconciliation procedure. One simple technique is to overwrite any negatives in the reconciled forecasts with zeros. However, the resulting approximate solution has ill-defined mathematical properties. This type of overwriting approximation method is used commonly in alternating least squares estimations. The problem is that it does not necessarily lower the objective function in each iteration, and therefore the convergence of the solution to the least squares minimum is not guaranteed. There are available algorithms that solve these problems in a mathematically rigorous way; we discuss how these can be applied to the forecast reconciliation problem.

2 The optimization problem

2.1 Notation and MinT reconciliation

We follow the notation of Wickramasuriya, Athanasopoulos, and Hyndman (2018) and let $\boldsymbol{y}_t \in \mathbb{R}^m$ denote a vector of observations at time t comprising all series of interest including both disaggregated and aggregated time series. We also define $\boldsymbol{b}_t \in \mathbb{R}^n$ to be the vector of the most disaggregated series at time t. These two vectors are connected via $\boldsymbol{y}_t = \boldsymbol{S}\boldsymbol{b}_t$ where $\boldsymbol{S}$ is the "summing" matrix of order $m \times n$ showing how the various aggregated time series in $\boldsymbol{y}_t$ are constructed from the disaggregated series in $\boldsymbol{b}_t$.

Let $\hat{\boldsymbol{y}}_T(h)$ is a vector of initial ("base") h-step-ahead forecasts, made at time T, stacked in the same order as $\boldsymbol{y}_t$. These will not generally be coherent. The least squares reconciled forecasts are given by

$$\tilde{\boldsymbol{y}}_T(h) = \boldsymbol{S}\tilde{\boldsymbol{b}}_T(h),$$

where

$$\tilde{\boldsymbol{b}}_T(h) = \left[(\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S})^{-1}\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\right]\hat{\boldsymbol{y}}_T(h), \tag{1}$$

and $\boldsymbol{\Lambda}_h$ is a weighting matrix. Wickramasuriya, Athanasopoulos, and Hyndman (2018) showed that setting $\boldsymbol{\Lambda}_h = \text{var}[\boldsymbol{y}_{t+h} - \hat{\boldsymbol{y}}_t(h) \mid \boldsymbol{\mathcal{I}}_t]\}$ to be the covariance matrix of the h-step-ahead base forecast errors minimizes the trace of $\text{var}[\boldsymbol{y}_{t+h} - \tilde{\boldsymbol{y}}_t(h) \mid \boldsymbol{\mathcal{I}}_t]$ amongst all possible unbiased reconciliations. They derived an alternative expression for $\tilde{\boldsymbol{b}}_T(h)$:

$$\tilde{\boldsymbol{b}}_T(h) = \left[\boldsymbol{J} - \boldsymbol{J}\boldsymbol{\Lambda}_h\boldsymbol{U}(\boldsymbol{U}'\boldsymbol{\Lambda}_h\boldsymbol{U})^{-1}\boldsymbol{U}'\right]\hat{\boldsymbol{y}}_T(h), \tag{2}$$

where $\boldsymbol{J} = \left[\,\boldsymbol{0}_{n\times(m-n)} \,\vdots\, \boldsymbol{I}_n\,\right]$, $\boldsymbol{U}' = \left[\,\boldsymbol{I}_{m-n} \,\vdots\, -\boldsymbol{C}_{(m-n)\times n}\,\right]$, and $\boldsymbol{S} = \begin{bmatrix}\boldsymbol{C}_{(m-n)\times n} \\ \hdashline \boldsymbol{I}_n\end{bmatrix}$. The use of Eq. (2) is computationally less demanding especially for high-dimensional hierarchical time series. It needs only one matrix inversion of order $(m-n)\times(m-n)$, whereas Eq. (1) needs two matrix inversions of orders $m \times m$ and $n \times n$. Typically in many applications $m - n < n < m$.

2.2 A quadratic programming solution

To ensure that all entries in $\tilde{\boldsymbol{y}}_T(h)$ are non-negative, it is sufficient to guarantee that all entries in $\tilde{\boldsymbol{b}}_T(h)$ are non-negative. Even though the solution of $\tilde{\boldsymbol{b}}_T(h)$ is derived based on a minimization of the variances of the reconciled forecast errors across the entire structure, it is also apparent from Eq. (1) that $\tilde{\boldsymbol{b}}_T(h)$ is the generalized least squares solution to the following regression problem:

$$\min_{\ddot{\boldsymbol{b}}} \frac{1}{2}[\hat{\boldsymbol{y}}_T(h) - \boldsymbol{S}\ddot{\boldsymbol{b}}]'\boldsymbol{\Lambda}_h^{-1}[\hat{\boldsymbol{y}}_T(h) - \boldsymbol{S}\ddot{\boldsymbol{b}}] = \min_{\ddot{\boldsymbol{b}}} \frac{1}{2}\ddot{\boldsymbol{b}}'\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}\ddot{\boldsymbol{b}} - \ddot{\boldsymbol{b}}'\boldsymbol{S}'\boldsymbol{\Lambda}_h'^{-1}\hat{\boldsymbol{y}}_T(h) + \frac{1}{2}\hat{\boldsymbol{y}}_T'(h)\boldsymbol{\Lambda}_h^{-1}\hat{\boldsymbol{y}}_T(h).$$

This suggests that the non-negativity issue can be handled by solving the following quadratic programming problem:

$$\begin{aligned} \min_{\ddot{\boldsymbol{b}}} q(\ddot{\boldsymbol{b}}) := \min_{\ddot{\boldsymbol{b}}} \frac{1}{2}\ddot{\boldsymbol{b}}'\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}\ddot{\boldsymbol{b}} - \ddot{\boldsymbol{b}}'\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\hat{\boldsymbol{y}}_T(h) \\ \text{s.t.} \quad \ddot{\boldsymbol{b}} \geq \boldsymbol{0}, \end{aligned} \tag{3}$$

where the inequalities in Eq. (3) have to hold component-wise. The final non-negative reconciled forecasts are then

$$\check{\boldsymbol{y}}_T(h) = \boldsymbol{S}\check{\boldsymbol{b}}_T(h),$$

where $\check{\boldsymbol{b}}_T(h)$ is the solution to the quadratic programming problem in Eq. (3). This estimation problem is also referred to as "non-negative least squares" (NNLS).

There are a few important features of this minimization problem that are worth discussing. Consider the following definitions:

Definition 2.1. *A vector $\ddot{\boldsymbol{b}}$ is said to be feasible if it satisfies all of the constraints in the quadratic programming problem in Eq. (3). The feasible region is the set of all feasible vectors $\ddot{\boldsymbol{b}}$, and the quadratic programming problem is said to be feasible if the feasible region is non-empty.*

Definition 2.2. *If $\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}$ is a positive definite matrix, i.e., $\boldsymbol{x}'(\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S})\boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$, then the objective function of the minimization problem in Eq. (3) is a strictly convex function.*

It is easy to show that $\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}$ is a positive definite matrix by using the fact that the matrix $\boldsymbol{S}$ is of full column rank and assuming that $\boldsymbol{\Lambda}_h$ is positive definite. These simple non-negativity constraints will also ensure that the feasible region is non-empty. Therefore, the quadratic

programming problem in Eq. (3) has a unique global solution; i.e., there are no local minima apart from the global minimum (Turlach and Wright, 2015).

Unlike Wickramasuriya, Athanasopoulos, and Hyndman (2018), we will not impose a constraint of unbiasedness on the reconciled forecasts obtained as the solution of the minimization problem in Eq. (3).

The well-known quadratic programming problem in Eq. (3) is easy to solve for small scale hierarchical or grouped structures using the `quadprog` package for R, which is designed to handle dense matrices (Turlach and Weingessel, 2013). However, we require matrices to be stored in a sparse format due to the large size of the structures that typically arise in forecast reconciliation.

Since we don't explicitly impose the condition for reconciled forecasts to be unbiased, non-negative reconciled forecasts can be slightly biased. Should we mention it here?

2.3 First-order optimality conditions

We can derive first-order necessary conditions for $\breve{\boldsymbol{b}}_T(h)$ to minimize the non-negative least squares problem.

Consider the Lagrangian function for the minimization problem in Eq. (3):

$$\mathcal{L}(\ddot{\boldsymbol{b}}, \boldsymbol{\lambda}) = q(\ddot{\boldsymbol{b}}) - \boldsymbol{\lambda}'\ddot{\boldsymbol{b}},$$

where $q(\ddot{\boldsymbol{b}}) = \frac{1}{2}\ddot{\boldsymbol{b}}'\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}\ddot{\boldsymbol{b}} - \ddot{\boldsymbol{b}}'\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\hat{\boldsymbol{y}}_T(h)$ and $\boldsymbol{\lambda}$ is a Lagrange multiplier vector. The Karush-Kuhn-Tucker (KKT) optimality conditions that need to be satisfied by $\breve{\boldsymbol{b}}_T(h)$ are

$$\begin{aligned}
\nabla_{\boldsymbol{b}}\mathcal{L}[\breve{\boldsymbol{b}}_T(h), \boldsymbol{\lambda}^*] &= \boldsymbol{0}, && && \text{(4a)}\\
\breve{b}_{T,i}(h) &= 0, && \forall i \in \mathcal{A}[\breve{\boldsymbol{b}}_T(h)], && \text{(4b)}\\
\breve{b}_{T,i}(h) &> 0, && \forall i \notin \mathcal{A}[\breve{\boldsymbol{b}}_T(h)], && \text{(4c)}\\
\lambda_i^* &\geq 0, && \forall i \in \mathcal{A}[\breve{\boldsymbol{b}}_T(h)], && \text{(4d)}\\
\lambda_i^* &= 0, && \forall i \notin \mathcal{A}[\breve{\boldsymbol{b}}_T(h)], && \text{(4e)}\\
\lambda_i^*\breve{b}_{T,i}(h) &= 0, && \forall i \in \{1, 2, \ldots, n\}, && \text{(4f)}
\end{aligned}$$

where $\breve{b}_{T,i}(h)$ is the ith component of $\breve{\boldsymbol{b}}_T(h)$ and $\mathcal{A}[\breve{\boldsymbol{b}}_T(h)]$ is referred to as the active set, and is defined as

$$\mathcal{A}[\breve{\boldsymbol{b}}_T(h)] = \left\{ i \in (1, 2, \ldots, n) \;\middle|\; \breve{b}_{T,i}(h) = 0 \right\}.$$

The first optimality condition in Eq. (4a) leads to $\boldsymbol{\lambda}^*$ being computed as

$$\boldsymbol{\lambda}^* = \nabla q[\check{\boldsymbol{b}}_T(h)] = \boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}\check{\boldsymbol{b}}_T(h) - \boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\hat{\boldsymbol{y}}_T(h).$$

The conditions given in Eq. (4f) are referred as complementarity conditions. They indicate that at the optimal solution, either the ith constraint is active or $\lambda_i^* = 0$, or both. Specifically, it implies that the Lagrange multipliers that are associated with inactive inequality constraints are zero.

3 Algorithms

Since the pioneering work of Lawson and Hanson (1974), a variety of methods for solving the non-negative least squares problem have been proposed. In general, these can be divided into two main categories: active set(-like) and projection-based methods. The following sections briefly explain a few of the algorithms that belong to each of these classes and that are suitable for solving large-scale problems. A detailed review of methods for non-negative least squares is given by Chen and Plemmons (2009).

3.1 Active set(-like) methods

The first widely used active set method for solving non-negative least squares was that proposed by Lawson and Hanson (1974). The basic idea of this method is to transform the inequality constrained least squares problem into a sequence of equality constrained problems.

The feasible region that corresponds to a given set of consistent inequality constraints is given by a volume in multidimensional space. Any point in this region (a feasible solution) consists of one of two types of variables, depending on whether it lies on the boundary or within the feasible region. The former set of variables is referred to as the "active set". These are variables that can only lower the objective function by moving away from the feasible region, and must be assumed to lie on the boundary in order to ensure the feasibility of the solution. The latter set of variables is referred to as the "passive set". The active set method relies on the idea that, if the true active set is known in advance, the solution to the least squares problem is given by the unconstrained least squares solution, obtained by using the variables in the passive set and setting the regression coefficients for the active set variables to zero. This can be interpreted as an attempt to forecast hierarchical or grouped time series by simply refining the structure through the removal of the bottom-level series that are responsible for any negative or zero

reconciled forecasts. In practice, though, it is difficult to define such sets in advance. Therefore, we have to search for the optimal set iteratively.

Even though the standard active set method guarantees to terminate within a finite number of iterations, it has pessimistic upper bound on the numbers of iterations needed to reach the optimal solution. Suppose that we have n series at the bottom level, resulting in a total of 2^n possible active sets. Since the solution is unique, one of the 2^n combinations corresponds to the optimal active set. Therefore, finding the optimal set gives rise to a problem known as combinatorial difficulty. The main shortcoming of the standard active set method is that the algorithm is initialized by assuming an empty passive set, and only one variable is added from active set to passive set. Although QR updating and down-dating techniques are used to speed up the computations, more iterations (in other words, more time) might be required to find the optimal active set when handling large scale non-negative least squares problems. Therefore, several alternatives have been proposed for overcoming these difficulties.

One possibility is to initialize the algorithm with a non-empty passive set. A simple way to introduce such a set is to solve the original unconstrained least squares problem and then overwrite any negative values with zeros. This type of initialization is used widely in the field of multivariate curve resolution (Andrew and Hancewicz, 1998; Gemperline and Cash, 2003). Wang and Wang (2012) pointed out that such a guess would be reasonable in cases where there are few active constraints or when non-negativity is enforced in order to remove any observational or measurement errors that may be present. They proposed an alternative initialization strategy by using the solution of gradient projection methods after a few iterations. They showed theoretically that the sequence of sets generated by the projection-based algorithm converges to the optimal passive set of the active set method. Furthermore, they compared their choice of initialization with that of the overwrite solution of unconstrained least squares, and found the former to be better.

The second approach to enhancing the speed of the active set method involves including more than one variable from active set. This should be handled carefully, as it could lead to endless loops in the algorithm. Thus, these types of methods, which are referred to as "block principal pivoting methods", include a procedure for selecting a group of variables to exchange, and a backup rule in order to ensure the finite termination of the algorithm. Of the different variations of the active set methods that we have discussed thus far, we are particularly interested in this

type of active-set-like method, as they are quite efficient for handling large scale strictly convex quadratic programming problems with non-negativity constraints.

Block principal pivoting method

This section briefly reviews the theory behind the block principal pivoting method, which was developed for non-negative least squares problems by Júdice and Pires (1994). The procedure begins with the monotone linear complementarity problem (LCP) induced by the KKT optimal conditions given in Eq. (4), which needs to be satisfied by the optimal solution.

The monotone linear complementarity problem is

$$\ddot{\boldsymbol{g}} = \boldsymbol{S}'[\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}\ddot{\boldsymbol{b}} - \boldsymbol{\Lambda}_h^{-1}\hat{\boldsymbol{y}}_T(h)], \tag{5}$$

$$\ddot{\boldsymbol{g}} \geq \boldsymbol{0}, \tag{6}$$

$$\ddot{\boldsymbol{b}} \geq \boldsymbol{0}, \tag{7}$$

$$\ddot{b}_i\ddot{g}_i = 0, \;\; i = 1, 2, \ldots, n, \tag{8}$$

where $\ddot{b}_i$ and $\ddot{g}_i$ are the ith components of the vectors $\ddot{\boldsymbol{b}}$ and $\ddot{\boldsymbol{g}}$ respectively. The matrix $\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}$ is positive definite, as $\boldsymbol{S}$ is of full column rank and $\boldsymbol{\Lambda}_h$ is assumed to be positive definite. Thus, this strictly monotone LCP has a unique solution.

A point $(\ddot{\boldsymbol{b}}, \ddot{\boldsymbol{g}}) \in \mathbb{R}^{2n}$ is defined as a complementary solution if it satisfies Eq. (5) and (8).

Let the index set $\{1, 2, \ldots, n\}$ be partitioned into two mutually exclusive subsets F and G, such that $F \cup G = \{1, 2, \ldots, n\}$ and $F \cap G = \emptyset$. Furthermore, consider the partitions of $\ddot{\boldsymbol{b}}, \ddot{\boldsymbol{g}}$ and $\boldsymbol{S}$ according to the index sets F and G using the following notation:

$$\ddot{\boldsymbol{b}}_F = [\ddot{b}_i]_{i\in F}, \qquad \ddot{\boldsymbol{b}}_G = [\ddot{b}_i]_{i\in G},$$
$$\ddot{\boldsymbol{g}}_F = [\ddot{g}_i]_{i\in F}, \qquad \ddot{\boldsymbol{g}}_G = [\ddot{g}_i]_{i\in G},$$
$$\boldsymbol{S}_F = [\boldsymbol{S}_i]_{i\in F}, \qquad \boldsymbol{S}_G = [\boldsymbol{S}_i]_{i\in G},$$

where $\boldsymbol{S}_i$ is the ith column of $\boldsymbol{S}$.

Based on these partitions, we can re-express Eq. (5) as

$$\begin{bmatrix} \ddot{\boldsymbol{g}}_F \\ \ddot{\boldsymbol{g}}_G \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_F'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}_F & \boldsymbol{S}_F'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}_G \\ \boldsymbol{S}_G'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}_F & \boldsymbol{S}_G'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}_G \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{b}}_F \\ \ddot{\boldsymbol{b}}_G \end{bmatrix} - \begin{bmatrix} \boldsymbol{S}_F'\boldsymbol{\Lambda}_h^{-1}\hat{\boldsymbol{y}}_T(h) \\ \boldsymbol{S}_G'\boldsymbol{\Lambda}_h^{-1}\hat{\boldsymbol{y}}_T(h) \end{bmatrix}.$$

The algorithm starts by assigning $\ddot{\boldsymbol{b}}_G = \mathbf{0}$ and $\ddot{\boldsymbol{g}}_F = \mathbf{0}$. This particular choice will ensure that Eq. (8) is always satisfied for any values of $\ddot{\boldsymbol{b}}_F$ and $\ddot{\boldsymbol{g}}_G$.

The computation of the remaining unknown quantities $\ddot{\boldsymbol{b}}_F$ and $\ddot{\boldsymbol{g}}_G$ can be carried out by solving the following unconstrained least squares problem:

$$\bar{\boldsymbol{b}}_F = \min_{\ddot{\boldsymbol{b}}_F} \frac{1}{2} \left\| \boldsymbol{S}_F \ddot{\boldsymbol{b}}_F - \hat{\boldsymbol{y}}_T(h) \right\|^2_{\boldsymbol{\Lambda}_h^{-1}}, \tag{9}$$

and then setting

$$\bar{\boldsymbol{g}}_G = \boldsymbol{S}_G' [\boldsymbol{\Lambda}_h^{-1} \boldsymbol{S}_F \bar{\boldsymbol{b}}_F - \boldsymbol{\Lambda}_h^{-1} \hat{\boldsymbol{y}}_T(h)], \tag{10}$$

where $\|.\|_{\boldsymbol{\Lambda}_h^{-1}} = [\boldsymbol{S}_F \ddot{\boldsymbol{b}}_F - \hat{\boldsymbol{y}}_T(h)]' \boldsymbol{\Lambda}_h^{-1} [\boldsymbol{S}_F \ddot{\boldsymbol{b}}_F - \hat{\boldsymbol{y}}_T(h)]$. The solution pair $(\bar{\boldsymbol{b}}_F, \bar{\boldsymbol{g}}_G)$ is referred to as a complementary basic solution. If a complementary basic solution satisfies $\bar{\boldsymbol{b}}_F \geq \mathbf{0}$ and $\bar{\boldsymbol{g}}_G \geq \mathbf{0}$, then it is said to be feasible. In other words, it leads to the optimal solution of the constrained quadratic programming problem that we are interested in. Otherwise, the complementary basic solution is infeasible; i.e., there exists at least one $i \in F$ with $\bar{b}_i < 0$ or one $i \in G$ with $\bar{g}_i < 0$.

In the presence of an infeasible solution, we need to update sets F and G by exchanging the variables for which Eqs. (6) or (7) do not hold. Define the following two sets, which correspond to the infeasibilities in sets F and G:

$$I_1 = \{i \in F : \bar{b}_i < 0\} \qquad \text{and} \qquad I_2 = \{i \in G : \bar{g}_i < 0\}, \tag{11}$$

which gives the total set of infeasibilities, $I = I_1 \cup I_2$. For a given $\bar{I}_1 \subseteq I_1$ and $\bar{I}_2 \subseteq I_2$, F and G can be updated according to the following rules:

$$F = (F - \bar{I}_1) \cup \bar{I}_2 \qquad \text{and} \qquad G = (G - \bar{I}_2) \cup \bar{I}_1. \tag{12}$$

If $\text{card}(\bar{I}_1 \cup \bar{I}_2) > 1$, where $\text{card}(X)$ is the cardinality of set X, the algorithm is known as a block principal pivoting algorithm, whereas if $\text{card}(\bar{I}_1 \cup \bar{I}_2) = 1$, it is known as a single principal pivoting algorithm. The active set method that we discussed earlier can be seen as a single principal pivoting algorithm. However, they are not identical. The first block principal pivoting algorithm for solving a strictly monotonic LCP is due to the work of Kostreva (1978). Unfortunately, unlike the active set method, in which the variable to be exchanged is chosen carefully with the aim of reducing the objective function, the use of blocks of variables for

exchange can lead to a cycle, meaning that it is not guaranteed to provide the optimal solution. Although this occurs rarely, it is problematic. Júdice and Pires (1989) evaluated the performance of this algorithm on several test problems with a symmetric positive definite matrix. Their experiments concluded that this algorithm is quite efficient for handling large-scale strictly monotonic LCPs. Furthermore, they noted that the occurrence of cycles is extremely rare. In a later paper, Júdice and Pires (1994) proposed an extension of this algorithm to include finite termination, by incorporating Murty's single principal pivoting algorithm. Even though the algorithm proposed by Murty (1974) has finite termination, convergence can be slow for applications with large numbers of variables, as it changes only one variable per iteration. However, this algorithm is still beneficial for obtaining a complementary basic solution with a smaller number of infeasibilities than before. The steps of the hybrid algorithm are given below.

Algorithm 3.1 Block principal pivoting algorithm

INITIALIZATION: Let $F = \emptyset$, $G = \{1, 2, \ldots, n\}$, $\ddot{\boldsymbol{b}} = \mathbf{0}$, $\ddot{\boldsymbol{g}} = -\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\hat{\boldsymbol{y}}_T(h)$, $p = \bar{p} \leq 10$, and ninf $= n + 1$, and let α be a permutation of the set $\{1, 2, \ldots, n\}$.

1: IF $(\ddot{\boldsymbol{b}}_F \geq \mathbf{0}$ & $\ddot{\boldsymbol{g}}_G \geq \mathbf{0})$ THEN
2: Terminate the algorithm and $\breve{\boldsymbol{b}} = (\ddot{\boldsymbol{b}}_F, \mathbf{0})$ is the unique global solution.
3: ELSE
4: Define I_1 and I_2 as given in Eq. (11).
5: IF (card$(I_1 \cup I_2) <$ ninf) THEN
6: Set ninf = card$(I_1 \cup I_2)$, $p = \bar{p}$, $\bar{I}_1 = I_1$ and $\bar{I}_2 = I_2$.
7: ELSEIF (card$(I_1 \cup I_2) \geq$ ninf and $p \geq 1$) THEN
8: Set $p = p - 1$, $\bar{I}_1 = I_1$ and $\bar{I}_2 = I_2$.
9: ELSEIF (card$(I_1 \cup I_2) \geq$ ninf and $p = 0$) THEN
10: Set $\bar{I}_1 = \{r\}$ and $\bar{I}_2 = \emptyset$, if $r \in I_1$,
11: $\bar{I}_1 = \emptyset$ and $\bar{I}_2 = \{r\}$, if $r \in I_2$,
12: where r is the last element of the set $I_1 \cup I_2$ as for the order defined by α.
13: ENDIF
14: Update F and G as given by Eq. (12).
15: Compute $\bar{\boldsymbol{b}}_F$ and $\bar{\boldsymbol{g}}_G$ using Eqs. (9) and (10) respectively, and assign $\ddot{\boldsymbol{b}}_F = \bar{\boldsymbol{b}}_F$ and $\ddot{\boldsymbol{g}}_G = \bar{\boldsymbol{g}}_G$.
16: Return to line 1.
17: ENDIF

In particular, if $\boldsymbol{\Lambda}_h$ is a diagonal matrix with positive elements and $\hat{\boldsymbol{y}}_T(h) > \mathbf{0}$, Algorithm 3.1 can start by defining the initial conditions as

$$F = \{1, 2, \ldots, n\},\ G = \emptyset,\ \ddot{\boldsymbol{b}} = \tilde{\boldsymbol{b}},\ \ddot{\boldsymbol{g}} = \mathbf{0},$$

where $\tilde{\boldsymbol{b}}$ is the original unconstrained least squares solution with positive diagonal elements for $\boldsymbol{\Lambda}_h$. This algorithm always terminates within a finite number of iterations, because ninf decreases with time over the course of the algorithm. This is a consequence of the use of Murty's method.

Rather than selecting a subset of variables from I_1 and I_2, we speed up the computations by using the full sets of variables as $\bar{I}_1$ and $\bar{I}_2$ respectively. This is generally referred to as the "full exchange rule". The variable p in Algorithm 3.1 acts as a buffer for determining the number of full exchange rules that may be tried. If the number of infeasible variables increases due to the exchange rule, then p decreases by one. If the number of infeasible variables is still larger after $p-1$ steps, then Murty's method is used until it reaches a complementary solution with a smaller number of infeasible variables than a certain required value that is stored in ninf. This will occur in a finite number of iterations, due to the finite termination property of the Murty's method. The algorithm then switches to the full exchange rule, and iterates until the number of infeasible variables reaches zero.

This algorithm works well for numerically non-degenerate problems; i.e., each variable in F and G has a value that is clearly different from zero, given by the unconstrained solver in the feasible solution. When such is not the case, a variable with a value that is close to zero may be passed between F and G, due to the slight negative error caused by the unconstrained solver. In handling this issue, Cantarella and Piatek (2004) suggested that we assign

$$\begin{aligned} \ddot{b}_{F,i} &= 0, \qquad \text{if} \quad i \in H_1, \\ \ddot{g}_{G,i} &= 0, \qquad \text{if} \quad i \in H_2, \end{aligned}$$

where $H_1 = \{i \in F : \ddot{b}_i < \epsilon\}$ and $H_2 = \{i \in G : \ddot{g}_i < \epsilon\}$; $\ddot{b}_{F,i}$ and $\ddot{g}_{G,i}$ are the ith components of the $\ddot{\boldsymbol{b}}_F$ and $\ddot{\boldsymbol{g}}_G$ vectors respectively, and ϵ is set to 10^{-12}.

The choice of p plays an important rule. It should be fairly small in order to prevent unnecessary computations in which the full exchange rule is not effective. However, it should not be too small, otherwise Murty's method may be activated several times, thus reducing the efficiency of the algorithm. In general, $p = 3$ is a good choice (Júdice and Pires, 1994).

The performance of the block pivoting algorithm depends heavily on the number of times the full exchange rule fails and it has to switch to Murty's method. Based on an extensive study,

Kim and Park (2011) noted that Murty's method was not activated during the algorithm, which suggests that the full exchange rule will work effectively in practice.

The two sets F and G that are used in the block pivoting algorithm are not necessarily the same as the passive and active sets that are used in the active set method. In the active set method, each iteration is designed to find a solution, while remaining in the feasible region. As a consequence, a variable $\ddot{b}_i$, where i is in the passive set, has to satisfy $\ddot{b}_i \geq 0$ at every iteration; on the other hand, in the block pivoting algorithm, if i is in the set F, a variable $\ddot{b}_i$ can take any sign at any iteration except the last. Thus, the active set method needs an initial feasible solution to initiate the algorithm whereas block principal pivoting algorithm does not.

Kim and Park (2011) extended the ideas of Bro and De Jong (1997) and Van Benthem and Keenan (2004) used in the active set method to run the block principal pivoting algorithm efficiently for multiple right hand side columns in the normal equation; in other words, when multiple columns of $\hat{\boldsymbol{y}}_T(h)$ exist.

3.2 Projection-based methods

This class includes a set of methods that can incorporate multiple active constraints at each iteration, using the gradient information. The gradient projection method that belongs to this class is used commonly in practice, and is extremely useful for obtaining a set of non-negative reconciled forecasts for hierarchical and grouped time series because, in this context, (i) the constraints defined are simple, and (ii) the matrices involved are large and/or sparse (Nocedal and Wright, 2006). These facts facilitate the efficient performance of the computations, because iterations of the algorithm mainly involve projections onto the feasible set and the determination of a (steepest) descent direction.

The following sections review the details of two algorithms that are of interest in this research. The first algorithm combines the gradient projection with a conjugate gradient method in order to solve the non-negative least squares problem (Nocedal and Wright, 2006). The second algorithm uses the scaled gradient projection method, which differs from standard gradient projection methods by the presence of a scaling matrix that multiplies the gradient. The numerical experiments have shown that this method is also effective at handling large scale problems for a proper scaling matrix (Bonettini, Zanella, and Zanni, 2009).

There are also various other methods that fall into this class. For a detailed discussion, the interested reader is referred to the work of Chen and Plemmons (2009) and the references provided therein.

Gradient projection + conjugate gradient approach

Each iteration of the algorithm is designed to follow two main steps. In the first step, the current feasible solution $\ddot{\boldsymbol{b}}$ is updated by searching along the steepest direction; in other words, the direction $-\ddot{\boldsymbol{g}}$ from $\ddot{\boldsymbol{b}}$. If the lower bound of the inequality constraints (i.e., $\mathbf{0}$) is encountered before a minimizer is found along the line, the search direction is "bent" to ensure that it remains feasible. The search is continued along the resulting piecewise-linear path in order to locate the first local minimizer of the objective function, q. This point is referred to as the Cauchy point, and denoted by $\boldsymbol{b}^c$. Based on the Cauchy point, it is possible to define a set of constraints that are active at this point. Hence, in the second step, a subproblem is solved by fixing the constraints of the active set to zero. The following sections outline the procedure involved in each step in more detail.

Step 1: Computation of the Cauchy point The search along the steepest direction from the current point $\ddot{\boldsymbol{b}}$ is denoted by $\ddot{\boldsymbol{b}}(s)$ and obtained by simply projecting the search direction at $\ddot{\boldsymbol{b}}$ onto the feasible region as follows:

$$\ddot{\boldsymbol{b}}(s) = (\ddot{\boldsymbol{b}} - s\ddot{\boldsymbol{g}})_+,$$

where $s \geq 0$ and $(\cdot)_+$ is the projection onto the non-negative orthant, defined as $(x)_+ = \max(0, x)$. It is a piecewise-linear function of s, and the objective function along this path $q[\ddot{\boldsymbol{b}}(s)]$ is quadratic in s on every linear piece. Therefore, a suitable value for s at each iteration can be found by searching along the path explicitly. The first local minimizer of $q[\ddot{\boldsymbol{b}}(s)]$ (in other words, the Cauchy point) is identified by examining each line segment that can be formed from $\ddot{\boldsymbol{b}}(s)$. The procedure is given below.

1. For each component, identify the value of s that reaches the lower bound of zero when searching along the steepest direction. The value for each component is computed as

$$\ddot{s}_i = \begin{cases} \ddot{b}_i/\ddot{g}_i, & \text{if } \ddot{g}_i > 0, \\ \infty, & \text{otherwise.} \end{cases}$$

For any s, the components of $\ddot{\boldsymbol{b}}(s)$ that retain feasibility are defined as

$$\ddot{b}_i(s) = \begin{cases} \ddot{b}_i - s\ddot{g}_i, & \text{if } s \leq \ddot{s}_i, \\ \ddot{b}_i - \ddot{s}_i\ddot{g}_i, & \text{otherwise.} \end{cases}$$

2. To obtain the first local minimizer along $\ddot{\boldsymbol{b}}(s)$, any duplicates or zeros are removed from the collection of $\ddot{s}_i$ values. The resulting unique set of breakpoints is then sorted in ascending order, giving $0 < \ddot{s}_1 < \ddot{s}_2 < \cdots < \ddot{s}_l$. Each interval, $[0, \ddot{s}_1], [\ddot{s}_1, \ddot{s}_2], \ldots, [\ddot{s}_{j-1}, \ddot{s}_j], \ldots, [\ddot{s}_{l-1}, \ddot{s}_l]$, is evaluated sequentially, to locate the first local minimizer.

3. Suppose that we could not locate the local minimizer up to $\ddot{s}_{j-1}$ and moved on to the next interval $[\ddot{s}_{j-1}, \ddot{s}_j]$. The line segment that passes through $[\ddot{\boldsymbol{b}}(\ddot{s}_{j-1}), \ddot{\boldsymbol{b}}(\ddot{s}_j)]$ is given by

$$\ddot{\boldsymbol{b}}(s) = \ddot{\boldsymbol{b}}(\ddot{s}_{j-1}) + (\Delta s)\boldsymbol{p}^{j-1},$$

where

$$\Delta s = s - \ddot{s}_{j-1} \in [0, \ddot{s}_j - \ddot{s}_{j-1}],$$

and

$$p_i^{j-1} = \begin{cases} -\ddot{g}_i, & \text{if } \ddot{s}_{j-1} < \ddot{s}_i, \\ 0, & \text{otherwise.} \end{cases}$$

4. Substituting $\ddot{\boldsymbol{b}}(s)$ into the objective function q and minimizing with respect to Δs yields

$$\Delta s^* = -\frac{f'_{j-1}}{f''_{j-1}},$$

where

$$\begin{aligned} f'_{j-1} &= \ddot{\boldsymbol{b}}(\ddot{s}_{j-1})'\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}\boldsymbol{p}^{j-1} - \hat{\boldsymbol{y}}_T(h)'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}\boldsymbol{p}^{j-1}, \\ f''_{j-1} &= (\boldsymbol{p}^{j-1})'\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}\boldsymbol{p}^{j-1}. \end{aligned}$$

   (a) If $f'_{j-1} > 0$, a local minimizer can be obtained at $s = \ddot{s}_{j-1}$.

   (b) If $\Delta s^* \in [0, \ddot{s}_j - \ddot{s}_{j-1})$, there is a minimizer at $s = \ddot{s}_{j-1} + \Delta s^*$.

   (c) Otherwise, consider the next interval $[\ddot{s}_j, \ddot{s}_{j+1}]$ and continue.

Step 2: Solving the subproblem Once the Cauchy point $\boldsymbol{b}^c$ has been identified, a set of active constraints that corresponds to $\boldsymbol{b}^c$ can be defined as

$$\mathcal{A}\left(\boldsymbol{b}^c\right) = \left\{ i \in (1, 2, \ldots, n) \;\middle|\; b_i^c = 0 \right\}.$$

This reduces the original minimization problem to be of the form

$$\begin{aligned} \min_{\ddot{\boldsymbol{b}}} q(\ddot{\boldsymbol{b}}) \coloneqq \min_{\ddot{\boldsymbol{b}}} \frac{1}{2} \ddot{\boldsymbol{b}}' \boldsymbol{S}' \boldsymbol{\Lambda}_h^{-1} \boldsymbol{S} \ddot{\boldsymbol{b}} - \ddot{\boldsymbol{b}}' \boldsymbol{S}' \boldsymbol{\Lambda}_h^{-1} \hat{\boldsymbol{y}}_T(h) \\ \text{s.t.} \qquad \ddot{b}_i = 0, \quad i \in \mathcal{A}(\boldsymbol{b}^c), \\ \ddot{b}_i \geq 0, \quad i \notin \mathcal{A}(\boldsymbol{b}^c). \end{aligned}$$

As was pointed out by Nocedal and Wright (2006), it is sufficient to find an approximate non-negative solution $\ddot{\boldsymbol{b}}^+$ of the above minimization problem such that $q(\ddot{\boldsymbol{b}}^+) \leq q(\boldsymbol{b}^c)$. In general, the conjugate gradient (CG) algorithm is preferred and more appropriate in this context, as the Jacobian of the above equality constraints and the corresponding null-space basis matrix have simple representations. Specifically, the projected conjugate gradient algorithm is used (as explained in Algorithm 3.2) with a preconditioner. The algorithm terminates as soon as a non-negative solution with a better objective function value is encountered.

Algorithm 3.2 Projected CG algorithm

INITIALIZATION: Assign $\ddot{\boldsymbol{b}}^+ = \boldsymbol{b}^c$ and compute $\boldsymbol{r} = \boldsymbol{S}' \boldsymbol{\Lambda}_h^{-1} \boldsymbol{S} \ddot{\boldsymbol{b}}^+ - \boldsymbol{S}' \boldsymbol{\Lambda}_h^{-1} \hat{\boldsymbol{y}}_T(h)$, $\boldsymbol{g}_{\text{proj}} = \boldsymbol{P}_{\text{proj}} \boldsymbol{r}$ and $\boldsymbol{d} = -\boldsymbol{g}_{\text{proj}}$ for a given $\boldsymbol{P}_{\text{proj}}$.

1: REPEAT
2: $\quad \alpha = \boldsymbol{r}' \boldsymbol{g}_{\text{proj}} / \boldsymbol{d}' \boldsymbol{S}' \boldsymbol{\Lambda}_h^{-1} \boldsymbol{S} \boldsymbol{d}$.
3: $\quad \ddot{\boldsymbol{b}}^+ = \ddot{\boldsymbol{b}}^+ + \alpha \boldsymbol{d}$.
4: $\quad \boldsymbol{r}^+ = \boldsymbol{r} + \alpha \boldsymbol{S}' \boldsymbol{\Lambda}_h^{-1} \boldsymbol{S} \boldsymbol{d}$.
5: $\quad \boldsymbol{g}^+ = \boldsymbol{P}_{\text{proj}} \boldsymbol{r}^+$.
6: $\quad \beta = (\boldsymbol{r}^+)' \boldsymbol{g}^+ / \boldsymbol{r}' \boldsymbol{g}_{\text{proj}}$.
7: $\quad \boldsymbol{d} = -\boldsymbol{g}^+ + \beta \boldsymbol{d}$.
8: $\quad \boldsymbol{g}_{\text{proj}} = \boldsymbol{g}^+$.
9: $\quad \boldsymbol{r} = \boldsymbol{r}^+$.
10: UNTIL $\ddot{\boldsymbol{b}}^+$ is feasible and $q(\ddot{\boldsymbol{b}}^+) \leq q(\boldsymbol{b}^c)$ is satisfied.

In Algorithm 3.2, $\boldsymbol{P}_{\text{proj}}$ denotes the projection matrix involving a diagonal precondition matrix. It is defined as

$$\boldsymbol{P}_{\text{proj}} = \begin{cases} \left[\left(\boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}\right)_{ii}\right]^{-1}, & \text{if } i \notin \mathcal{A}(\boldsymbol{b}^c), \\ 0, & \text{otherwise.} \end{cases}$$

The gradient projection method using the Cauchy point to solve the non-negative quadratic programming problem is summarized in Algorithm 3.3.

Algorithm 3.3 Gradient projection based on the Cauchy point

INITIALIZATION: Choose a feasible initial solution $\ddot{\boldsymbol{b}}^0$.

1: FOR k in 0, 1, 2, ... DO
2: IF $\ddot{\boldsymbol{b}}^k$ satisfies the KKT conditions THEN
3: Terminate the algorithm. $\check{\boldsymbol{b}} = \ddot{\boldsymbol{b}}^k$ is the unique global solution.
4: ELSE
5: Find the Cauchy point $\boldsymbol{b}^c$ using $\ddot{\boldsymbol{b}}^k$.
6: Use $\boldsymbol{b}^c$ to find an approximate feasible solution $\ddot{\boldsymbol{b}}^+$ that satisfies $q(\ddot{\boldsymbol{b}}^+) \leq q(\boldsymbol{b}^c)$.
7: $\ddot{\boldsymbol{b}}^{k+1} = \ddot{\boldsymbol{b}}^+$.
8: ENDIF
9: ENDFOR

Scaled gradient projection

This section reviews the details of the diagonally scaled gradient projection algorithm proposed by Bonettini, Zanella, and Zanni (2009). The algorithm propagates by determining the descent direction at each iteration, based on the current feasible solution, step length and scaling matrix. The solution vector is then adjusted along this direction using a non-monotone line search that does not guarantee a decrease in the objective function value at each iteration. This is in order to increase the likelihood of locating a global optimum in practice (Birgin, Martínez, and Raydan, 2003). The main steps are given in Algorithm 3.4.

Selection of the scaling matrix and step length

The scaling matrix should be selected carefully so as to improve the convergence rate of the algorithm while avoiding any unnecessary computational costs at each iteration. One possible choice is to use a diagonal matrix with entries that approximate the diagonal elements of the inverse of the Hessian matrix $\nabla^2 q(\boldsymbol{b}) = \boldsymbol{S}'\boldsymbol{\Lambda}_h^{-1}\boldsymbol{S}$. One specific choice is to assume $\boldsymbol{D} = \boldsymbol{D}_k =$

Algorithm 3.4 Diagonally scaled gradient projection algorithm

INITIALIZATION: Choose a feasible initial solution $\ddot{\boldsymbol{b}}^0$. Set the parameters $\eta, \theta \in (0,1)$, $0 < \alpha_{\min} < \alpha_{\max}$, and a positive integer M.

1: FOR k in 0, 1, 2, ... DO
2: Choose the parameter $\alpha_k \in [\alpha_{\min}, \alpha_{\max}]$ and the diagonal scaling matrix $\boldsymbol{D}_k \in \boldsymbol{\mathcal{D}}$. $\boldsymbol{\mathcal{D}}$ denotes the set of diagonal positive definite matrices.
3: Projection: $\boldsymbol{z}^k = [\ddot{\boldsymbol{b}}^k - \alpha_k \boldsymbol{D}_k \ddot{\boldsymbol{g}}(\boldsymbol{b}^k)]_+$.
4: IF $\boldsymbol{z}^k = \ddot{\boldsymbol{b}}^k$ THEN
5: Terminate the algorithm. $\check{\boldsymbol{b}} = \boldsymbol{b}^k$ is the unique global minimum.
6: ELSE
7: Descent direction: $\boldsymbol{d}^k = \boldsymbol{z}^k - \ddot{\boldsymbol{b}}^k$.
8: Set $\lambda_k = 1$ and $q_{\max} = \max_{j \in 0,1,\ldots,\min(k,M-1)} q(\ddot{\boldsymbol{b}}^{k-j})$
9: Backtracking loop:
IF $q(\ddot{\boldsymbol{b}}^k + \lambda_k \boldsymbol{d}^k) \leq q_{\max} + \eta \lambda_k [\ddot{\boldsymbol{g}}(\ddot{\boldsymbol{b}}^k)]' \boldsymbol{d}^k$ THEN
Go to line 10.
ELSE
Set $\lambda_k = \theta \lambda_k$ and go to line 9.
ENDIF
10: Set $\ddot{\boldsymbol{b}}^{k+1} = \ddot{\boldsymbol{b}}^k + \lambda_k \boldsymbol{d}^k$.
11: ENDIF
12: ENDFOR

$\text{diag}(d_1, d_2, \ldots, d_{m_K})$ such that

$$d_i = \left[\left(\boldsymbol{S}' \boldsymbol{\Lambda}_h^{-1} \boldsymbol{S} \right)_{ii} \right]^{-1}, \quad i = 1, 2, \ldots, n.$$

This structure is considered throughout this study because the computations are less expensive, especially when $\boldsymbol{\Lambda}_h$ is diagonal. Several other choices for a diagonal scaling matrix are provided by Bonettini, Zanella, and Zanni (2009) and Bertero et al. (2013).

In selecting the step-length parameter for the scaled gradient projection algorithm, Bonettini, Zanella, and Zanni (2009) extended the original ideas of Barzilai and Borwein (1988) that are commonly used for improving the convergence rate of standard gradient methods. The generalized Barzilai & Borwein (BB) rules are

$$\alpha_k^{BB1} = \frac{(\boldsymbol{u}^{k-1})' \boldsymbol{D}^{-1} \boldsymbol{D}^{-1} \boldsymbol{u}^{k-1}}{(\boldsymbol{u}^{k-1})' \boldsymbol{D}^{-1} \boldsymbol{v}^{k-1}},$$

$$\alpha_k^{BB2} = \frac{(\boldsymbol{u}^{k-1})'\boldsymbol{D}\boldsymbol{v}^{k-1}}{(\boldsymbol{v}^{k-1})'\boldsymbol{D}\boldsymbol{D}\boldsymbol{v}^{k-1}},$$

where $\boldsymbol{u}^{k-1} = \ddot{\boldsymbol{b}}^k - \ddot{\boldsymbol{b}}^{k-1}$ and $\boldsymbol{v}^{k-1} = \ddot{\boldsymbol{g}}(\ddot{\boldsymbol{b}}^k) - \ddot{\boldsymbol{g}}(\ddot{\boldsymbol{b}}^{k-1})$. Specifically, these equations reduce to the standard BB rules when $\boldsymbol{D} = \boldsymbol{I}_{m_K}$.

Moreover, it is interesting to note that when $[\ddot{\boldsymbol{b}}^k - \alpha_k \boldsymbol{D}\ddot{\boldsymbol{g}}(\ddot{\boldsymbol{b}}^k)]_+$ is used to generate a descent direction from $\ddot{\boldsymbol{b}}^k$ in hierarchical and grouped time series, the diagonal scaling matrix with $\boldsymbol{\Lambda}_h \propto \boldsymbol{I}_n$ and the generalized BB rules indicate that the whole process reduces to the use of a standard gradient projection method with standard BB rules. Thus, the scaled gradient projection algorithm might not be computationally advantageous for obtaining non-negative reconciled forecasts from the OLS approach proposed by Hyndman, Ahmed, et al. (2011).

Rather than either α^{BB1} or α^{BB2}, Bonettini, Zanella, and Zanni (2009) proposed the use of a hybrid, by alternating between the BB1 and BB2 rules. The alternation is determined based on a threshold τ_k that gets updated at each iteration. This choice seems to avoid applying the same step-length rule in many consecutive iterations. The details of the step-length selection based on the alternating strategy are given in Algorithm 3.5. It combines the ideas of Bonettini, Zanella, and Zanni (2009) and Bertero et al. (2013) with a procedure for estimating the initial step-length parameter.

In estimating the initial guess of α_k, we consider a method similar to that of Figueiredo, Nowak, and Wright (2007), implemented in standard gradient projection methods, within the context of the scaled gradient projection method. It considers the initial value α_0 as the exact minimizer of the objective function along the direction of $\ddot{\boldsymbol{b}}^0 - \alpha \boldsymbol{D}\ddot{\boldsymbol{g}}(\ddot{\boldsymbol{b}}^0)$, if no new constraints are to be satisfied. This involves defining

$$\boldsymbol{p}^0 = (p_i^0) = \begin{cases} [\ddot{\boldsymbol{g}}(\ddot{\boldsymbol{b}}^0)]_i, & \text{if } \ddot{b}_i^0 > 0 \text{ or } [\ddot{\boldsymbol{g}}(\ddot{\boldsymbol{b}}^0)]_i < 0, \\ 0, & \text{otherwise.} \end{cases}$$

Then, the initial guess is estimated as

$$\alpha_0 = \min_{\alpha} q[\ddot{\boldsymbol{b}}^0 - \alpha \boldsymbol{D}\ddot{\boldsymbol{g}}(\ddot{\boldsymbol{b}}^0)],$$

and can be computed analytically using

$$\alpha_0 = \frac{(\boldsymbol{p}^0)'\boldsymbol{D}\boldsymbol{p}^0}{(\boldsymbol{S}\boldsymbol{D}\boldsymbol{p}^0)'\boldsymbol{\Lambda}_h^{-1}(\boldsymbol{S}\boldsymbol{D}\boldsymbol{p}^0)}.$$

Algorithm 3.5 Step-length alternation for scaled gradient projection

IF $k = 0$ THEN

Define $\alpha_{\max}$, $\alpha_{\min}$, $\tau_1 \in (0,1)$ and a non-negative integer M_α.

Set $\alpha_0 = \min\left\{\alpha_{\max}, \max\left\{\frac{(\boldsymbol{p}^0)'\boldsymbol{D}\boldsymbol{p}^0}{(\boldsymbol{S}\boldsymbol{D}\boldsymbol{p}^0)'\boldsymbol{\Lambda}_h^{-1}(\boldsymbol{S}\boldsymbol{D}\boldsymbol{p}^0)}, \alpha_{\min}\right\}\right\}$.

ELSE

IF $(\boldsymbol{u}^{k-1})'\boldsymbol{D}^{-1}\boldsymbol{v}^{k-1} \leq 0$ THEN

$\alpha_k^1 = \alpha_{\max}$.

ELSE

$\alpha_k^1 = \min\left\{\alpha_{\max}, \max\left\{\alpha_k^{BB1}, \alpha_{\min}\right\}\right\}$.

ENDIF

IF $(\boldsymbol{u}^{k-1})'\boldsymbol{D}\boldsymbol{v}^{k-1} \leq 0$ THEN

$\alpha_k^2 = \alpha_{\max}$.

ELSE

$\alpha_k^2 = \min\left\{\alpha_{\max}, \max\left\{\alpha_k^{BB2}, \alpha_{\min}\right\}\right\}$.

ENDIF

IF $\alpha_k^2/\alpha_k^1 \leq \tau_k$ THEN

$\alpha_k = \min_{j \in \max\{1, k+1-M_\alpha\}, \ldots, k} \alpha_j^2$.

$\tau_{k+1} = 0.9 \cdot \tau_k$.

ELSE

$\alpha_k = \alpha_k^1$.

$\tau_{k+1} = 1.1 \cdot \tau_k$.

ENDIF

ENDIF

Extreme values for α_0 are avoided by bounding it to be in the interval $[\alpha_{\min}, \alpha_{\max}]$.

Selection of tuning parameters

This section highlights the roles of the remaining tuning parameters and the optimized values that are obtained by testing several applications.

- η and θ control the amount by which the objective function should be decreased and the number of backtracking reductions to be performed, respectively. The values of $\eta = 10^{-4}$ and $\theta = 0.4$ have been used in order to get a sufficiently large step size with fewer reductions (Bonettini, Zanella, and Zanni, 2009).

- $\alpha_{\min}$ and $\alpha_{\max}$ are the lower and upper bounds of the step-length parameter α_k, in order to avoid unnecessary extreme values. Even though a large range is defined for the BB-type

rules in practice, Bertero et al. (2013) found the interval $(10^{-5}, 10^{5})$ to be suitable for the generalized BB rules.

- τ_1 is the initial switching condition that activates the step-length alternation strategy, and a value of 0.5 is suitable in many applications (Bertero et al., 2013; Bonettini, Zanella, and Zanni, 2009).

- Frassoldati, Zanni, and Zanghirati (2008) showed that the use of $\min_{j \in 0,1,\ldots,M_\alpha} \alpha_{k-j}^{BB2}$ in the standard gradient methods for unconstrained quadratic minimization problems can be effective for increasing the ability of the BB1 rule to approximate the smallest eigenvalues of the inverse of the Hessian matrix in subsequent iterations. In the context of scaled gradient projection methods, Bonettini, Zanella, and Zanni (2009) set $M_\alpha = 3$.

- M determines the monotone ($M = 1$) or non-monotone ($M > 1$) line search to be performed in the backtracking loop. This value should not be too large, as the decrease in the objective function is difficult to control, and is set to 10 here, as was done by Bonettini, Zanella, and Zanni (2009).

4 Monte Carlo experiments

This section aims to show the practical usefulness of the aforementioned algorithms for obtaining a set of non-negative reconciled forecasts using a few special cases of MinT. For the sake of simplicity, the OLS and WLS based on structural weights (WLS_s) approaches are considered (Wickramasuriya, Athanasopoulos, and Hyndman, 2018). The computational efficiency of these algorithms is evaluated over a series of hierarchies, ranging in size from small to large. We study the behaviours of two possible choices of the initial solution: (i) base forecasts at the bottom level; and (ii) the unconstrained OLS or WLS_s forecasts. The latter choice can sometimes be a better alternative than the former, as it is aggregate consistent, and computationally less demanding. For the projection-based approaches, the unconstrained OLS or WLS_s forecasts are projected on to the non-negative orthant, as these algorithms need a feasible initial solution. However, the block principal pivoting algorithm can use the unconstrained solution in its original condition. The acronyms used to distinguish the algorithms and their variations of interest are listed in Table 1.

Sections 4.1 and 4.2 demonstrate respectively for OLS and WLS_s, the simulation design, the numbers of negative reconciled forecasts observed at the bottom level and the performances of the algorithms. All experiments are performed in R on a Linux machine equipped with

Table 1: *Acronyms for non-negative least squares algorithms.*

Algorithm	Notation
Scaled gradient projection	
Step-length $\alpha_k = \alpha_k^1$	BB1
Step-length $\alpha_k = \alpha_k^2$	BB2
Step-length α_k alternates	ABB
Gradient projection + conjugate gradient	PCG
Block principal pivoting	BPV

Note: α_k^1 and α_k^2 are defined in Algorithm 3.5.

a 3.20GHz Intel Quad-core processor and 8GB memory. A parallel computing environment with two workers is established by using the `doParallel` (Revolution Analytics and Weston, 2015a) and `foreach` (Revolution Analytics and Weston, 2015b) packages in R to parallelize the procedure of computing non-negative reconciled forecasts for different forecast horizons.

4.1 Non-negativity in the OLS approach

Initially, we consider a hierarchy with $K = 1$ level, having three series at the bottom level. The base forecast for the most aggregated series in the hierarchy is generated from a uniform distribution on the interval $(e^K, 1.2e^K)$, where K is the number of levels in the hierarchy. This is then disaggregated to the bottom level based on a set of proportions that sum to one. Specifically, a set of values is chosen from a gamma distribution, with the shape and scale parameters set to two, and the values are normalized to ensure that they sum to one. Noise is then added to the series at the aggregated levels to make them aggregate-inconsistent. If any of the base forecasts become negative after the noise is added, they are set to zero in order to ensure that all base forecasts in the hierarchy are strictly non-negative. The whole procedure is repeated until there is at least one negative reconciled forecast at the bottom level, and six of these sets with negative reconciled forecasts are used to denote a forecast horizon of length 6. The number of levels in the hierarchy is increased gradually to construct much larger hierarchies, by adding three nodes below each of the bottom-level nodes in the preceding hierarchy. Table 2 presents the structure of each hierarchy constructed and the number of negative reconciled forecasts at the bottom level for each forecast horizon. Each hierarchy constructed contains approximately 10% of negative reconciled forecasts at the bottom level. These are then revised using the algorithms discussed in Section 3, in order to obtain a set of non-negative reconciled forecasts.

Table 2: *The structure of each hierarchy generated and the number of negative reconciled forecasts that result from the OLS approach.*

			Forecast horizon (h)					
K	m	m_K	1	2	3	4	5	6
1	4	3	1	1	1	1	1	1
2	13	9	1	1	1	2	1	2
3	40	27	1	1	3	2	1	1
4	121	81	1	4	7	2	1	5
5	364	243	9	12	11	15	4	18
6	1093	729	50	45	36	39	37	48
7	3280	2187	137	147	138	137	143	134
8	9841	6561	532	476	572	490	524	520
9	29524	19683	1604	1920	1589	1551	1778	1662
10	88573	59049	6087	6106	5888	5199	5243	6193
11	265720	177147	21696	19439	20935	19318	19121	21674
12	797161	531441	64668	63866	71347	68648	63031	69600

K: number of levels.
m: total number of series.
n: number of series at the bottom level.

Tables 3 and 4 present the numbers of iterations and average computational times in seconds (s) that are required to reach the KKT optimality conditions given in Eq. (4) when the base and (projected) unreconciled OLS forecasts, respectively, are used as the initial solution. Cases where an algorithm reaches the maximum number of iterations considered (10^4) are marked with an asterisk, and the average time corresponding to that number of iterations is given. The first row in each hierarchy of Table 3 gives the computational time required to produce the unconstrained OLS reconciled forecasts, which is always the best time, because the weights matrix computes only ones for all forecast horizons. The bold entries identify the non-negative algorithms with the best computational performances.

The main conclusion that can be drawn from these sets of results is that the BPV algorithm always has the best computational performance. This is due in part to the alternative analytical representation proposed for MinT (Wickramasuriya, Athanasopoulos, and Hyndman, 2018). In addition, it should be noted that the computational performance of BPV algorithm depends strongly on how often the full exchange rule fails and Murty's method or the back-up rule has to be activated. Interestingly, the back-up rule was inactive for all experiments carried out in this section. This is also observed in the tests performed by Kim and Park (2011). However, there is no theoretical justification for this, nor do we have conditions under which we know that

the back-up rule is always inactive. Hence, the performance of the algorithm will be affected slightly if it is activated in a certain application.

The second best timing is achieved by the PCG algorithm. Unfortunately, it is inefficient for larger hierarchies, as locating the Cauchy point can be time consuming. Of the two initial solutions considered, the (projected) unconstrained OLS is a good choice, as expected. The scaled gradient projection algorithm performed the worst. This is not surprising, as it reduces to using the standard gradient projection algorithm for the OLS approach.

Table 3: *Computational efficiency of the non-negative forecast reconciliation from the OLS approach using base forecasts as the initial solution.*

				Forecast horizon (h)						
K	m	m_K		1	2	3	4	5	6	Time (s)
1	4	3	OLS							0.01
			BB1	3	3	3	3	3	3	**0.03**
			BB2	492	330	357	644	495	300	1.49
			ABB	3	3	3	3	3	3	**0.03**
			PCG	1	1	1	1	1	1	**0.03**
2	13	9	OLS							0.01
			BB1	24	47	519	697	506	541	1.73
			BB2	471	540	434	691	442	514	2.46
			ABB	495	435	431	659	35	517	2.34
			PCG	1	2	1	2	2	2	**0.11**
3	40	27	OLS							0.01
			BB1	395	280	496	279	419	416	1.86
			BB2	377	345	769	228	433	392	2.38
			ABB	268	299	476	272	333	416	1.69
			PCG	2	2	3	3	2	1	**0.15**
4	121	81	OLS							0.01
			BB1	510	498	10^{4*}	2135	566	429	22.03
			BB2	839	399	10^{4*}	1820	557	552	22.46
			ABB	384	387	10^{4*}	7700	510	518	21.85
			PCG	4	6	9	4	3	7	**0.30**
5	364	243	OLS							0.01
			BB1	10^{4*}	7156	2748	10^{4*}	1324	3324	45.20
			BB2	7997	10^{4*}	820	10^{4*}	2075	8831	64.63
			ABB	10^{4*}	4381	976	6114	2948	2524	30.44
			PCG	9	11	12	15	7	16	**0.53**
6	1093	729	OLS							0.02
			BB1	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	85.59
			BB2	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	74.48
			ABB	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	74.22
			PCG	49	41	44	43	41	44	**1.32**
7	3280	2187	OLS							0.03
			BB1	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	88.23
			BB2	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	76.98
			ABB	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	77.53
			PCG	133	139	168	113	136	137	**4.48**
8	9841	6561	OLS							0.09
			PCG	461	445	440	384	423	374	**26.25**
9	29524	19683	OLS							0.19
			PCG	1296	1663	1206	1359	1625	1388	**227.40**
10	88573	59049	OLS							0.55
			PCG	5792	4046	3684	3551	4385	4100	**3274.80**

Notes: OLS defines the unconstrained OLS approach.
The computational time is averaged over 50 replications for $K = 1$ to $K = 9$, but only 10 for $K = 10$, as the computational time is considerable.
Only PCG is performed up to $K = 10$, due to the high computational time.

Table 4: *Computational efficiency of the non-negative forecast reconciliation from the OLS approach using (projected) unconstrained OLS forecasts as the initial solution.*

				Forecast horizon (h)						
K	m	m_K		1	2	3	4	5	6	Time (s)
1	4	3	BB1	1	1	1	1	1	1	**0.02**
			BB2	1	1	1	1	1	1	**0.02**
			ABB	1	1	1	1	1	1	0.03
			PCG	1	1	1	1	1	1	0.03
			BPV	1	1	1	1	1	1	0.06
2	13	9	BB1	26	24	38	15	12	551	0.77
			BB2	439	401	452	752	1358	480	3.10
			ABB	451	27	430	692	11	488	1.75
			PCG	1	1	1	1	1	1	**0.07**
			BPV	1	1	1	1	1	1	**0.07**
3	40	27	BB1	40	40	551	325	562	461	1.66
			BB2	332	298	792	285	403	391	2.37
			ABB	395	323	548	274	339	383	1.91
			PCG	1	1	2	1	1	1	0.12
			BPV	1	1	2	1	1	1	**0.09**
4	121	81	BB1	444	379	10^{4*}	1905	505	581	21.94
			BB2	412	376	10^{4*}	2018	513	506	21.87
			ABB	339	348	10^{4*}	2833	535	448	22.09
			PCG	1	1	2	2	1	1	0.20
			BPV	1	2	3	2	1	2	**0.12**
5	364	243	BB1	10^{4*}	10^{4*}	625	10^{4*}	2315	2726	52.32
			BB2	5019	10^{4*}	807	10^{4*}	1046	5305	56.74
			ABB	10^{4*}	10^{4*}	1715	10^{4*}	529	7930	64.04
			PCG	1	1	1	4	1	3	0.39
			BPV	1	1	1	2	1	2	**0.12**
6	1093	729	BB1	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	88.44
			BB2	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	75.29
			ABB	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	75.79
			PCG	38	9	13	11	15	18	1.04
			BPV	2	2	2	2	3	2	**0.19**
7	3280	2187	BB1	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	93.11
			BB2	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	77.30
			ABB	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	10^{4*}	85.74
			PCG	49	49	71	54	64	43	2.64
			BPV	2	2	3	3	3	3	**0.31**
8	9841	6561	PCG	255	201	287	218	196	213	17.58
			BPV	3	3	3	2	3	2	**1.13**
9	29524	19683	PCG	699	781	704	630	807	679	182.34
			BPV	3	3	3	3	3	3	**2.80**
10	88573	59049	PCG	2622	2501	2166	2448	2081	2583	2302.11
			BPV	4	4	4	4	4	3	**9.33**
11	265720	177147	BPV	4	4	4	4	4	4	**28.99**
12	797161	531441	BPV	4	4	4	4	4	5	**98.57**

Notes: The computational time is averaged over 50 replications for $K = 1$ to $K = 9$, but only 10 for $K > 9$, as the computational time is considerable.
Only PCG is performed up to $K = 10$, due to the high computational time.

4.2 Non-negativity in the WLS$_s$ approach

The simulation design of Section 4.1 is biased against the scaled gradient projection algorithms. We therefore reassess the performances of the competing algorithms by changing the structure of the hierarchy and using WLS with structural weights as the forecast reconciliation approach.

As with the previous simulation setup, we consider a hierarchy with $K = 1$ level and three series at the bottom level. The base forecast for the most aggregated series in the hierarchy is generated from a uniform distribution on the interval $(1.5e^K, 2e^K)$, where K is the number of levels in the hierarchy. This is then disaggregated to the bottom level based on a set of proportions that sum to one. Specifically, a set of values is chosen from a gamma distribution with the shape and scale parameters set at two, and the values are normalized to ensure that they sum to one. Noise is also added to the series at the aggregated level to make them aggregate-inconsistent. If any of the base forecasts become negative after adding the noise, they are set to zero to ensure that all base forecasts in the hierarchy are strictly non-negative. The whole procedure is repeated until there is at least one negative reconciled forecast at the bottom level, and six of these sets containing negative reconciled forecasts are obtained in order to represent a forecast horizon of length 6. The number of levels in the hierarchy is increased gradually to construct much larger hierarchies, by adding a mixture of three and four nodes to the bottom-level nodes in the preceding hierarchy. Table 5 presents the structure of each hierarchy constructed and the number of negative reconciled forecasts at the bottom level for each forecast horizon. Each hierarchy contains approximately 10% of negative reconciled forecasts at the bottom level.

Tables 6 and 7 present the numbers of iterations and the average computational times in seconds (s) that are required to reach the KKT optimality conditions when the base and projected (unconstrained) WLS$_s$ forecasts, respectively, are used as the initial solution. Cases where an algorithm reaches the maximum number of iterations (10^4) are marked with an asterisk, and the average computational time corresponding to that number of iterations is given. The first row in each hierarchy of Table 6 gives the computational times required to produce the unconstrained WLS$_s$ reconciled forecasts. As with OLS, this always shows the best timing. The bold entries identify the non-negative algorithms with the best computational performances.

The overall pattern that is observed in these two tables is quite similar to that in Section 4.1. However, for moderate-sized hierarchies, the scaled gradient projection algorithms based on BB rules occasionally terminate with fewer iterations than the cut-off, unlike the previous simulation setup. Nevertheless, the use of this algorithm is still problematic for large hierarchies.

Table 5: *The structure of each hierarchy generated and the numbers of negative reconciled forecasts that result from the WLS_s approach.*

			Forecast horizon (h)					
K	m	m_K	1	2	3	4	5	6
1	4	3	1	1	1	1	1	1
2	14	10	1	1	1	2	1	1
3	49	35	1	1	1	1	2	1
4	171	122	1	1	3	5	2	4
5	598	427	13	10	10	9	6	7
6	2092	1494	62	38	60	56	44	43
7	7321	5229	211	229	255	270	222	203
8	25622	18301	1128	1166	1026	1268	1285	1186
9	89675	64053	4738	5619	6244	4812	5779	5637
10	249808	160133	17744	17790	16023	17261	19462	17258
11	650141	400333	36271	47845	44040	40790	47879	38753
12	1650974	1000833	105462	84998	76817	102130	95530	80090

Note: For $K > 9$, either two or three nodes are added to each of the bottom-level nodes in the preceding hierarchy.

In summary, both simulations performed illustrate that the block principal pivoting algorithm is computationally efficient in obtaining a set of non-negative reconciled forecasts. The projected conjugate gradient algorithm is the second best. To further examine the superiority of these competing algorithms on real applications, an extensive case study is provided in Section 5.

Table 6: *Computational efficiency of the non-negative forecast reconciliation from the WLS_s approach using base forecasts as the initial solution.*

				Forecast horizon (h)						
K	m	m_K		1	2	3	4	5	6	Time (s)
1	4	3	WLS_s							0.01
			BB1	3	3	3	3	3	3	**0.03**
			BB2	2682	2830	2648	833	2035	2347	11.05
			ABB	3	3	3	3	3	3	0.04
			PCG	1	1	1	1	1	1	0.04
2	14	10	WLS_s							0.01
			BB1	115	16	293	471	354	522	1.29
			BB2	396	302	285	469	443	579	1.58
			ABB	342	265	23	481	22	497	1.52
			PCG	1	2	1	1	1	2	**0.10**
3	49	35	WLS_s							0.01
			BB1	320	542	462	557	712	546	2.22
			BB2	363	655	438	479	748	444	2.10
			ABB	276	461	391	392	737	423	1.86
			PCG	2	2	2	1	1	2	**0.17**
4	171	122	WLS_s							0.01
			BB1	556	511	486	575	494	680	2.64
			BB2	550	445	393	386	586	540	2.17
			ABB	456	496	389	480	364	498	2.13
			PCG	2	2	2	2	2	2	**0.29**
5	598	427	WLS_s							0.01
			BB1	390	426	446	428	366	440	2.36
			BB2	376	389	399	355	322	364	1.84
			ABB	375	357	379	352	359	270	1.94
			PCG	2	3	2	3	2	3	**0.33**
6	2092	1494	WLS_s							0.02
			BB1	1766	493	630	10^{3*}	462	493	37.00
			BB2	1390	473	435	7985	476	336	29.51
			ABB	1239	354	419	7803	354	382	29.11
			PCG	5	6	6	6	4	6	**0.51**
7	7231	5229	WLS_s							0.06
			BB1	1215	1417	10^{4*}	10^{4*}	734	567	104.33
			BB2	597	890	10^{4*}	10^{4*}	570	432	97.06
			ABB	1020	851	10^{4*}	10^{4*}	614	369	103.16
			PCG	9	8	8	9	7	7	**2.09**
8	25622	18301	WLS_s							0.19
			PCG	13	14	13	13	13	12	**19.08**
9	89675	64053	WLS_s							0.48
			PCG	16	16	16	17	17	21	**244.68**
10	249808	160133	WLS_s							1.43
			PCG	19	20	19	17	20	20	**1660.12**

Notes: WLS_s defines the unconstrained WLS_s approach.
The computational time is averaged over 50 replications for $K = 1$ to $K = 9$, but only 10 for $K = 10$, as the computational time is considerable.
Only PCG is performed up to $K = 10$, due to the high computational time.

Table 7: *Computational efficiency of the non-negative forecast reconciliation from the WLS_s approach using (projected) unconstrained WLS_s forecasts as the initial solution.*

				Forecast horizon (h)						
K	m	m_K		1	2	3	4	5	6	Time (s)
1	4	3	BB1	1	1	1	1	1	1	**0.03**
			BB2	1	1	1	1	1	1	**0.03**
			ABB	1	1	1	1	1	1	**0.03**
			PCG	1	1	1	1	1	1	**0.03**
			BPV	1	1	1	1	1	1	0.06
2	14	10	BB1	300	189	258	482	358	18	1.11
			BB2	263	179	283	514	282	449	1.47
			ABB	287	17	22	512	357	35	0.88
			PCG	1	1	1	1	1	2	0.09
			BPV	1	1	1	1	1	2	**0.08**
3	49	35	BB1	275	496	366	454	747	437	2.12
			BB2	366	442	400	416	710	427	2.14
			ABB	290	347	297	374	621	388	1.80
			PCG	1	1	1	1	1	1	0.13
			BPV	1	1	1	1	1	1	**0.07**
4	171	122	BB1	572	428	411	448	508	573	2.45
			BB2	580	328	440	450	453	529	2.23
			ABB	515	312	423	431	368	12	2.02
			PCG	1	1	1	1	1	1	0.17
			BPV	1	1	1	1	1	1	**0.09**
5	598	427	BB1	336	429	416	434	443	364	2.21
			BB2	448	337	465	408	329	283	2.09
			ABB	455	380	363	334	419	368	2.27
			PCG	1	1	1	1	1	1	0.22
			BPV	1	1	1	1	2	1	**0.12**
6	2092	1494	BB1	1920	446	545	9086	496	379	33.45
			BB2	1367	459	486	8081	425	345	29.68
			ABB	1214	282	424	5947	390	340	22.32
			PCG	3	2	2	2	2	1	0.36
			BPV	2	2	1	2	2	1	**0.17**
7	7321	5229	BB1	1260	1247	10^{4*}	10^{4*}	885	520	103.30
			BB2	719	725	10^{4*}	10^{4*}	620	463	95.14
			ABB	879	663	10^{4*}	10^{4*}	766	374	101.72
			PCG	3	4	6	4	3	3	0.68
			BPV	2	2	2	2	2	2	**0.57**
8	25622	18301	PCG	6	8	8	7	6	5	3.18
			BPV	2	2	2	2	3	2	**1.76**
9	89675	64053	PCG	12	11	14	11	10	14	35.08
			BPV	3	3	3	3	3	3	**6.45**
10	249808	160133	PCG	16	16	18	17	17	16	250.22
			BPV	3	3	3	3	3	3	**21.00**
11	650141	400333	PCG	18	19	21	22	18	21	1597.10
			BPV	3	3	3	3	3	3	**56.84**
12	1650974	1000833	BPV	3	3	3	3	3	3	**3247.09**

Notes: The computational time is averaged over 50 replications for $K = 1$ to $K = 9$, but only 10 for $K > 9$, as the computational time is considerable.
Only PCG is performed up to $K = 11$, due to the high computational time.

5 Non-negative reconciled forecasts for Australian domestic tourism flows

This section evaluates the impact of imposing the strict non-negativity constraints on the forecast reconciliation approaches, using the Australian domestic tourism flows as a case study. The grouped structure consisting of 555 series is considered. A detailed description of the structure and a comprehensive analysis of the forecast performances of different reconciliation approaches are given in Wickramasuriya, Athanasopoulos, and Hyndman (2018). However, the base and reconciled forecasts in these applications are allowed to take any values on the real line; in other words, they have not explicitly restricted them to be non-negative, even though the original data are. Hence, this section considers the non-negative nature of these sets of forecasts, as obtaining a set of non-negative reconciled forecasts requires the base forecasts to be non-negative too. The empirical study performed in Wickramasuriya, Athanasopoulos, and Hyndman (2018) is repeated by log-transforming the original data. For the series that include zeros, a small constant — specifically, half of the minimum positive value — is added and aggregated before transforming the data. Then, ARIMA and ETS models are fitted by minimizing the AICc to obtain the base forecasts using the default settings as implemented in the `forecast` package for R Hyndman (2016) and described in Hyndman and Khandakar (2008). If the forecast densities on the transformed data are symmetric, then the back-transformed base forecasts give the median of the forecast density. Even though this is acceptable in practice, we need to obtain the mean of the forecast densities, as means can be aggregated but medians cannot.

Considering the first three terms of the Taylor series expansion around μ (i.e., the mean on the transformed scale), the mean on the back-transformed scale can be obtained as

$$e^{\mu}\left(1+\frac{1}{2}\sigma^2\right),$$

where σ^2 is the variance on the transformed scale.

These back-transformed means (base forecasts) are then reconciled using OLS, WLS based on variance scaling (WLS_v) and MinT. MinT uses the covariance estimator that shrinks the sample correlation matrix towards an identity matrix; refer to Wickramasuriya, Athanasopoulos, and Hyndman (2018) for further details about covariance estimators. The expanding window forecast evaluation procedure results in a total of 96 1-step-ahead reconciled forecasts, 95 2-step-ahead reconciled forecasts, and so on, down to 85 12-step-ahead reconciled forecasts. These include iterations in which all reconciled forecasts are positive. Table 8 presents the summary

statistics of the negative reconciled forecasts obtained from the ARIMA and ETS base forecasts. Each cell gives (i) the number of iterations reported with negative reconciled forecasts; and the minimum and maximum of (ii) the number of negative reconciled forecasts; (iii) the largest negative value (in thousands); and (iv) the smallest negative value (in thousands) resulting from each of the forecast reconciliation approaches. For example, the first cell in the table illustrates that all 96 iterations of the 1-step-ahead OLS reconciled forecasts include negative forecasts, and the number of negatives can be as small as 5 or as large as 80. The largest negative value varies between -71.8 and -2.33, and the smallest negative value varies between -1.52 and -0.002. It is clear that both the number of negative reconciled forecasts and their magnitudes are considerably large with the OLS approach. The presence of such negative values can degrade the quality of the forecasts in the entire structure.

These forecasts are then revised using the algorithms discussed in Section 3. For the scaled gradient projection and projected conjugate gradient algorithms, these forecasts are projected into the non-negative orthant and used as the initial solution. When the MinT approach is coupled with the block principal pivoting algorithm, a vector of zeros is used as the initial solution. Table 9 summarizes the total computational time (in seconds) of each forecast reconciliation approach for obtaining a set of non-negative reconciled forecasts. For example, the first cell in the table illustrates that the BB1 algorithm takes 948.9 seconds to revise all 96 iterations with negative reconciled forecasts. Cases where an algorithm reaches the maximum number of iterations (10^4) are marked with an asterisk, and the computational time that corresponds to that number of iterations is given.

As was observed in the simulation exercise, the BPV algorithm showed the best computational performance for the OLS approach, while PCG was the second best. The worst performances of all variations of the scaled gradient projection algorithms fit with the previous findings. The WLS_v approaches based on the BB2 and ABB algorithms showed the best performances, challenging the BPV algorithm. The BB1 algorithm terminated with fewer iterations than the cut-off, but it is still inefficient compared to the rest.

Even though MinT resulted in fewer negative reconciled forecasts, the computational time is much larger than the best timings of the WLS_v and even OLS approaches using the BPV and PCG algorithms. This is reasonable because a full variance covariance matrix means that some of the matrix manipulations can result in dense matrices, and hence do not benefit fully from the special matrix multiplication strategies that we considered for the OLS and WLS_v approaches.

The impact of imposing the non-negativity constraints on the estimation procedure is evaluated by comparing the non-negative reconciled forecasts obtained using each reconciliation approach with the negative reconciled forecasts. Table 10 represents the percentage differences in average RMSEs between the reconciled forecasts with non-negatives and negatives at each level in the grouped structure.

It can be seen that the non-negative reconciled forecasts from the OLS approach showed slight gains over the negative reconciled forecasts at each forecast horizon considered (with rare exceptions). These gains are most pronounced at the disaggregated levels. In addition, WLS_v and MinT also showed gains at the most disaggregated level, though the consideration of aggregated levels sometimes introduced slight losses. As these gains or losses are negligible, it can be argued that the non-negativity constraint does not significantly affect the performances of the forecast reconciliation approaches. However, it is useful in real applications for making meaningful managerial decisions and policy implementations.

Table 8: *Summary statistics of the negative reconciled forecasts.*

	ARIMA					
	OLS		WLS_v		MinT	
$h = 1$		96		39		61
	5	80	1	4	1	8
	−71.8	−2.33	−17.5	−0.07	−22.8	−0.05
	−1.52	−2e-3	−13.2	−0.06	−10.9	−4e-3
2		95		37		64
	5	91	1	3	1	9
	−72.3	−2.6	−12.9	−0.09	−23.1	−0.08
	−3.98	−1e-3	−10.7	−0.09	−18.3	−0.01
3		94		37		62
	5	90	1	2	1	8
	−67.3	−1.31	−16.3	−0.04	−14.5	−0.02
	−2.36	−3e-4	−16.3	−0.04	−14.5	−0.02
6		91		40		56
	4	87	1	3	1	9
	−87.4	−0.62	−12.3	−0.11	−16.5	−0.03
	−1.16	−4e-4	−9.91	−0.11	−10.49	−0.03
12		85		44		59
	5	83	1	2	1	8
	−79.5	−2.30	−19.2	−0.18	−26.4	−0.07
	−3.66	−5e-4	−19.2	−0.18	−15.6	−5e-3
	ETS					
	OLS		WLS_v		MinT	
$h = 1$		93		45		43
	1	86	1	3	1	3
	−41.0	−0.03	−13.2	−0.02	−14.9	−0.01
	−6.16	−4e-3	−12.7	−0.02	−9.78	−0.01
2		93		43		45
	1	63	1	3	1	3
	−38.4	−0.19	−13.2	−0.11	−16.9	−0.10
	−7.00	−1e-3	−12.0	−0.07	−8.80	−0.09
3		91		46		46
	1	66	1	3	1	3
	−34.0	−0.12	−11.7	−0.08	−13.4	−0.02
	−4.58	−2e-3	−11.7	−0.08	−8.80	−3e-3
6		90		40		43
	1	75	1	3	1	3
	−31.8	−0.15	−15.2	−3e-4	−13.4	−0.17
	−4.40	−1e-3	−15.2	−3e-4	−8.12	−1e-3
12		84		37		39
	1	117	1	3	1	4
	−170	−0.15	−15.2	−0.07	−17.4	−0.10
	−3.44	−3e-3	−14.1	−0.04	−10.5	−0.05

Notes: Each cell gives (i) the number of iterations reported with negative reconciled forecasts; and the minimum and maximum of (ii) the number of negative reconciled forecasts; (iii) the largest negative value (in thousands); and (iv) the smallest negative value (in thousands).

Table 9: *Computational times (in seconds) of obtaining the non-negative reconciled forecasts.*

		$h=1$	2	3	6	12	$h=1$	2	3	6	12
		ARIMA					ETS				
OLS	BB1	948.9*	910.0*	904.5*	874.5*	818.8*	905.0*	891.9*	874.2*	863.1*	806.8*
	BB2	1012.3*	975.8*	968.6*	936.4*	877.8*	974.1*	948.7*	935.2*	920.3*	862.6*
	ABB	971.1*	953.7*	943.8*	912.4*	858.2*	947.5*	918.4*	897.5*	885.7*	828.6*
	PCG	27.6	27.5	27.6	27.0	24.2	23.0	22.8	21.8	21.4	21.1
	BPV	**3.0**	**2.8**	**2.9**	**2.8**	**2.6**	**2.3**	**2.3**	**2.2**	**2.1**	**2.0**
WLS_v	BB1	26.1	25.1	23.0	25.9	36.0	30.2	28.7	45.1	29.9	25.2
	BB2	**2.0**	**1.9**	**1.9**	**1.9**	**1.8**	**2.0**	**1.9**	**1.9**	**1.9**	**1.7**
	ABB	**2.0**	**1.9**	**1.9**	**1.9**	**1.8**	2.1	2.0	2.0	**1.9**	1.8
	PCG	5.2	5.0	4.8	5.0	5.5	5.7	5.5	5.6	5.0	4.8
	BPV	2.2	2.1	2.1	2.0	2.0	2.2	2.1	2.2	2.0	1.9
MinT	BB1	649.0	678.7	651.1	554.8	665.1	658.9†	536.4	658.0†	536.4	663.0†
	BB2	52.4	51.4	50.9	47.9	46.5	45.2	45.3	44.7	43.7	40.6
	ABB	54.4	55.2	55.2	50.5	48.7	46.8	47.1	46.8	44.7	41.4
	PCG	50.1	49.9	49.1	46.5	44.7	45.1	44.9	44.6	42.9	39.8
	BPV	**38.4**	**37.8**	**37.5**	**36.2**	**33.7**	**38.2**	**37.7**	**37.3**	**36.0**	**33.6**

Notes: The bold entries identify the non-negative algorithms with the best computational performances.
† denotes cases where only few iterations reach the maximum number of iterations.

Table 10: *Impact of the non-negativity constraints on forecast performances.*

	$h=1$	2	3	6	12	$h=1$	2	3	6	12
	ARIMA									
	Australia					Australia by purpose of travel				
OLS	0.0081	**−0.0138**	**−0.0230**	**−0.0094**	**−0.0269**	**−0.0757**	**−0.0503**	0.0125	**−0.0065**	**−0.0157**
WLS_v	0.0091	0.0056	0.0100	0.0006	0.0018	0.0009	0.0027	0.0027	0.0031	0.0008
MinT	**−0.0090**	0.0267	0.0290	0.0427	**−0.0222**	0.0295	0.0143	0.0514	0.0597	0.0279
	States					States by purpose of travel				
OLS	**−0.0560**	**−0.0744**	**−0.0543**	**−0.0159**	**−0.0904**	**−0.5054**	**−0.8173**	**−0.7670**	**−0.8597**	**−0.7745**
WLS_v	0.0045	0.0016	0.0004	**−0.0077**	0.0003	**−0.0009**	**−0.0004**	**−0.0013**	**−0.0023**	**−0.0037**
MinT	0.0366	0.0007	0.0123	0.0140	0.0074	0.0160	0.0120	0.0135	0.0255	**−0.0164**
	Zones					Zones by purpose of travel				
OLS	**−0.1115**	**−0.1038**	**−0.1090**	**−0.1141**	**−0.1489**	**−0.4320**	**−0.6144**	**−0.5723**	**−0.7088**	**−0.6646**
WLS_v	0.0015	**−0.0009**	0.0011	0.0019	**−0.0018**	**−0.0004**	0.0005	0.0001	0.0001	**−0.0071**
MinT	0.0028	**−0.0037**	0.0031	0.0019	**−0.0161**	0.0004	**−0.0044**	0.0032	0.0072	**−0.0234**
	Regions					Regions by purpose of travel				
OLS	**−0.1474**	**−0.1246**	**−0.1164**	**−0.1263**	**−0.1480**	**−0.5573**	**−0.6581**	**−0.5831**	**−0.7258**	**−0.7480**
WLS_v	0.0014	0.0017	0.0008	0.0048	0.0114	**−0.0052**	**−0.0049**	**−0.0044**	**−0.0057**	**−0.0150**
MinT	**−0.0002**	**−0.0056**	0.0002	**−0.0003**	**−0.0012**	**−0.0134**	**−0.0189**	**−0.0104**	**−0.0097**	**−0.0265**
	ETS									
	Australia					Australia by purpose of travel				
OLS	**−0.0236**	**−0.0021**	**−0.0028**	**−0.0116**	**−0.0468**	**−0.0066**	**−0.0088**	**−0.0060**	**−0.0103**	**−0.0222**
WLS_v	0.0052	0.0040	0.0044	0.0059	**−0.0002**	**−0.0018**	**−0.0026**	**−0.0023**	**−0.0043**	**−0.0026**
MinT	0.0089	0.0193	0.0207	0.0203	0.0048	**−0.0012**	**−0.0118**	0.0020	**−0.0105**	**−0.0127**
	States					States by purpose of travel				
OLS	**−0.0333**	**−0.0314**	**−0.0207**	**−0.0510**	**−0.2106**	**−0.0636**	**−0.0423**	**−0.0455**	**−0.0593**	**−0.1031**
WLS_v	**−0.0017**	**−0.0016**	0.0009	0.0033	**−0.0009**	0.0026	0.0022	0.0050	0.0040	0.0011
MinT	0.0129	0.0090	0.0144	0.0165	0.0031	0.0101	0.0060	0.0095	0.0056	**−0.0001**
	Zones					Zones by purpose of travel				
OLS	**−0.0274**	**−0.0213**	**−0.0226**	**−0.0350**	**−0.1622**	**−0.0752**	**−0.0632**	**−0.0703**	**−0.0543**	**−0.0962**
WLS_v	**−0.0084**	**−0.0095**	**−0.0120**	**−0.0133**	**−0.0099**	0.0038	0.0033	0.0028	0.0033	0.0034
MinT	**−0.0008**	**−0.0036**	**−0.0057**	**−0.0071**	**−0.0093**	0.0027	**−0.0018**	0.0004	0.0012	0.0014
	Regions					Regions by purpose of travel				
OLS	**−0.0262**	**−0.0239**	**−0.0297**	**−0.0288**	**−0.1026**	**−0.1249**	**−0.0940**	**−0.0897**	**−0.0912**	**−0.1295**
WLS_v	0.0081	0.0060	0.0040	0.0076	0.0085	**−0.0148**	**−0.0131**	**−0.0112**	**−0.0119**	**−0.0140**
MinT	0.0121	0.0102	0.0083	0.0110	0.0135	**−0.0149**	**−0.0148**	**−0.0131**	**−0.0138**	**−0.0152**

Notes: Each entry shows the percentage difference in average RMSEs between reconciled forecasts with non-negatives and negatives. A negative (positive) entry shows a percentage decrease (increase) in average RMSEs relative to the negative reconciled forecasts. Bold entries identify improvements due to the non-negativity constraints.

6 Conclusions and Discussion

References

Andrew, JJ and Hancewicz, TM (1998). Rapid analysis of Raman image data using two-way multivariate curve resolution. *Applied Spectroscopy* **52**(6), 797–807.

Barzilai, J and Borwein, JM (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis* **8**(1), 141–148.

Bertero, M, Boccacci, P, Prato, M, and Zanni, L (2013). Scaled gradient projection methods for astronomical imaging. *EAS Publications Series* **59**, 325–356.

Birgin, EG, Martínez, JM, and Raydan, M (2003). Inexact spectral projected gradient methods on convex sets. *IMA Journal of Numerical Analysis* **23**(4), 539–559.

Bonettini, S, Zanella, R, and Zanni, L (2009). A scaled gradient projection method for constrained image deblurring. *Inverse Problems* **25**(1).

Bro, R and De Jong, S (1997). A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics* **11**(5), 393–401.

Cantarella, J and Piatek, M (2004). Tsnnls: A solver for large sparse least squares problems with non-negative variables. arXiv: cs/0408029v1.

Chen, D and Plemmons, RJ (2009). "Nonnegativity constraints in numerical analysis". In: *The birth of numerical analysis*. Ed. by A Bultheel and R Cools. New Jersey, NJ: World Scientific Publishing, pp.109–140.

Figueiredo, MAT, Nowak, RD, and Wright, SJ (2007). Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing* **1**(4), 586–597.

Frassoldati, G, Zanni, L, and Zanghirati, G (2008). New adaptive stepsize selections in gradient methods. *Journal of Industrial and Management Optimization* **4**(2), 299–312.

Gemperline, PJ and Cash, E (2003). Advantages of soft versus hard constraints in self-modeling curve resolution problems. Alternating least squares with penalty functions. *Analytical Chemistry* **75**(16), 4236–4243.

Hyndman, RJ (2016). *forecast: Forecasting functions for time series and linear models*. R package version 7.2. http://github.com/robjhyndman/forecast.

Hyndman, RJ, Ahmed, RA, Athanasopoulos, G, and Shang, HL (2011). Optimal combination forecasts for hierarchical time series. *Computational Statistics and Data Analysis* **55**, 2579–2589.

Hyndman, RJ and Khandakar, Y (2008). Automatic Time Series Forecasting : The forecast Package for R. *Journal of Statistical Software* **27**(3).

Hyndman, RJ, Lee, AJ, and Wang, E (2016). Fast computation of reconciled forecasts for hierarchical and grouped time series. *Computational Statistics and Data Analysis* **97**, 16–32.

Júdice, JJ and Pires, FM (1989). Bard-type methods for the linear complementarity problem with symmetric positive definite matrices. *IMA Journal of Management Mathematics* **2**(1), 51–68.

Júdice, JJ and Pires, FM (1994). A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems. *Computers and Operations Research* **21**(5), 587–596.

Kim, J and Park, H (2011). Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing* **33**(6), 3261–3281.

Kostreva, MM (1978). Block pivot methods for solving the complementarity problem. *Linear Algebra and its Applications* **21**, 207–215.

Lawson, CL and Hanson, RJ (1974). *Solving least squares problems*. Prentice-Hall series in automatic computation. New Jersey, NJ: Prentice-Hall.

Murty, KG (1974). Note on a Bard-type scheme for solving the complementarity problem. *Opsearch* **11**(2-3), 123–130.

Nocedal, J and Wright, SJ (2006). *Numerical optimization*. Ed. by TV Mikosch, SI Resnick, and SM Robinson. 2nd ed. Springer series in operations research and financial engineering. New York, NY: Springer Science and Business Media.

Revolution Analytics and Weston, S (2015a). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.10. https://CRAN.R-project.org/package=doParallel.

Revolution Analytics and Weston, S (2015b). *foreach: Provides Foreach Looping Construct for R*. R package version 1.4.3. https://CRAN.R-project.org/package=foreach.

Turlach, BA and Weingessel, A (2013). *quadprog: Functions to solve Quadratic Programming Problems.* R package version 1.5-5. https://CRAN.R-project.org/package=quadprog.

Turlach, BA and Wright, SJ (2015). Quadratic programming. *Wiley Interdisciplinary Reviews: Computational Statistics* **7**(2), 153–159.

Van Benthem, MH and Keenan, MR (2004). Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Journal of Chemometrics* **18**(10), 441–450.

Wang, M and Wang, X (2012). "A jump-start of non-negative least squares solvers". In: *High-performance scientific computing: Algorithms and applications*. Ed. by MW Berry, KA Gallivan, E Gallopoulos, A Grama, B Philippe, F Saied, and Y Saad. London, England: Springer-Verlag, pp.295–310.

Wickramasuriya, SL, Athanasopoulos, G, and Hyndman, RJ (2018). Optimal forecast reconciliation of hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*. To appear.