Appendix 1: Standard Kalman Filter Smoother

Algorithm Standard Kalman Filter Smoother for estimating the moments required in the E-step of an EM algorithm for a linear dynamical system

0. Define $\boldsymbol{x}_t^\tau = \mathrm{E}(\boldsymbol{x}_t|\boldsymbol{Y}_1^\tau)$,$\mathbf{V}_t^\tau = \mathrm{Var}(\boldsymbol{x}_t|\boldsymbol{Y}_1^\tau)$, $\hat{\boldsymbol{x}}_t \equiv \boldsymbol{x}_t^T$ and $\hat{P}_t \equiv V_t^T + \boldsymbol{x}_t^T \boldsymbol{x}_t^{T^\intercal}$
1. Forward Recursions:
   $\boldsymbol{x}_t^{t-1} = A\boldsymbol{x}_{t-1}^{t-1}$
   $\mathbf{V}_t^{t-1} = A\mathbf{V}_{t-1}^{t-1} + \mathbf{Q}$
   $K_t = \mathbf{V}_t^{t-1} C^\intercal (CV_t^{t-1}C^\intercal + R)^{-1}$
   $\boldsymbol{x}_t^t = \boldsymbol{x}_t^{t-1} + K_t(\boldsymbol{y}_t - C\boldsymbol{x}_t^{t-1})$
   $V_t^t = V_t^{t-1} - K_t C V_t^{t-1}$
   $\boldsymbol{x}_1^0 = \pi_0$, $V_1^0 = \mathbf{V}_0$
2. Backward Recursions:
   $J_{t-1} = V_{t-1}^{t-1} A^\intercal (V_t^{t-1})^{-1}$
   $\boldsymbol{x}_{t-1}^T = \boldsymbol{x}_{t-1}^{t-1} + J_{t-1}(\mathbf{x_t^T} - \mathbf{A}\boldsymbol{x}_{\mathbf{t-1}}^{\mathbf{t-1}})$
   $V_{t-1}^T = V_{t-1}^{t-1} + J_{t-1}(V_t^T - V_t^{t-1})J_{t-1}^\intercal$
   $\hat{P}_{t,t-1} \equiv V_{t,t-1}^T + \boldsymbol{x}_t^T \boldsymbol{x}_t^{T^\intercal}$
   $V_{T,T-1}^T = (I - K_T C)AV_{T-1}^{T-1}$

Appendix 2: Derivation of The EM Algorithm

By the chain rule, the full likelihood is

$$P(\boldsymbol{X}, \boldsymbol{Y}) = P(\boldsymbol{Y}|\boldsymbol{X})P(\boldsymbol{X}) = P(\boldsymbol{x}_0)\prod_{t=1}^T P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\prod_{t=1}^T P(\boldsymbol{y}_t|\boldsymbol{x}_t)$$
$$= \prod_{t=1}^T P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\prod_{t=1}^T P(\boldsymbol{y}_t|\boldsymbol{x}_t)\mathbb{1}_{\pi_0}(\boldsymbol{x}_0)$$

where $\mathbb{1}_{\pi_0}(\boldsymbol{x}_0)$ is the indicator function. Conditional likelihoods are

$$P(\boldsymbol{y}_t|\boldsymbol{x}_t) = (2\pi)^{-\frac{p}{2}}|R|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}[\boldsymbol{y}_t - C\boldsymbol{x}_t]^\intercal R^{-1}[\boldsymbol{y}_t - C\boldsymbol{x}_t]\right\}$$
$$P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = (2\pi)^{-\frac{d}{2}} \exp\left\{-\frac{1}{2}[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}]^\intercal[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}]\right\}$$

Then the log-likelihood, after dropping a constant, is just a sum of quadratic terms:

$$
\begin{aligned}
\log P(\boldsymbol{X},\boldsymbol{Y}) = &-\sum_{t=1}^{T}\big(\frac{1}{2}[\boldsymbol{y}_t - C\boldsymbol{x}_t]^{\intercal} R^{-1}[\boldsymbol{y}_t - C\boldsymbol{x}_t]\big) - \frac{T}{2}\log|R| \\
&-\sum_{t=1}^{T}\big(\frac{1}{2}[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}]^{\intercal}[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}]\big) - \frac{T}{2}\log|\mathbf{I}| \\
&+\log(\mathbb{1}_{\pi_0}(\boldsymbol{x}_0)).
\end{aligned}
$$

Then the optimization problem boils down to

$$
\begin{aligned}
\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\Big\{ &\sum_{t=1}^{T}\big(\frac{1}{2}[\boldsymbol{y}_t - C\boldsymbol{x}_t]^{\intercal} R^{-1}[\boldsymbol{y}_t - C\boldsymbol{x}_t]\big) - \frac{T}{2}\log|R| \\
&+\sum_{t=1}^{T}\big(\frac{1}{2}[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}]^{\intercal}[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}]\big) - \frac{T}{2}\log|\mathbf{I}| \\
&-\log(\mathbb{1}_{\pi_0}(\boldsymbol{x}_0)) + \lambda_1\|A\|_1 + \lambda_2\|C\|_2^2\Big\}
\end{aligned}
\tag{1}
$$

Let the target function in the curly braces be denoted as $\boldsymbol{\Phi}(\theta,\boldsymbol{Y},\boldsymbol{X})$. Then $\boldsymbol{\Phi}$ can be optimized with Mr. Sid, a generalized Expectation-Maximization (EM) algorithm.

E Step

The E step of EM requires computation of the expected log likelihood, $\Gamma = E[\log P(\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{Y}]$. This quantity depends on three expectations: $E[\boldsymbol{x}_t|\boldsymbol{Y}]$, $E[\boldsymbol{x}_t\boldsymbol{x}_t^{\intercal}|\boldsymbol{Y}]$ and $E[\boldsymbol{x}_t\boldsymbol{x}_{t-1}^{\intercal}|\boldsymbol{Y}]$. For simplicity, we denote their finite sample estimators by:

$$
\hat{\boldsymbol{x}}_t \equiv E[\mathbf{x_t}|\boldsymbol{Y}],\ \hat{P}_t \equiv E[\boldsymbol{x}_t\boldsymbol{x}_t^{\intercal}|\boldsymbol{Y}],\ \hat{P}_{t,t-1} \equiv E[\boldsymbol{x}_t\boldsymbol{x}_{t-1}^{\intercal}|\boldsymbol{Y}]. \tag{2}
$$

Expectations (2) are estimated with a Kalman filter/smoother (KFS), which is detailed in the Appendix. Notice that all expectations are taken with respect to the current estimations of parameters.

M Step

Each of the parameters in $\theta = \{A, C, R, \pi_0\}$ is estimated by taking the corresponding partial derivatives of $\boldsymbol{\Phi}(\theta,\boldsymbol{Y},\boldsymbol{x})$, setting them to zero, and then solving the equations.

Let the estimations from the previous step be denoted as $\theta^{\text{old}} = \{A^{\text{old}}, C^{\text{old}}, R^{\text{old}}, \pi_0^{\text{old}}\}$ and the current estimations as $\theta^{\text{new}} = \{A^{\text{new}}, C^{\text{new}}, R^{\text{new}}, \pi_0^{\text{new}}\}$. The estimation for the R matrix has a closed form, as follows:

$$\frac{\partial \mathbf{\Phi}}{\partial R^{-1}} = \frac{T}{2}R - \sum_{t=1}^{T}\left(\frac{1}{2}\boldsymbol{y}_t\boldsymbol{y}_t^{\intercal} - C\hat{\boldsymbol{x}}_t\boldsymbol{y}_t^{\intercal} + \frac{1}{2}C\hat{P}_tC^{\intercal}\right) = 0$$

$$\implies R = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{y}_t\boldsymbol{y}_t^{\intercal} - C\hat{\boldsymbol{x}}_t\boldsymbol{y}_t^{\intercal})$$

$$\implies R^{\text{new}} = \text{diag}\left\{\frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{y}_t\boldsymbol{y}_t^{\intercal} - C^{\text{new}}\hat{\boldsymbol{x}}_t\boldsymbol{y}_t^{\intercal})\right\}$$

In the bottom line, diag extracts only the diagonal of the in-bracket term, as we constrain R to be diagonal in Constraint 4.

The estimation for π_0 has a closed form. The relevant term $\log(\mathbb{1}_{\pi_0}(\hat{\boldsymbol{x}}_0))$ is minimized only when $\pi_0^{\text{new}} = \hat{\boldsymbol{x}}_0$.

The estimation for the C matrix also has a closed form. Terms relevant to C are

$$f_{\lambda_2}(C; \boldsymbol{X}, \boldsymbol{Y}) = \sum_{t=1}^{T}\left(\frac{1}{2}[\boldsymbol{y}_t - C\boldsymbol{x}_t]^{\intercal}R^{-1}[\boldsymbol{y}_t - C\boldsymbol{x}_t]\right) + \lambda_2\|C\|_2. \tag{3}$$

In $f_{\lambda_2}(C; \boldsymbol{X}, \boldsymbol{Y})$, C is a matrix, we vectorized it to ease optimization and notation. Without loss of generality, assume R is the identity matrix in equation (3); otherwise, one can always write equation (3) as

$$\sum_{t=1}^{T}\left(\frac{1}{2}[R^{-\frac{1}{2}}y_t - R^{-\frac{1}{2}}Cx_t]^{\intercal}[R^{-\frac{1}{2}y_t} - R^{-\frac{1}{2}}Cx_t]\right) + \lambda_2\|R^{-\frac{1}{2}}C\|$$

Let $\boldsymbol{Y}' = (y_{11}, \ldots, y_{T1}, y_{12}, \ldots, y_{T2}, \ldots, y_{1p}, \ldots, y_{Tp})^{\intercal}$ be a $Tp \times 1$ vector from rearranging $\boldsymbol{Y}$. In addition, let

$$\boldsymbol{X}' = \begin{pmatrix} \boldsymbol{X}^{\intercal} & & \\ & \ddots & \\ & & \boldsymbol{X}^{\intercal} \end{pmatrix}_{pT\times pd}.$$

Finally, vectorize C^{old} as

$$\mathbf{c}^{\text{old}} = (C_{11}^{\text{old}}, \ldots, C_{1d}^{\text{old}}, C_{21}^{\text{old}}, \ldots, C_{2d}^{\text{old}}, C_{p1}^{\text{old}}, \ldots, C_{pd}^{\text{old}})^{\intercal} \tag{4}$$

where C_{ij} is the element at row i and column j of C. With these new notations, the equation (3) is equivalent to

$$f_{\lambda_2}(C; \boldsymbol{X}, \boldsymbol{Y}) = \|\boldsymbol{Y}' - \mathbf{X}'\mathbf{c}\|_2^2 + \lambda_2\|\mathbf{c}\|_2^2. \tag{5}$$

With the Tikhonov regularization, equation (5) has closed form solution

$$\mathbf{c}^{\text{new}} = (\boldsymbol{X}'^{\intercal}\boldsymbol{X}' + \lambda_2\mathbf{I})^{-1}\boldsymbol{X}'^{\intercal}\boldsymbol{Y}'$$
$$C^{\text{new}} = \text{Rearrange } \mathbf{c}^{\text{new}} \text{ by equation (4)}$$

In $f_{\lambda_2}(C; \boldsymbol{X}, \boldsymbol{Y})$, C is a matrix. To simplify notation and optimization, we vectorized it to a vector $\mathbf{c}$ following the methods of Turlach et al. (2005). A closed form solution for $\mathbf{c}$, denoted $\mathbf{c}^{\text{new}}$, is given by the Tikhonov regularization. By rearranging the elements in $\mathbf{c}^{\text{new}}$, one gets an estimation of matrix C. That is,

$$C^{\text{new}} = \text{Rearrange } \mathbf{c}^{\text{new}}$$

Now consider matrix A. Terms involving A in Eq. (1) are

$$f_{\lambda_1}(A; \boldsymbol{X}, \boldsymbol{Y}) = \sum_{t=1}^{T}\big(\frac{1}{2}[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}]^{\intercal}[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}]\big) + \lambda_1\|A\|_1$$

Similar to what we have done to C, $f_{\lambda_1}(A; \boldsymbol{X}, \boldsymbol{Y})$ is equivalent to

$$f_{\lambda_1}(A; \boldsymbol{X}, \boldsymbol{Y}) = \|\mathbf{z} - \mathbf{Z}\mathbf{a}\|_2^2 + \lambda_1\|\mathbf{a}\|_1$$

where $\mathbf{z}$ is a $Td \times 1$ vector obtained by rearranging $\boldsymbol{X}$, and $\mathbf{Z}$ is a block diagonal matrix with diagonal component $Z^{\intercal} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{T-1})^{\intercal}$.

$f_{\lambda_1}(A; \boldsymbol{X}, \boldsymbol{Y})$ does not have a closed form solution due to the ℓ_1 term. However, it can be solved numerically with a Fast Iterative Shrinkage-Thresholding Algorithm (FISTA). The FISTA algorithm is detailed in the Appendix.

With FISTA, matrix A can be updated as follows:

$$A^{\text{new}} = \text{FISTA}(\|\mathbf{Z}^{\intercal}\mathbf{a}^{\text{old}} - \mathbf{z}\|_2^2, \quad \lambda_1)$$

0.1 Initialization

The R matrix is initialized as the identity matrix, while π_0 is initialized as the $\mathbf{0}$ vector. For A and C, denote $\boldsymbol{Y} = [\mathbf{y_1}, \cdots, \mathbf{y_T}]$, a $p \times T$ matrix, then the singular value decomposition (SVD) of $\boldsymbol{Y}$ is $\boldsymbol{Y} = \mathbf{U}\mathbf{D}\mathbf{V}^{\intercal} \approx \mathbf{U}_{p\times d}\mathbf{D}_{d\times d}\mathbf{V}^{\intercal}_{d\times T} = \mathbf{U}_{p\times d}\boldsymbol{X}_{d\times T}$, where $\mathbf{U}_{p\times d}$ is the first d columns of $\mathbf{U}$ and $\mathbf{D}_{d\times d}$ is the upper left block of $\mathbf{D}$. This notation also applies to $\mathbf{V}^{\intercal}_{d\times T}$.

C is then initialized as $\mathbf{U}_{p\times d}$, while the columns of $\boldsymbol{X}_{d\times T}$ are used as input for a vector autoregressive (VAR) model to estimate the initial value for A.

0.2 Improving Computational Efficiency

The major factors that affect the efficiency and scalability of the above EM algorithm involve the storage and computations of the covariance matrix R, which is a $p \times p$ matrix. The following computational techniques are utilized to make the code highly efficient and scalable. For the covariance matrix R, with constraint 4 (i.e. the diagonal assumption), we employ a sparse matrix to represent R, and only the diagonal elements are directly calculated. In the E-step, the term $K_t = V_t^{t-1}C^{\intercal}(CV_t^{t-1}C^{\intercal} + R)^{-1}$ involves the inverse of a large square $p \times p$ matrix, which might be intractable. The Woodbury Matrix Identity is employed to turn a high dimensional matrix inverse to a low dimensional one: $(CV_t^{t-1}C^{\intercal} + R)^{-1} = R^{-1} - R^{-1}C[(V_t^{t-1})^{-1} + C^{\intercal}R^{-1}C]^{-1}C^{\intercal}R^{-1}$.

Note that quantities like R^{-1} and $C^{\intercal}R^{-1}C$ can be pre-computed and reused throughout the E step. With the above three techniques, the EM algorithm can scale to very high dimensions in terms of p, d, and T, without causing any computational issues.

Appendix 3: FISTA Algorithm

In general, FISTA optimize a target function

$$\min_{x\in\mathcal{X}} \quad \mathbf{F}(\mathbf{x};\lambda) = \mathbf{g}(\mathbf{x}) + \lambda\|\mathbf{x}\|_{\mathbf{1}} \tag{6}$$

where $\mathbf{g} : R^n \to R$ is a continuously differentiable convex function and $\lambda > 0$ is the regularization parameter. A FISTA algorithm with constant step is detailed below

Algorithm FISTA($\mathbf{g}, \lambda$).

1. Input an initial guess $\mathbf{x_0}$ and Lipschitz constant $\mathbf{L}$ for $\nabla \mathbf{g}$, set $\mathbf{y_1} = \mathbf{x_0}, t_1 = 1$
2. Choose $\tau \in (0, 1/\mathbf{L}]$; Set $k \leftarrow 0$.
3. **loop**
4. Evaluate $\nabla \mathbf{g}(\mathbf{y_k})$
5. Compute $\mathbf{x_1}= \mathbf{S}_{\tau\lambda}(\mathbf{y_k} - \tau \nabla \mathbf{g}(\mathbf{y_k}))$
6. Compute $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
7. $\mathbf{y_{k+1}} = \mathbf{x_k} + \left(\frac{t_k - 1}{t_{k+1}}\right)\left(\mathbf{x_k} - \mathbf{x_{k-1}}\right)$
8. Set $k \leftarrow k + 1$
9. **end loop**

In the above

$$\mathbf{S}_\lambda(\mathbf{y}) = (|\mathbf{y}| - \lambda)_+ \mathbf{sign}(\mathbf{y}) = \begin{cases} y - \lambda & \text{if } y > \lambda \\ y + \lambda & \text{if } y < -\lambda \\ 0 & \text{if } |y| \leq \lambda. \end{cases}$$

The Lipschitz constant L for $\nabla \mathbf{g}(\mathbf{z}) = \mathbf{Z}^\intercal(\mathbf{Z}\mathbf{a} - \mathbf{z})$, where $\mathbf{g}(\mathbf{z}) = \|\mathbf{Z}^\intercal \mathbf{a} - \mathbf{z}\|_2^2$, is calculated as follows. Denote $\|Z\|$ as the induced norm of matrix Z, then L is

$$L = \sup_{x \neq y} \frac{\|\mathbf{Z}^\intercal(\mathbf{Z}x - \mathbf{Z}y)\|}{\|x - y\|} = \sup_{x \neq 0} \frac{\|\mathbf{Z}^\intercal \mathbf{Z}x\|}{\|x\|} \leq \|\mathbf{Z}^\intercal\|\|\mathbf{Z}\| = \|Z^\intercal\|\|Z\|.$$

Appendix 4: k-step predictions with PCA and Mr. Sid

Algorithm k-step predictions with PCA and Mr. Sid

1. Denote estimations with PCA and Mr. Sid as $A_{pca}, C_{pca}, A_{plds}$, and C_{plds} respectively.
2. PCA estimated latent states at $t = 1000$: $x_{1000,pca}$ = column 1000 of $\boldsymbol{X}_{d\times T}$ from Section 3.3
3. Mr. Sid estimated latent states at $t = 1000$: $x_{1000,pls}$ is from the E step in Section 3.4
4. **for i = 1 to k**
5. $x_{1000+k,pca} = A_{pca}\ x_{999+k,pca}$
6. $y_{1000+k,pca} = C_{pca}\ x_{1000+k,pca}$
7. $x_{1000+k,plds} = A_{plds}\ x_{999+k,plds}$
8. $y_{1000+k,plds} = C_{plds}\ x_{1000+k,plds}$
9. **end**

Appendix 5: Simulation Data Generation

Algorithm Simulation Data Generation

1. Denote the dimensions as p, d and T respectively
2. Generate a $p \times d$ matrix C_0 from a standard Gaussian distribution
3. Sort each column of C_0 in ascending order to get matrix C
4. Generate a $d \times d$ matrix A_0 from a standard Gaussian distribution
5. Add a multiple of the identity matrix to A_0
6. Replace entries in A_0 with small absolute values with 0
7. Scale A_0 to make sure its eigen values are between -1 and 1; use A_0 as the A matrix
8. Let R be a diagonal matrix with positive diagonal entries and Q be the identity matrix
9. Generate simulation data with A, C, Q and R
10. **end**