Accelerated Estimation of Switching Algorithms: The Cointegrated VAR Model and Other Applications

JURGEN A. DOORNIK
Economics Department and Institute for New Economic Thinking at the Oxford Martin School, University of Oxford

ABSTRACT. Restricted versions of the cointegrated vector autoregression are usually estimated using switching algorithms. These algorithms alternate between two sets of variables but can be slow to converge. Acceleration methods are proposed that combine simplicity and effectiveness. These methods also outperform existing proposals in some applications of the expectation–maximization method and parallel factor analysis.

Key words: acceleration, cointegration, EM algorithm, line search, maximization, PARAFAC, switching algorithm

1. Introduction

Estimation of models with cointegration involves non-linear optimization, except in the basic case. Following Johansen & Juselius (1994), the standard approach is to alternate between sets of coefficients. They called this a ‘switching’ algorithm and that terminology is now common in this literature. The numerical literature tends to refer to ‘alternating variables’ algorithms. Subsequently, there have been many examples of switching algorithms in related settings.

The advantage of switching is that each step is easy to implement, and no derivatives are required. Furthermore, the partitioning circumvents the lack of identification that can otherwise occur in these models and which makes it harder to use Newton-type methods. The drawback is that progress is often slow, taking many iterations to converge and occasionally resulting in premature convergence. This paper proposes a modification that accelerates estimation. It amounts to adding a line search, which is both simple and not problem specific. A minimal extra effort results in a substantial speed-up, as well as better quality of convergence.

The focus at first is on the cointegrated vector autoregression (VAR) with I(1) cointegration. This is introduced in Section 2, together with the basic estimation algorithms. Then, Section 3 considers several line search procedures, which are evaluated in Section 4. While we limit ourselves to I(1) models, we note that the proposed line search works in all cointegration models with switching, including our recently developed algorithms for I(2) models (Doornik, 2017).

The expectation–maximization (EM) algorithm has a similar structure to switching in cointegration models and can also be very slow to converge. Acceleration procedures have been proposed for EM, and we compare some of them with our approach in Section 6. The paper finishes with an application to parallel factor models and low-rank matrix application.

2. The I(1) model

The starting point is the VAR with \( p \) dependent variables and \( m \geq 1 \) lags:

\[
y_t = A_1 y_{t-1} + \ldots + A_m y_{t-m} + \Phi x_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{II} \mathbb{N}_p(0, \Omega),
\]
for $p \times 1$ vector $y_t, t = 1, \ldots, T$, with $y_j, j = -m + 1, \ldots, 0$ fixed and given; $x_t$ is a $k$-vector of additional regressors and $\Omega$ is a $p \times p$ positive definite matrix. This model can be rewritten in equilibrium correction form without imposing any restrictions as

$$
\Delta y_t = y_t - y_{t-1} = \Pi y_{t-1} + \Gamma_1 \Delta y_{t-1} + \ldots + \Gamma_{m-1} \Delta y_{t-m+1} + \Phi x_t + \epsilon_t.
$$

(2)

The cointegrated VAR (CVAR) restricts the rank of $\Pi$ to at most $r$ by writing $\Pi = \alpha \beta', \beta' = (\beta_y', \beta_x')$. A specific case is the VAR with a restricted linear trend, where $x_t^R = t$ and $x_t^U = 1$, so $p_1 = p + 1$. This is the specification used below. Gaussian maximum likelihood estimation is via a reduced-rank regression (RRR) after partialling out the unrestricted coefficients $\Psi$ (see, e.g. Johansen & Juselius, 1990 and Johansen, 1995b). The maximum can be determined by solving an eigen problem. This is no longer the case when imposing restrictions on the columns of $\beta$, requiring iterative maximization instead.

Writing $z_{0t}$ for the residuals from regressing $\Delta y_t$ on $w_{2t}$, and $z_{1t}$ for the residuals from regressing $w_{1t}$ on $w_{2t}$, the concentrating model becomes

$$
z_{0t} = \alpha \beta' z_{1t} + \epsilon_t, \quad \epsilon_t \sim \mathcal{IN}_p[0, \Omega].
$$

(4)

with moment matrices

$$
S_{ij} = \frac{1}{T} \sum_{t=1}^{T} z_{it} z_{jt}', \quad i, j = 0, 1.
$$

The concentrating log-likelihood

$$
l_c(\alpha, \beta) = c - \frac{T}{2} \log |\Omega(\alpha, \beta)|,
$$

(5)

based on

$$
\Omega(\alpha, \beta) = \frac{1}{T} \sum_{t=1}^{T} \epsilon_t \epsilon_t', \quad \text{where} \quad \epsilon_t = z_{0t} - \alpha \beta' z_{1t},
$$

is maximized by $\hat{\beta} = (\hat{\beta}_y, \ldots, \hat{\beta}_r)$, where $\hat{\beta}_i$ are the eigenvalues corresponding to the eigenvalues $\hat{\lambda}_i$, in descending order, of the generalized eigenvalue problem:

$$
|\hat{\lambda} S_{11} - S_{10} S_{00}^{-1} S_{01}| = 0.
$$

Then, $\hat{\alpha} = S_{01} \hat{\beta}$. We write RRR($z_{0t}, z_{1t}|z_{2t}$) for the RRR of $z_{0t}$ on $z_{1t}$ corrected for $z_{2t}$. Note that the second moment matrices can be avoided when computing the eigenvalues (Doornik & O’Brien, 2002).
Testing the rank involves non-standard distributions consisting of functionals of Brownian motions (see Johansen, 1995b, Juselius, 2006 and the references therein). Convenient approximations are provided by Doornik (1998).

Only the cointegrating space is estimated, because \( \alpha V^{-1}V\beta' = \alpha\beta' \), for any non-singular \( r \times r \) matrix \( V \). Identification of the cointegrating vectors \( \beta \) is usually guided by suggestions from economic theory about long-run relations. Exact identification imposes no restrictions, but overidentification can be tested with standard \( \chi^2 \) inference provided the rank of \( \Pi \) is kept fixed. Normalization of each column of \( \beta \) may be kept separate from identification; it will be helpful to avoid normalizing on a coefficient that is close to zero.

2.1. The restricted I(1) model

Johansen & Juselius (1994) consider the case where the cointegrating vectors, i.e. the columns of \( \beta \), are split in two sets, each with a common linear restriction. This is estimated by alternating between the two sets of vectors, keeping the other fixed in turn. The procedure was subsequently generalized by Johansen (1995a) to the case where each column is restricted independently. This is called the ‘beta-switching’ algorithm and described in Section 2.3.

Boswijk & Doornik (2004, section 4.4) present the \( \alpha\beta \)-switching algorithm for linear restrictions on \( \alpha \) and \( \beta \) when the rank of \( \alpha\beta' \) is \( r \). Homogenous linear restrictions between elements of \( \alpha \) can be expressed as \( \text{vec} \alpha = G\phi \), and similarly for \( \beta \) as \( \text{vec} \beta = H\phi \), but our implementation is limited to separate restrictions within columns:

\[
\mathcal{H} : \alpha = (G_1\theta_1, \ldots, G_r\theta_r), \beta = (H_1\phi_1, \ldots, H_r\phi_r),
\]

where the \( H_i \) are known \( p_1 \times m_i \) matrices and \( \phi_i \) vectors of length \( m_i \). Similarly, the \( G_i \) are known \( p \times s_i \) matrices and \( \theta_i \) vectors of length \( s_i \); when \( G_i = I_p \), the corresponding vector in \( \alpha \) is unrestricted. The \( H_i \) need not identify the cointegrating space, but the column rank of \( \alpha \) and \( \beta \) must remain \( r \). Now, \( H = \text{dg}(H_1, \ldots, H_r) \) is block-diagonal and of dimension \( rp_1 \times \sum m_i \) and \( \phi' = (\phi_1', \ldots, \phi_r') \), with \( G, \theta \) defined analogously.

For numerical reasons, we keep the two steps in a generalized least squares (GLS) form that can be estimated by least squares (using a precomputed QR decomposition results in more efficient computations, see the appendix of Doornik, 2017). First, with \( \theta \) fixed at \( \theta^F \), so \( \alpha^F = \alpha(\theta^F) = (G_1\theta_1^F, \ldots, G_r\theta_r^F) \), and \( \Omega^F \), writing \( \text{vec} \beta = H\phi \):

\[
z_{0t} = \text{vec} \left( z_{1t}' \beta \alpha^F \right) + \epsilon_t = \left( \alpha^F \otimes z_{1t'} \right) H\phi + \epsilon_t.
\]

This can be estimated by GLS using \( \Omega^F = PP' \).

© 2018 The Authors. Scandinavian Journal of Statistics published by John Wiley & Sons Ltd on behalf of The Board of the Foundation of the Scandinavian Journal of Statistics
Next, if $\alpha$ is unrestricted, it can be estimated by ordinary least squares (OLS) given $\beta^F = \beta(\phi^F)$. Otherwise, given $\beta^F$ and $\Omega^F$,

$$z_{0t} = \text{vec} \left( \alpha \beta^F z_{1t} \right) + \epsilon_t = \left( I \otimes \beta' z_{1t} \right) G \theta + \epsilon_t,$$

(9)

which can again be estimated by GLS.

We now present the algorithm more formally.

**Algorithm 1: $\alpha\beta$-switching algorithm** To start set $k = 0$ and choose $\phi^{(-1)}, \theta^{(-1)}, \epsilon_1$, and the maximum number of iterations. Compute $\Omega^{(-1)} = \Omega(\beta^{(-1)}, \phi^{(-1)})$ and $f^{(-1)}$.

1. Use (8) given $\Omega^{(k-1)}$ and $\theta^{(k-1)}$ to obtain $\phi^{(k)} = \arg\max_{\phi} f(\theta^{(k-1)}, \phi)$ and so the updated $\theta^{(k)} = (H_1 \phi^{(k)}_1, \ldots, H_r \phi^{(k)}_r)$.

2. Compute the average log-likelihood at the candidate parameters:

$$f^{(k)}(\theta) = -\log |\Omega(\theta^{(k)}, \phi^{(k)})|.$$

3. Accept the candidates as the new actual values: $\theta^{(k)} = \theta^{(k)}_c, \phi^{(k)} = \phi^{(k)}_c, f^{(k)} = f^{(k)}_c$.

T. Terminate with $\theta^{(k)}, \phi^{(k)}$ if

$$|\epsilon^{(k)}| = \left| \frac{f^{(k)}_c - f^{(k-1)}_c}{1 + f^{(k-1)}_c} \right| \leq \epsilon_1$$

and

$$p^{(k)} = \max_{i,j} \left| \frac{\Pi^{(k)}_{ij} - \Pi^{(k-1)}_{ij}}{1 + \Pi^{(k-1)}_{ij}} \right| \leq \epsilon_1^{1/2},$$

(10)

using elements $i,j$ of the matrix $\Pi^{(k)} = \alpha^{(k)} \beta^{(k)}$ and $c^{(0)} = 1$.

Else set increment $k$ and return to step 1.

Step 1 of Algorithm 1 updates the parameters, providing new candidate values. Step 2 evaluates the function at the updated parameters. Step 3, which just accepts the candidates as the new values, will be improved by a line search below, resulting in a significant acceleration. Finally, step T is the convergence decision, based on the relative change in the function value as well as the parameters. The latter, $p^{(k)}$, is based on the long-run coefficients $\Pi$, which are always identified. This is a stronger criterion than the change in the log-likelihood, and the square root of $\epsilon_1$ is used. Basing the convergence decision on the change in the objective function only is more likely to result in premature convergence.

2.3. Beta switching

The $\beta$-switching algorithm of Johansen (1995a) fixes all columns of $\beta$ except one, which is estimated by RRR. One iteration then cycles over all columns freeing one in turn, and this process is repeated until convergence.

Beta switching can handle restrictions of the form:

$$\mathcal{H} : \alpha = C \tilde{\theta}, \beta = (H_1 \phi_1, \ldots, H_r \phi_r),$$

(11)

for a known matrix $C(s \times s)$ and coefficients $\tilde{\theta}(s \times r)$, $p > s \geq r$ and $H_l$ as in (7). The restriction on $\alpha$ can be removed (given $\Omega$) through multiplication of (4) by $\overline{C} = \Omega^{-1} C (C' \Omega^{-1} C)^{-1}$:

$$\overline{C} z_{0t} = \tilde{\theta} \beta' z_{1t} + \overline{C} \epsilon_t, \quad \epsilon_t \sim \text{HN}_p[0_p, \overline{C} \Omega \overline{C}].$$

(12)
The full system is transformed by $(\bar{C} : C_i^\perp)$, which splits it in two independent systems: (12) and $C_i^\perp z_{0t} = C_i^\perp \epsilon_t$. If $\Omega$ is known, we can work with (12) instead of the full system. During iteration, we use the $\Omega$ that corresponds to the most recent estimates of $\alpha$ and $\beta$. It is also possible to remove $C$ exactly (see, e.g. Juselius, 2006, section 11.1). However, the current approach resulted in faster convergence in our experiments, in addition to being more convenient in our implementation.

For the main part of the algorithm, assume that $\alpha$ is unrestricted and partition $\beta = (H_i \phi_i \beta_{\setminus i})$ keeping $\beta_{\setminus i}$, which has $r - 1$ columns, fixed. Then,

$$z_{0t} = \alpha_i \phi_i' H_i' z_{1t} + \alpha_{\setminus i} \beta_{\setminus i}' z_{1t} + \epsilon_t,$$

(13)

and $\hat{\phi}_i$ can be obtained from RRR$(z_{0t}, H_i' z_{1t} | \beta_{\setminus i}' z_{1t})$.

**Algorithm 2: $\beta$-switching algorithm** To start set $k = 0$ and choose $\phi^{(-1)}$, $\epsilon_1$, and the maximum number of iterations.

1. Solve (13) for $i = 1, \ldots, r$ to give $\phi^{(k)}_c$ and so $\beta^{(k)}_c$. Obtain $\alpha^{(k)}_c$ by regressing $z_{0t}$ on $\beta^{(k)}_c' z_{1t}$.

2. Same as Algorithm 1.

Normalization has been omitted from this implementation. Very occasionally, it can help to include an intermittent check on the scale to prevent some coefficients running away, while their product stays unchanged.

### 3. Accelerated estimation

Newton and quasi-Newton-type maximization algorithms that iterate over all parameters use a line search to protect against a downwards step that oversteps the maximum in the current upward direction. The line search can also ensure that there is sufficient progress to prove convergence to a stationary point (see, e.g. Nocedal & Wright, 2006, ch. 3). In practice, the added line search results in much better performance, but there is no need to spend too much effort in trying to achieve high accuracy in these intermediate stages.

An alternating variables maximization algorithm uses no derivative information, although the steps can generally be shown to be in a non-downward direction. In practice, progress is often slow and occasionally so slow that it results in premature convergence. Lack of orthogonality of the sets of parameters over which the algorithm alternates exacerbates this problem. By splitting the parameter space in different directions, the actual step has a tendency to be overly conservative. These algorithms can be accelerated by a line search that allows for an expansion (in contrast to the Newton-type line searches that are usually contracting).

To formalize the line search algorithm, we first write $\pi$ for the parameter vector $(\theta' : \phi')'$. At the start of iteration $k$, when only the iteration counter has been incremented, we possess current parameter values $\pi^{(k-1)}$. One update of the switching algorithm gives candidate values $\pi^{(k)}_c$. Next, a line search gives new values $\pi^{(k)}$, where a step length of unity implies that $\pi^{(k)} = \pi^{(k)}_c$. The standard line search uses $\Delta = \pi^{(k)}_c - \pi^{(k-1)}_c$ together with scalar $\mu$ in

$$\pi^{(k)}(\mu) = \pi^{(k-1)} + \mu \Delta.$$

The proposal here is to define instead

$$\nabla \pi = \pi^{(k)}_c - \pi^{(k-1)}_c,$$

(14)
Fig. 1. Illustration of line search algorithms from actual values $\pi^{(k)}$ (left) and candidate values $\pi^{(k)}_c$ (right). [Colour figure can be viewed at wileyonlinelibrary.com]

and run the line search over $\lambda$ using

$$\pi^{(k)}(\lambda) = \pi^{(k-1)}_c + \lambda \nabla \pi.$$  \hspace{1cm} (15)

This differs from the standard approach (14) because it is based on the previous candidate values. The parameter $\lambda$ is the objective of a scalar maximization:

$$\max_{\lambda} f^{(k)}(\lambda) = \max_{\lambda} \left\{ -\log |\Omega(\pi^{(k)}(\lambda))| \right\}. \hspace{1cm} (16)$$

Denoting the approximate solution as $\lambda^*$, the new parameter values are $\pi^{(k)} = \pi^{(k-1)}_c + \lambda^* \nabla \pi$.

The principle of the algorithm is visualized in Fig. 1, where triangles denote candidate values and circles actual values. In the left graph, the change is expressed relative to the previous actual value. On the right, it is relative to the previous candidate value, which is the proposed method (15). The start in Fig. 1 (right) is at $\pi^{(2)}$, the alternating steps move first along the horizontal axis, then the vertical axis to the next candidate point $\pi^{(3)}_c$, indicated by the solid triangle in the centre. This takes us to $\pi^{(3)}$ from which the process repeats.

For the $\alpha\beta$-switching algorithm, the new candidate values $\theta^{(k)}_c$ and $\phi^{(k)}_c$ are defined in Step 1 of Algorithm 1. The acceleration scheme is

$$\theta^{(k)}(\lambda) = \theta^{(k-1)}_c + \lambda \nabla \theta, \quad \phi^{(k)}(\lambda) = \phi^{(k-1)}_c + \lambda \nabla \phi.$$ \hspace{1cm} (17)

with $\nabla \theta = \theta^{(k)}_c - \theta^{(k-1)}_c$ and $\nabla \phi = \phi^{(k)}_c - \phi^{(k-1)}_c$. The new parameter values are $\theta^{(k)} = \theta^{(k)}(\lambda^*)$ and $\phi^{(k)} = \phi^{(k)}(\lambda^*)$, both using the same $\lambda^*$. Some care is needed if the cointegrating vectors are normalized in an iteration: if the normalization changes, then $\nabla \phi$ will be meaningless.

An unusual aspect of (15) and (17) is that it is defined in terms of the previous candidate parameter values, rather than the previous actual values. This is based on practical experience, and the benefit may come from the fact that it offers some protection against actual steps becoming prematurely too small. We will compare several schemes to estimate the step length of the line search.
3.1. Simple line search

In alternating variables algorithms, the emphasis should be on expanding searches, and a simple solution is just to try a few values for \( \lambda \). We found \( \lambda = 1.2, 2, 4, 8 \) to be effective, resulting in the following simple stepwise procedure:

**Algorithm 3: Line search L1Step** Using (17) and (16), evaluate \( f^{(k)}(\lambda_i) \), for \( \lambda_1, \ldots, \lambda_4 = 1.2, 2, 4, 8 \), as long as \( f^{(k)}(\lambda_i) > \max\{f^{(k)}(\lambda_0), \ldots, f^{(k)}(\lambda_{i-1})\} \) for \( i = 1, \ldots, 4 \) with \( \lambda_0 = 1 \). The step size is \( \lambda^* = \lambda_{i-1} \in \{1, 1.2, 2, 4, 8\} \).

A parallel implementation could evaluate \( f^{(k)}(\lambda_i) \) simultaneously for all \( \lambda_i \) and then choose the best.

3.2. Concentrated line search

The loadings \( \alpha \) and cointegrating vectors \( \beta \) have different properties in the CVAR, with the latter converging at a faster rate. This suggests a line search that is only over the parameters in the cointegrating vectors, re-evaluating the loadings each time. Now, each function evaluation in the line search requires an additional regression:

**Algorithm 4: Line search L1Beta** Use \( \nabla_{\phi} \) from (17) and define \( \alpha \) as \( \theta(\phi(\lambda)) \) based on (9). Then follow the recipe of line search L1Step.

3.3. Least-squares line search

Because the restrictions are linear, we can write:

\[
\alpha(\lambda) = \alpha_0 + \lambda \nabla_\alpha, \quad \beta(\lambda) = \beta_0 + \lambda \nabla_\beta,
\]

with residuals

\[
e_t(\lambda) = z_{0t} - \alpha(\lambda)\beta(\lambda)'z_{1t} = e_t(0) - \left( \lambda \left[ \alpha_0 \nabla_\beta' + \nabla_a \beta_0' \right] + \lambda^2 \nabla_\alpha \nabla_\beta' \right) z_{1t}.
\]

In matrix form

\[
E(\lambda) = E_0 - \lambda Z_a - \lambda^2 Z_b,
\]

where all matrices are \( T \times p \) and 
\[
Z_a = Z_1[\nabla_\beta \alpha_0' + \beta_0 \nabla_\alpha], \quad Z_b = Z_1 \nabla_\beta \nabla_\alpha', \quad Z_i' = (z_{i1} \ldots z_{iT}).
\]

The least-squares solution involves minimizing the trace \( \text{tr} \Omega^{-1} E(\lambda)' E(\lambda) \). This leads to a cubic equation:

\[
\text{tr} \left[ \Omega^{-1} Z_a' E_0 \right] + \lambda \text{tr} \left[ \Omega^{-1} (2Z_b' E_0 - Z_a' Z_a) \right] - 3\lambda^2 \text{tr} \left[ \Omega^{-1} Z_b Z_a \right] - 2\lambda^3 \text{tr} \left[ \Omega^{-1} Z_b' Z_b \right] = 0,
\]

which has either one or three real solutions. In the latter case, which we very rarely observed, the real value closest to unity is used. The least-squares line search is

**Algorithm 5: Line search LSq** Find \( \lambda^* \) by solving (18) using \( \Omega(\theta^{(k)}_c, \phi^{(k)}_c) \) and the change based on candidate values as in (17). Do not accept \( \lambda^* \) if it leads to a worse function value.
4. Performance of accelerated switching algorithms

To illustrate the impact of the proposed additions on the various switching algorithms, we look at a model for the Danish data based on Juselius (2006, section 4.1.1). The model has five dependent variables: real money, real GDP, GDP deflator, and two interest rates. There are two lags in the VAR, with unrestricted constant and restricted trend for the deterministic terms. The estimation period is 1973(3)–2003(1), so 119 observations and cointegrating rank $r = 3$. Note that the aim is not to find a good model, or to test valid restrictions, but rather to stress the algorithms Ox code to replicate all reported results can be found online.

Table 1 lists the range of restrictions under which we estimate the models. Restrictions on the columns of $\alpha$ are indicated by an uppercase letter, those on $\beta$ in lower case. Restrictions Aa, for example refers to the model where $\alpha$ is unrestricted and $\beta = (I_{6;1:3}\phi_1, I_{6;1:6}\phi_2, I_{6;3:6}\phi_3)$. Matrices that are subsets of the identity matrix are written as, for example $I_{6;1,6}$ to select the first and last column of $I_6$, while $I_{6;2:5}$ keeps columns two to five. Further matrices used in Table 1 are

$$H_b = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}, G = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, H_c = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, H_d = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, H_e = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -500 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

Table 2 shows the number of iterations required by the two switching algorithms using $\epsilon_1 = 10^{-12}$ when estimating the model under a range of restrictions. Note that, in our

| Specification of imposed restrictions, see Table 1 | Aa | Ab | Bb | Cb | Ac | Dc |
|---------------------------------------------------|----|----|----|----|----|----|
| No line search | 77,320 | 3573 | 268 | 18 | 4017 | 6200 |
| L1Beta | 369 | 23 | 19 | 12 | 46 | 69 |
| Alpha–beta–switching algorithm | | | | | | |
| No line search | 123 | 10,548 | 1337 | 480 | 104 | 122 |
| L1Step | 28 | 220 | 74 | 45 | 24 | 34 |
| LLsq | 38 | 231 | 75 | 43 | 25 | 42 |
| L1Beta | 27 | 213 | 206 | 26 | 43 | 35 |

Note: $I_p$ is the $p \times p$ identity matrix; $I_{p;a:b}$ selects all columns from $a$ to $b$ from this, while $I_{p:a,b}$ selects only columns $a$ and $b$. The $G, H_b, \ldots, H_e$ matrices are defined in the text.
implementation, the number of iterations equals the number of parameter updates. In each case, the unrestricted estimates for \( r = 3 \) are used as starting values. A better starting value routine should be used in practice. However, here, we focus on algorithm performance and wish to keep the iteration counts comparable.

When there is no line search, the algorithm is occasionally slow to reach the specified precision \( \varepsilon_1 \). Much faster convergence is achieved with the line searches; one case shows an almost 200-fold reduction in the iteration count. For \( \alpha\beta \) switching, we have also implemented the least-squares line search. However, \( \text{LLsq} \) does not offer any advantage here over \( \text{L1Step} \), while requiring more effort to derive and implement.

\( \text{L1Beta} \) involves a line search over the coefficients in \( \beta \) only, otherwise it behaves as \( \text{L1Step} \). Beta switching estimates the model in terms of \( \beta \) only, so here, we only use \( \text{L1Beta} \). This is not the case for \( \alpha\beta \) switching, but there is no clear advantage of \( \text{L1Beta} \) over \( \text{L1Step} \). For restrictions \( \text{Bb} \), it requires 206 iterations where \( \text{L1Step} \) uses 74.

4.1. A more comprehensive comparison

A Monte Carlo experiment is used to show the impact of the line search in more detail for three choices of restrictions. Data are generated from (2) without further lags:

\[
y_t = (I_p + \alpha \beta_t')y_{t-1} + \alpha \beta_t' t + \epsilon_t, \quad \epsilon_t \sim \text{IIN}_p[0, \Omega].
\]

The data generation process (DGP) parameters \( \alpha, \beta' = (\beta'_y, \beta'_c) \) and \( \Omega \) are the estimates of the I(1) model for the Danish data with 2 lags and rank 3 (but no further restrictions). The generated sample has the same size, and \( y_{-1} \) and \( y_0 \) are taken from the actual data. The estimated models using the generated data are the same, except for the additional restrictions on \( \alpha \) and \( \beta \).

\( M = 1000 \) random samples are drawn in this way. The maximum number of iterations was set to 10,000, \( \varepsilon_1 = 10^{-12} \), and all replications are counted. Table 3 gives the average number of iterations required to achieve convergence as well as the total central processing unit (CPU) time for each experiment (in seconds running on all cores of an Intel i7 at 2.3Ghz).

Comparing the \( \text{L1Step} \) and \( \text{LLsq} \) line searches first, we see that \( \text{L1Step} \) has the better performance: it requires more function evaluations, but this is offset by the reduction in updates. This is surprising, because \( \text{L1Step} \) is ad hoc in comparison, using a limited \{1.2, 2, 4, 8\} line search grid.

| Restrictions Ab | Iters | Logliks | CPU(s) | Iters | Logliks | CPU(s) |
|-----------------|-------|---------|--------|-------|---------|--------|
| No line search  | 4580  | 4581    | 244.0  | 50    | 51      | 5.9    |
| \text{L1Beta}  | 26    | 101     | 2.3    | 11    | 47      | 1.1    |
| Alpha–beta–switching algorithm | | | | | | |
| No line search  | 4385  | 4386    | 199.3  | 198   | 199     | 11.6   |
| \text{LStd}    | 3019  | 12529   | 186.1  | 152   | 639     | 12.4   |
| \text{L1Step}  | 586   | 2344    | 35.8   | 40    | 168     | 2.9    |
| \text{LLsq}    | 684   | 1369    | 47.8   | 50    | 102     | 4.1    |
| \text{L1Beta}  | 135   | 597     | 13.1   | 28    | 125     | 2.8    |

Mean iteration count (Iters), mean log-likelihood evaluation count (Logliks) and total central processing time (CPU) time in seconds to convergence for \( \varepsilon_1 = 10^{-12} \) in 1000 replications.
Fig. 2. Beta-switching algorithm with and without line search: histogram of iteration counts to convergence (top row), difference between log-likelihoods $l_c$ upon convergence for each replication. Restrictions Aa. [Colour figure can be viewed at wileyonlinelibrary.com]

**LStd** is the ‘standard’ line search (14), using the same 4-point grid, but based on the previous actual values:

$$\Delta \theta = \hat{\theta}^{(k)} - \hat{\theta}^{(k-1)}, \quad \Delta \phi = \hat{\phi}^{(k)} - \hat{\phi}^{(k-1)},$$

instead of the previous candidate variables as in **L1Step**, **L1Beta** and **LLsq**. Indeed, between **LStd** and **L1Step**, this is the only difference: **L1Step** is substantially faster.

The best performance is with **L1Beta**: for Aa, it is more than twice as fast, although for Dc there is almost no difference. The reduction in iteration count is more pronounced than the gain in speed, because of the overhead of evaluating the concentrated $\alpha$ parameters.

The quality of convergence is just as important as the speed. More detailed statistics are reported for the restrictions denoted by Aa, starting with Fig. 2 presenting results for $\beta$ switching. The first two histograms show the number of iterations (i.e. parameter updates) needed to obtain convergence (or reach the upper limit of 10,000). The **L1Beta** line search is about 50 times faster, reducing updates by two orders of magnitude.

The bottom graph of Fig. 2 shows the difference in the log-likelihoods for the 1000 replications consecutively. In the experiments $\hat{l} = l_c(\hat{\pi}) = -T/2 \log |\Omega(\hat{\theta}, \hat{\phi})|$ is around 3000, and adding the used line search as an argument, the graph plots $\hat{l}(\text{no line search}) - \hat{l}(\text{L1Beta})$. A negative number indicates that the **L1Beta** version converged to a better value. That this happens regularly with small differences is an indication of premature convergence when using no line search. In practice, we should use a better initial value procedure, which is likely to reduce this effect.

The same statistics are shown for $\alpha \beta$ switching in Fig. 3 for three different line searches and none. The histograms along the top show the reductions in iteration count. The remaining graphs compare the achieved maxima, always in such a way that a negative number corresponds to **L1Step** having found a higher maximum. With $\alpha \beta$ switching, discrepancies occur less frequently.

Finally, we compare the rates of convergence. For each experiment, this is measured as the change from the first log-likelihood (i.e. from the first update, not the initial values), set to zero, to the final value upon termination, set to one. Figure 4 plots the 50%, 90% and 99% quantiles, both without line search and using **L1Beta**. In this case, the line search has little impact in the first few iterations of $\beta$ switching but does result in cutting down the tail. With $\alpha \beta$ switching, the 90% and 99% quantiles are clearly shifted to the left from the early stage onwards.
Acceleration of switching algorithms

Fig. 3. Alpha–beta switching algorithm with and without line search: histogram of iteration counts to convergence (top row), difference between log-likelihoods $\hat{L}$ upon convergence for each replication. Restrictions Aa. [Colour figure can be viewed at wileyonlinelibrary.com]

Fig. 4. Speed of convergence of switching algorithms for restrictions Aa: $\beta$-switching (left) and $\alpha\beta$-switching (right). The dashed line is without line search, the dotted line with L1Beta. Shown are the median, 90% and 99% quantiles of the number of iterations to convergence. [Colour figure can be viewed at wileyonlinelibrary.com]

5. Quadratic line search

Generic procedures that maximize a scalar function, such as Brent (1973, ch. 5) or Powell (1964, section 8), could be used but make the overall maximization significantly slower. It may be possible, as suggested by an anonymous referee, to specialize the quadratic procedure of Powell (1964) for the line search. To be competitive with the simple approach of L1Step and L1Beta, the number of function calls must be kept small. We will provide an implementation and compare it to the approaches so far.

5.1. Implementation

Already available are function values (16) at $\lambda = 0, 1$, to which we add $\lambda = 2$. Then, if the values accelerate upwards, the upper boundary of the specified interval is taken as the predicted value $\lambda_q$. Otherwise, we use the quadratic approximation to predict $\lambda_q$, unless the quadratic function is flat or leads to a minimum. Finally, if the prediction is close to a value that has already been evaluated (i.e. at 0, 1, 2), it is accepted. Otherwise, the function is evaluated at the prediction, and the best value returned.
Table 4. Performance of alpha–beta–switching algorithm for I(1) models under restrictions Ab and Dc with different line search algorithms

| Restrictions Ab | Iters | Logliks | CPU(s) | Restrictions Dc | Iters | Logliks | CPU(s) |
|-----------------|-------|---------|--------|-----------------|-------|---------|--------|
| LBrentStd       | 3440  | 14,407  | 217.5  | LBrent          | 162   | 687     | 14.3   |
| LBrent          | 677   | 3031    | 49.3   | L1Step          | 46    | 193     | 3.6    |
| L1Step          | 586   | 2344    | 35.8   | LQStep          | 40    | 168     | 2.9    |
| LQStep          | 550   | 1507    | 34.2   | L1Beta          | 41    | 110     | 3.2    |
| L1Beta          | 135   | 597     | 13.1   | LQBeta          | 28    | 125     | 2.8    |
| LQBeta          | 288   | 831     | 24.1   |                 | 27    | 73      | 2.5    |

Mean iteration count (Iters), log-likelihood evaluations (Logliks) and total central processing unit (CPU) time in seconds to convergence for $\varepsilon_1 = 10^{-12}$ in 1000 replications.

The resulting algorithm remains relative to the previous candidate point and is formalized as

**Algorithm 6: Line search LQStep**

\[ f_0 = f^{(k)}(0) \text{ and } f_1 = f^{(k)}(1), \text{ bounds } a \leq 0, b \geq 2 \text{ using } a = -1, b = 8 \text{ here. Machine precision } \varepsilon_m. \text{ Set } \varepsilon_1 = 10^{-4}, \varepsilon_x = 0.3, \varepsilon_f = \varepsilon_m|f_0| + |f_1|/2. \]

1. Evaluate \( f_2 = f^{(k)}(2), q = -f_0 + 2f_1 - f_2. \)
   (a) If \( f_1 - f_0 > \varepsilon_f \) and \( f_2 - f_1 > f_1 - f_0 + \varepsilon_f \) set \( \lambda_q = b. \)
   (b) Else if \( q \leq \varepsilon_f \) set \( \lambda_q = b/2 \) if \( f_1 - f_0 > -\varepsilon_f, \lambda_q = a/2 \) otherwise.
   (c) Else set \( \lambda_q = -(3f_0 + 4f_1 - f_2)/(2q), \lambda_q = \max(\lambda_q, a), \lambda_q = \min(\lambda_q, b/2 + 1). \)

2. \( \lambda_x = \arg\max(f_0, f_1, f_2), \text{ so } \lambda_x \in \{0, 1, 2\}; f_x = \max(f_0, f_1, f_2). \)
3. If \( |\lambda_q - \lambda_x| > \varepsilon_x \) and \( f(\lambda_q) > f_x \) then \( f_x = \lambda_q, \lambda_x = \lambda_q. \)
4. Return \( f_x, \lambda_x. \)

Steps 1a, 1b and 1c are the upwards acceleration, flatness or wrong curvature and quadratic approximation reflectively. Step 3 is only effective when the prediction is not close, in which case an additional function evaluation is required to check if we have a better point. So LQStep requires only one or two function evaluations.

### 5.2. Results

Table 4 extends Table 3 with the new procedure. To illustrate our claims, we have also added a line search based on the algorithm of Brent (1973, ch. 5). The first version, LBrentStd, maximizes the line search relative to the previous actual values (14). This is the analogue to LStd (Table 3) but now maximizing the scalar function, rather than evaluating it at a few points. LStd is the faster (and simpler) of the two. LBrent uses Brent’s maximization on the line search based on previous candidate values, the approach advocated in this paper, leading to a fivefold improvement. As expected, LBrent is very similar to LLsq.

There are two versions of the quadratic line search: LQStep, which can be directly compared to the simple grid style of L1Step, and LQBeta, which is the quadratic version of L1Beta. In three out of four cases, the quadratic line search has a reduced number of likelihood calls. This has limited impact here but can help in other settings where they are more costly.

### 6. Applications in other settings

It is somewhat surprising that this simple approach to the line search provides such an acceleration of the algorithm. To illustrate the applicability more generally, we consider several different
settings. The first two cases are models estimated by the EM algorithm, while the remaining are two instances of alternating least squares.

6.1. The general maximization algorithm

All algorithms considered here, including the switching procedures considered previously, fit in the framework of Algorithm 7:

Algorithm 7: Structure of alternating variables maximization algorithms:
Maximize($\pi^{(0)}$, Eval, Update):

1. $\pi_{c} = \pi_{0} = \pi = \pi^{(0)}$
2. $f = \text{Eval}(\pi)$
3. Repeat (iteration loop):
   4. $\pi^{(-1)} = \pi, \pi_{-1} = \pi_{0}, \pi_{0} = \pi_{c}, f^{(-1)} = f$
   5. $\pi_{c} = \pi = \text{Update}(\pi)$
   6. $f_{c} = f = \text{Eval}(\pi)$
   7. $[\pi, f] = \text{Linesearch}$
   8. CheckConvergence
   9. return: $\pi, f$.

Writing $\pi$ for the full parameter vector, the user must specify vectorized initial values $\pi^{(0)}$, a function $\text{Eval}$ that evaluates the average log-likelihood or other objective, and an update function. Other choices will have default values: the convergence tolerance $\epsilon_{1}$ in (10), the maximum number of iterations and the type of line search. The relative change in the parameters in (10) may not be on $\pi$ but on a derived set of parameters. For example, in cointegration, it is based on $\alpha\beta'$, which is always identified.

Just before entering the line search, the previous candidate parameter values are $\pi_{c}$, while $\pi^{(-1)}$ are the previous actual values. The main point of the L1Step line search is that it uses $\pi_{0}$, which equals $\text{Update}(\pi^{(-2)})$. This is different from $\pi^{(-1)}$ if a line search was entered previously.

The iteration count denotes the number of times the loop is executed. Without line search, this equals the number of updates and function evaluations (with one more at the start). With line search, there are more function evaluations. For example, a line search that is always entered but never successful doubles the number of calls to $\text{Eval}$. More generally, the reduction in updates of a line search will outweigh the increase in function evaluations. Some line search procedures look further ahead and involve additional calls to $\text{Update}$.

6.2. EM algorithm

The switching algorithms for restricted cointegration estimation exhibit occasional very slow convergence. They have this in common with EM algorithms (see, e.g. Jamshidian & Jennrich, 1997). Many proposals have been made in the literature to improve the EM algorithm by a line search or step adjustment based on recent iteration history (e.g. Varadhan & Roland, 2008, Berlinet & Roland, 2012). In some cases, the proposed line search comes with sophisticated heuristics.

The two test problems are taken from Varadhan & Roland (2008, sections 7.1 and 7.2), who provide a clear description of the data, log-likelihoods and the E and M steps for the EM algorithm. The $\text{Update}$ function for maximization consists of one application of the E and M steps, while $\text{Eval}$ returns the average log-likelihood.
The first model is a two-component mixture of the Poisson distribution, applied to mortality of women 80 years and older, as reported in *The Times* during 1910–1912. The Poisson model has three parameters \( \pi = (\rho, \mu_1, \mu_2)' \), and the model is repeatedly estimated starting from random initial values \((0.05 + 0.9u_0, 100u_1, 100u_2)\), where \( u_i \) is drawn from the uniform(0, 1) distribution.

Two accelerations of the EM algorithm are included in the comparison. The first, labelled LSqS3g, is the global S3 variant of the SQUAREM line search as preferred by Varadhan & Roland (2008). We found that the reported convergence failures could be avoided by forcing the initial step of the SQUAREM line search to stay inside the parameter space. The second is the parabolic line search of Berlinet & Roland (2012), called LParabolic here.

The second EM test case is for the multivariate \( \mathcal{t} \)-distribution. The estimated model assumes one degree of freedom (Cauchy), but the generated data are \( \text{IIN}(0, 1) \). We only report the variant that Varadhan & Roland (2008) call PX-EM. The dimension is 50 with 100 observations. The parameters are the mean \( \mu \) and the Choleski factor \( P \) of the scale \( \Sigma = PP' \). So there are 1325 parameters to estimate. The data are repeatedly drawn, and the model estimated with initial values based on the sample mean and variance of the generated data.

Both cases use \( \epsilon_1 = 10^{-12} \) and a warm-up of three iterations in which the line search is not entered (this is included when counting function calls).

### 6.3. PARAFAC

Parallel factor analysis, or PARAFAC, refers to a type of models that is used in food applications (Bro, 1998), chemometrics (Sanches & Kowalski, 1990) and psychometrics. PARAFAC has the following tensor structure:

\[
y_{ijk} = \sum_{f=1}^{F} a_{if} b_{jf} c_{kf} + e_{ijk}, \quad i = 1, \ldots, I, j = 1, \ldots, J, k = 1, \ldots, K,
\]

where \( e_{ijk} \) is an error term. The data \( y_{ijk} \) can be represented by a three-dimensional matrix, which can be visualized as a sequence of \( I \times J \) matrices stacked behind each other. If, instead, these \( K \) matrices are lined up next to each other, the three-dimensional matrix has been flattened to a normal matrix of dimension \( I \times JK \). This is written as \( Y^{(I \times JK)} \). This flattening can be performed in different ways, for example \( Y^{(K \times IJ)} \). The PARAFAC model can now be written as a trilinear model:

\[
Y^{(I \times JK)} = A(C \star B)' + E^{(I \times JK)},
\]

with objective function

\[
\max_{A,B,C} \frac{-1}{(IJK)^{1/2}} \| Y^{(I \times JK)} - A(C \star B)' \|_F.
\]

This uses the column-Kronecker, or Khatri–Rao product (see, e.g. Liu & Trenkler, 2008): \( C \star B = (c_1 \otimes b_1, \ldots, c_F \otimes b_F) \), which is a \( KJ \times F \) matrix constructed from the columns \( c_F \) of \( C(K \times F) \) and columns \( b_F \) of \( B(J \times F) \). The norm is the Frobenius norm, that is the square root of the sum of the squared elements. \( A, B \) and \( C \) are unique up to column scale.

The PARAFAC model is commonly estimated by alternating least squares (Bro, 1998, section 4.3), because when fixing \( B \) and \( C \), the model can be estimated by least squares. The update function involves three least squares steps:
Acceleration of switching algorithms

Update($A, B, C$):

fix $B, C$: $A_1 = Y^{(I \times JK)} Z (Z' Z)^+ \text{ where } Z = C \ast B$;
fix $A_1, C$: $B_1 = Y^{(J \times IK)} Z (Z' Z)^+ \text{ where } Z = C \ast A_1$;
fix $A_1, B_1$: $C_1 = Y^{(K \times JJ)} Z (Z' Z)^+ \text{ where } Z = B_1 \ast A_1$;
return: Normalize($A_1, B_1, C_1$).

Superscript + denotes the Moore–Penrose inverse; $Z' Z$ can be computed using a Hadamard product. The normalization is recommended by Uschmajew (2012) and takes the following form:

$$(a_f, b_f, c_f) \rightarrow \left( \frac{s_f a_f}{\|a_f\|}, \frac{s_f b_f}{\|b_f\|}, \frac{s_f c_f}{\|c_f\|} \right), s_f = (\|a_f\| \|b_f\| \|c_f\|)^{1/3}, f = 1, \ldots, F.$$

Rajih et al. (2008) propose using an exact line search, labelled ELS, which can be compared to LLSq, except that is formulated in terms of the previous actual values. The trilinear structure of PARAFAC means that now the roots of a fifth-order polynomial must be found. This may require five function evaluations to choose the best. We have not implemented ELS.

The experiment takes the following form: data are repeatedly generated using (19), with $A = 3 + I_{1F}, B = 2 + I_{1F}, C = 1 + I_{1F}$ and $e_{ijk} \sim 0.1 \text{IN}[0, 1]$. $I_{1F}$ is the $I \times F$ identity matrix, so $A$ has the value 4 on the diagonal and 3 elsewhere. For each replication, starting values are chosen as $a_{ij}^{(0)} = a_{ij} + (U(0, 1) - 0.5)/10$ with $U(0, 1)$ representing a uniform random number. Starting values for $B$ and $C$ are chosen in the same way.

6.4. Matrix approximation

The final test case is taken from the literature on low-rank matrix approximations; our test case is for an unstructured matrix. A line search is not commonly used in the proposed procedures, and we shall show the benefits of using L1Step. The algorithm is as in Zachariah et al. (2012) for

$$\max_{A, B} -n^{-1/2} \|y - X \text{vec}(A B')\|_F,$$

where $y(n \times 1)$ and $X(n \times qs)$ are known. The parameters to estimate are $A(q \times r)$ and $B(s \times r)$. The update function involves two regressions:

Update($A, B$):

fix $A$: regress $y$ on $X(I_s \otimes A)$ giving $B_1$;
fix $B_1$: regress $y$ on $X(B \otimes I_q)$ giving $A_1$;
return: $A_1, B_1$.

To test the algorithms, we generate $X$ as a draw from IIN[0, 1], keeping it fixed between replications. $A$ and $B$ are taken as the first $r$ columns of the appropriate identity matrix. In each replication, $y = X \text{vec}(A B') + \epsilon, \epsilon_i \sim \text{IN}[0, 1]$. Starting values are derived from the singular value decomposition (SVD) of the full rank $C = \hat{A} \hat{B}'$. The product $A B'$ is used for the parameter change in the convergence check.

6.5. Results

Table 5 shows the impact of adding a line search to the maximization algorithms. It reports the average number of calls to the Update function and to the objective function (Eval). CPU is the total CPU time for the experiment, on a single core of an Intel Xeon E5 at 2.9 Ghz.

The results show that L1Step provides better acceleration of the EM algorithm than the other line searches, with the lower number of function calls reflected in the reduced total time of
Table 5. Impact of line search algorithms on estimation performance in various model classes. Average number of calls to update and evaluate; total number of failures and total CPU time (in seconds, single core)

| Model Class                       | Avg calls Update | Avg calls Eval | Failures | CPU(s) |
|-----------------------------------|------------------|----------------|----------|--------|
| Poisson mixture, 5000 random starts |                  |                |          |        |
| No line search                    | 2081             | 2082           | 2        | 172.0  |
| L1Step                            | 52               | 200            | 0        | 9.6    |
| LSStep*                           | 75               | 142            | 0        | 7.7    |
| LgSqS3                            | 252              | 171            | 0        | 15.0   |
| LPParabolic                       | 70               | 258            | 0        | 10.0   |
| LQStep                            | 51               | 135            | 0        | 8.0    |
| Multivariate t-distribution, 1000 data samples |                  |                |          |        |
| No line search                    | 102              | 103            | 0        | 84.8   |
| L1Step                            | 31               | 105            | 0        | 48.2   |
| LSStep*                           | 44               | 72             | 0        | 43.0   |
| LgSqS3                            | 45               | 34             | 0        | 33.5   |
| LPParabolic                       | 41               | 109            | 0        | 55.3   |
| LQStep                            | 32               | 80             | 0        | 41.0   |
| PARAFAC, 100 random starts        |                  |                |          |        |
| No line search                    | 38736            | 38737          | 0        | 973.5  |
| L1Step                            | 245              | 989            | 0        | 12.0   |
| LQStep                            | 282              | 779            | 0        | 11.2   |
| Low-rank approximation, 100 random samples |                  |                |          |        |
| No line search                    | 178              | 179            | 0        | 59.1   |
| L1Step                            | 39               | 149            | 0        | 14.0   |
| LQStep                            | 41               | 108            | 0        | 14.1   |

CPU, central processing unit; PARAFAC, parallel factor analysis.

each experiment. Note that the parabolic line search obtains its speed from using the previous candidate, just as L1Step. The parabolic aspect does not yield any further improvement. There is a small additional advantage from using the quadratic version LQStep.

The parabolic line search uses a double update. We can also do this for L1Step: line 5 in Algorithm 7 becomes $\pi_c = \pi = \text{Update}(\text{Update}(\pi))$. This makes it a bit faster still, trading an increase in updates for a reduction in objective evaluations (see Table 5 under LSStep*).

As noted by Varadhan & Roland (2008), the PX variant of the multivariate t-model is already very good. Adding the line searches now only provides a limited improvement.

The speed-up of the PARAFAC algorithm from L1Step is 80-fold, almost two orders of magnitude. The reduction in the number of updates is larger still. In contrast, Rajih et al. (2008) report that, while ELS reduces the number of iterations, it is not really faster than no line search and slower in some cases. L1Step performs extremely well, with the added benefit that it is not problem specific, and extremely easy to implement, both unlike ELS.

Finally, there is a useful improvement of adding L1Step to the low-rank matrix approximation.

There are two cases with premature convergence in the Poisson mixture model without line search. These failures happen because the algorithm makes so little progress in the first 10 iterations that it reports convergence. In the low-rank case, there are five replications that converge to different values. This is not counted as a failure because they appear to be different local modes: different line searches converge to one of two modes. In all other test cases in Table 5, the different accelerations (or absence thereof) converge to the same maximum within the convergence tolerance.
7. Conclusions

We presented a line search to accelerate switching and alternating variables algorithms. The approach is exceedingly simple, but, nonetheless, provides some useful insights in the requirements for good acceleration. First, partitioning the parameter space results in default steps that are too small, and expanding searches are needed. Second, it is much more effective to use the previous candidate parameter values than the previous actual values. Our results illustrate this in many different settings. Finally, there is a penalty for expending too much effort in the line search. In some models, it is possible to explicitly optimize the step length, but even this results in slower algorithms than the proposed simple approximation of $L1\text{Step}$. Similarly, the parabolic line search proposed for EM algorithms spends a bit too much time in a more sophisticated search.

However, while there is limited return in trying to derive a slightly more optimal curve from updates along the iterative path of the algorithm, it may be possible to exploit statistical knowledge of the model that is estimated. The main focus of this paper is on models for cointegration, because these have lacked acceleration so far. Our results show that $L1\text{Step}$ works very well in the I(1) cointegrated VAR models with linear restrictions. Although not reported here, this also holds for non-linear restrictions, different ranks and lag lengths, as well as I(2) models. In the I(1) model, it helps that the $\alpha$ and $\beta$ coefficients are asymptotically independent. We used the fact that $\beta$ converges at a faster rate than $\alpha$ to implement the simple stepwise line search in terms of $\beta$ only. This did provide a further improvement.

Faster convergence facilitates bootstrapping and Monte Carlo experiments. It is also useful when multimodality is suspected, and the model is re-estimated from many randomized initial values.

Acknowledgements

All calculations were performed by the author in Ox 7.1 and 7.2 (Doornik, 2013). Financial support from the Robertson Foundation (award 9907422) and the Institute for New Economic Thinking (grant 20029822) is gratefully acknowledged.

Supporting information

Ox-Switching Acceleration, including code to replicate the experiments in this article are available online.

References

Berlinet, A. F. & Roland, C. (2012). Acceleration of the EM algorithm: P-EM versus epsilon algorithm. *Comput. Statist. Data Anal.* 56, 4122–4137.

Boswijk, H. P. & Doornik, J. A. (2004). Identifying, estimating & testing restricted cointegrated systems: An overview. *Stat. Neerl.* 58, 440–465.

Brent, R. P. (1973). *Algorithms for minimization without derivatives*, Prentice Hall, Englewood Cliffs, NJ.

Bro, R. (1998). Multi-way analysis in the food industry: Models, algorithms, & applications. Technical report, Universiteit van Amsterdam, Amsterdam.

Doornik, J. A. (1998). Approximations to the asymptotic distribution of cointegration tests. *J. Econ. Surv.* 12, 573–593. Reprinted in M. McAleer & L. Oxley (1999). *Practical issues in cointegration analysis*, Blackwell Publishers, Oxford.

Doornik, J. A. (2013). *Object-oriented matrix programming using Ox*, (7th edn.), Timberlake Consultants Press, London.

Doornik, J. A. (2017). Maximum likelihood estimation of the I(2) model under linear restrictions. *Econometrics* 5, 1–20.

© 2018 The Authors. Scandinavian Journal of Statistics published by John Wiley & Sons Ltd on behalf of The Board of the Foundation of the Scandinavian Journal of Statistics
Doornik, J. A. & O’Brien, R. J. (2002). Numerically stable cointegration analysis. *Comput. Statist. Data Anal.* **41**, 185–193.

Jamshidian, M. & Jennrich, R. (1997). Acceleration of the EM algorithm by using quasi–Newton methods. *J. R. Stat. Soc. B* **59**, 569–587.

Johansen, S. (1995a). Identifying restrictions of linear equations with applications to simultaneous equations & cointegration. *J. Econom.* **69**, 111–132.

Johansen, S. (1995b). *Likelihood-based inference in cointegrated vector autoregressive models*, Oxford University Press, Oxford.

Johansen, S. & Juselius, K. (1990). Maximum likelihood estimation & inference on cointegration – With application to the demand for money. *Oxf. Bull. Econ. Statist.* **52**, 169–210.

Johansen, S. & Juselius, K. (1994). Identification of the long-run & the short-run structure. An application to the ISLM model. *J. Econom.* **63**, 7–36.

Juselius, K. (2006). *The cointegrated VAR model: Methodology & applications*, Oxford University Press, Oxford.

Liu, S. & Trenkler, G. (2008). Hadamard, Khatri-Rao, Kronecker & other matrix products. *Int. J. Inf. Syst. Sci.* **4**, 160–177.

Nocedal, J. & Wright, S. J. (2006). *Numerical optimization*, (2nd edn.), Springer-Verlag, New York.

Powell, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.* **7**, 155–162.

Rajih, M., Comon, P. & Harshman, A. (2008). Enhanced line search: A novel method to accelerate PARAFAC. *SIAM J. Matrix Anal. Appl.* **30**, 1128–1147.

Sanches, E. & Kowalski, B. R. (1990). Tensorial resolution: A direct trilinear decomposition. *J Chemom.* **4**, 29–45.

Uschmajew, A. (2012). Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM J. Matrix Anal. Appl.* **33**, 639–652.

Varadhan, R. & Roland, C. (2008). Simple & globally convergent methods for accelerating the convergence of any EM algorithm. *Scand J. Stat.* **35**, 335–353.

Zachariah, D., Sundin, M., Iansson, M. & Chatterjee, S. (2012). Alternating least-squares for low-rank matrix reconstruction. *IEEE Signal Process. Lett.* **19**, 231–234.

Received March 2016, in final form October 2017

Jurgen A. Doornik, Economics Department and Institute for New Economic Thinking at the Oxford Martin School, University of Oxford, Oxford, UK.

E-mail: jurgen.doornik@nuffield.ox.ac.uk

© 2018 The Authors. Scandinavian Journal of Statistics published by John Wiley & Sons Ltd on behalf of The Board of the Foundation of the Scandinavian Journal of Statistics