Optimal Model Averaging of Varying Coefficient Models

Cong Li

School of Economics, Shanghai University of Finance and Economics,

777 Guoding Road, Shanghai, 200433, PR China, li.cong@mail.shufe.edu.cn.

Qi Li

ISEM, Capital University of Economics and Business,

Beijing, 100070, PR China; Department of Economics, Texas A&M University,

College Station, TX 77843, USA, qi-li@tamu.edu.

Jeffrey S. Racine

Department of Economics and Graduate Program in Statistics,

McMaster University, racinej@mcmaster.ca;

Department of Economics and Finance, La Trobe University;

Info-Metrics Institute, American University;

Rimini Center for Economic Analysis; Center for Research

in Econometric Analysis of Time Series (CREATES), Aarhus University.

Daiqiang Zhang

Department of Economics, University at Albany,

SUNY, Albany, NY 12222, USA, dzhang6@albany.edu.

Supplementary Material

Proofs of the main theorems are provided in Supplementary Material 1, while R code can be found in Supplementary Material 2.

S1 Proofs

Proof of Condition (2.14). We provide the following sufficient conditions for Condition (2.14):

(i) X_{ij} *takes values in a compact set* $\mathcal{D}_j \subset [-C_x, C_x]$ *for all* $j = 1, \cdots, \infty$, *where* C_x *is a fixed positive constant.*

(ii) Let $\theta(Z_{i,(s)}) = [E(X_{i,(s)}X'_{i,(s)}|Z_{i,(s)})]^{-1}$. *Then* $\theta(Z_{i,(s)})$ *is a finite and positive definite matrix with* $|\theta_{m,m'}(Z_{i,(s)})| \leq \tilde{C}$ *uniformly in* $s = 1, \cdots, S_n$, *where* $\theta_{m,m'}(Z_{i,(s)})$ *is the* (m, m') *element of* $\theta(Z_{i,(s)})$, $m, m' = 1, \ldots, p_s$, *and* $\tilde{C}$ *is a positive constant that is not related to* s.

(iii) $h_{(s),j} \to 0$ *and* $nH_{(s)} \to \infty$ *as* $n \to \infty$ *for all* $j = 1, \ldots, q_s$, $s = 1, \ldots, S_n$, *where* $H_{(s)} = h_{(s),1} \ldots h_{(s),q_s}$.

(iv) The kernel function $k(\cdot)$ *is a bounded symmetric (around zero) density*

function satisfying $\int k(v)v^4dv$ *is finite.*

Note that the Assumption (i) above implies that $|X_{i,(s),m}| \leq C_x$ and $E(|X_{i,(s),m}|\,|Z_{i,(s)}) \leq C_x$ for all $i,(s),m$. Then,

$$
\begin{aligned}
&\max_i \sum_{j=1}^{n} |P_{(s),ij}| \\
&\quad = \max_i \frac{1}{nH_{(s)}} \sum_{j=1}^{n} \left| X'_{i,(s)} \left[\frac{1}{nH_{(s)}} \sum_{l=1}^{n} X_{l,(s)} X'_{l,(s)} K_{(s)} \left(\frac{Z_{l,(s)} - Z_{i,(s)}}{h_{(s)}} \right) \right]^{-1} X_{j,(s)} K_{(s)} \left(\frac{Z_{j,(s)} - Z_{i,(s)}}{h_{(s)}} \right) \right| \\
&\quad = \max_i \frac{1}{nH_{(s)}} \sum_{j=1}^{n} \left| X'_{i,(s)} \theta(Z_{i,(s)}) X_{j,(s)} \right| K_{(s)} \left(\frac{Z_{j,(s)} - Z_{i,(s)}}{h_{(s)}} \right) f^{-1}(Z_{i,(s)}) + (s.o.) \\
&\quad = \max_i \frac{1}{nH_{(s)}} \sum_{j=1}^{n} \left| \sum_{m=1}^{p_s} \sum_{m'=1}^{p_s} X_{i,(s),m} \theta_{m,m'}(Z_{i,(s)}) X_{j,(s),m'} \right| K_{(s)} \left(\frac{Z_{j,(s)} - Z_{i,(s)}}{h_{(s)}} \right) f^{-1}(Z_{i,(s)}) + (s.o.) \\
&\quad \leq \max_i \sum_{m=1}^{p_s} \sum_{m'=1}^{p_s} \frac{1}{nH_{(s)}} \sum_{j=1}^{n} \left| X_{i,(s),m} \theta_{m,m'}(Z_{i,(s)}) X_{j,(s),m'} \right| K_{(s)} \left(\frac{Z_{j,(s)} - Z_{i,(s)}}{h_{(s)}} \right) f^{-1}(Z_{i,(s)}) + (s.o.) \\
&\quad = \max_i \sum_{m=1}^{p_s} \sum_{m'=1}^{p_s} \left| X_{i,(s),m} \theta_{m,m'}(Z_{i,(s)}) \right| E\left(\left| X_{j,(s),m'} \right| \left| Z_{j,(s)} = Z_{i,(s)} \right. \right) + (s.o.) \\
&\quad = O(p_s^2),
\end{aligned}
$$

where the first inequality is due to the fact that $X_{i,(s),m}\theta_{m,m'}(Z_{i,(s)})X_{i,(s),m'}$ may be positive for some (m,m') and negative for some other (m,m'), and the last equality is due to $|X_{i,(s),m}| = O(1)$, $|\theta_{m,m'}(Z_{i,(s)})| = O(1)$ and $E(|X_{i,(s),m'}|\,|Z_{i,(s)}) = O(1)$ for all s,m,m' implied by assumptions (i) and (ii) above. Hence, we obtain $\max_i \sum_{j=1}^{n} |P_{(s),ij}| = O(\overline{p}^2)$.

□

Proof of Theorem 1. The proof is similar to that of Theorem 1 of Wan, Zhang, and Zou (2010). Let the largest singular values of a matrix A be $\lambda(A)$. By Equation (2.12), we have

$$\lambda(\Omega) = O(1). \tag{S1.1}$$

Under Condition (2.14), by an inequality of Reisz (e.g., Speckman (1988)), we obtain

$$\lambda[P_{(s)}P'_{(s)}] \leq \lambda^2[P_{(s)}] \leq \max_i \sum_{j=1}^{n} |P_{(s),ji}| \max_j \sum_{i=1}^{n} |P_{(s),ji}| = O(\overline{p}^4). \tag{S1.2}$$

Hence,

$$\lambda(P_{(s)}) = \lambda(P(w_s^o)) = O(\overline{p}^2) \text{ for any } s \in \{1, \ldots, S_n\}. \tag{S1.3}$$

Let $A(w) = I - P(w)$. Note that

$$C_n(w) = L_n(w) + n^{-1}\|\epsilon\|^2 + 2n^{-1}\langle \epsilon, A(w)\mu\rangle + 2n^{-1}\left\{\text{trace}[P(w)\Omega] - \langle \epsilon, P(w)\epsilon\rangle\right\}$$

Theorem (1) is valid if the following is true: as $n \to \infty$,

$$\sup_{w \in \mathcal{W}} |\langle \epsilon, A(w)\mu \rangle| / [nR_n(w)] \xrightarrow{p} 0, \tag{S1.4}$$

$$\sup_{w \in \mathcal{W}} |\operatorname{trace}[P(w)\Omega] - \langle \epsilon, P(w)\epsilon \rangle| / [nR_n(w)] \xrightarrow{p} 0, \tag{S1.5}$$

$$\sup_{w \in \mathcal{W}} |L_n(w)/R_n(w) - 1| \xrightarrow{p} 0, \tag{S1.6}$$

First, we consider Equation (S1.4). $\forall \delta > 0$. By the triangle inequality, Chebyshev's inequality, Theorem 2 of Whittle (1960), Equation (S1.1), and

Equation (2.13), we obtain

$$
\begin{aligned}
&Pr\left\{\sup_{w\in\mathcal{W}}|\langle\epsilon,A(w)\mu\rangle|/[nR_n(w)]>\delta\right\}\\
&\quad\le Pr\left\{\sup_{w\in\mathcal{W}}\sum_{s=1}^{S_n}w_s|\epsilon'(I-P_{(s)})\mu|>\delta\xi_n\right\}\\
&\quad\le Pr\left\{\max_{1\le s\le S_n}|\epsilon'(I-P_{(s)})\mu|>\delta\xi_n\right\}\\
&\quad= Pr\left\{\{|\langle\epsilon,A(w_1^o)\mu\rangle|>\delta\xi_n\}\bigcup\{|\langle\epsilon,A(w_2^o)\mu\rangle|>\delta\xi_n\}\bigcup\ldots\bigcup\left\{|\langle\epsilon,A(w_{S_n}^o)\mu\rangle|>\delta\xi_n\right\}\right\}\\
&\quad\le\sum_{s=1}^{S_n}Pr\left\{|\langle\epsilon,A(w_s^o)\mu\rangle|>\delta\xi_n\right\}\text{ by the triangle inequality}\\
&\quad\le\sum_{s=1}^{S_n}E\left\{\frac{\langle\epsilon,A(w_s^o)\mu\rangle^{2N}}{\delta^{2N}\xi_n^{2N}}\right\}\text{ by Chebyshev's inequality}\\
&\quad\le C_1\delta^{-2N}\xi_n^{-2N}\sum_{s=1}^{S_n}\|\Omega^{1/2}A(w_s^o)\mu\|^{2N}\text{ by (7) in Theorem 2 of Whittle (1960)}\\
&\quad\le C_1\delta^{-2N}\xi_n^{-2N}\lambda(\Omega)^N\sum_{s=1}^{S_n}\|A(w_s^o)\mu\|^{2N}\\
&\quad\le C_1\delta^{-2N}\xi_n^{-2N}\lambda(\Omega)^N\sum_{s=1}^{S_n}[nR_n(w_s^o)]^N\to 0,\quad\text{as n}\to\infty\text{ by Equation (S1.1) and Equation (2.13),}
\end{aligned}
$$

where C_1 is a constant, the second to last inequality follows from the result that $\mu'A\mu\le\lambda(A)\mu'\mu$ and $\lambda(AA)=\lambda(A)^2$ for any symmetric square matrix A, and the last inequality follows from the fact that $nR_n(w_s^o)\ge\|A(w_s^o)\mu\|^2$ which is implied by Equation (2.9).

Similarly, for Equation (S1.5) we have

$$
\begin{aligned}
&Pr\left\{\sup_{w\in\mathcal{W}}\left|\text{trace}[P(w)\Omega]-\langle\epsilon,P(w)\epsilon\rangle\right|/[nR_n(w)]>\delta\right\}\\
&\quad=Pr\left\{\sup_{w\in\mathcal{W}}\left|\sum_{s=1}^{S_n}w_s[\text{trace}(P_{(s)}\Omega)-\langle\epsilon,P_{(s)}\epsilon\rangle]\right|/[nR_n(w)]>\delta\right\}\\
&\quad\le Pr\left\{\max_{1\le s\le S_n}\left|\text{trace}(P_{(s)}\Omega)-\langle\epsilon,P_{(s)}\epsilon\rangle\right|/[nR_n(w)]>\delta\right\}\\
&\quad\le\sum_{s=1}^{S_n}Pr\left\{\left|\text{trace}[P(w_s^o)\Omega]-\langle\epsilon,P(w_s^o)\epsilon\rangle\right|>\delta\xi_n\right\}\\
&\quad\le\sum_{s=1}^{S_n}E\left\{\frac{[\text{trace}[P(w_s^o)\Omega]-\langle\epsilon,P(w_s^o)\epsilon\rangle]^{2N}}{\delta^{2N}\xi_n^{2N}}\right\}\\
&\quad\le C_2\delta^{-2N}\xi_n^{-2N}\lambda(\Omega)^N\sum_{s=1}^{S_n}\left\{\text{trace}\left[\Omega P(w_s^o)'P(w_s^o)\right]\right\}^N\ \text{by (8) in Theorem 2 of Whittle (1960)}\\
&\quad\le C_2'\delta^{-2N}\xi_n^{-2N}\lambda(\Omega)^N\sum_{s=1}^{S_n}[nR_n(w_s^o)]^N\to 0,\quad\text{as n}\to\infty,
\end{aligned}
\tag{S1.7}
$$

where C_2 and C_2' are constants, and where the last inequality follows from the fact that $nR_n(w_s^o)\ge\text{trace}\left[\Omega P(w_s^o)'P(w_s^o)\right]$ which is implied by Equation (2.9).

Note that Equation (S1.6) is equivalent to

$$
\sup_{w\in\mathcal{W}}\left|\frac{n^{-1}\|P(w)\epsilon\|^2-n^{-1}\,\text{trace}[\Omega P(w)'P(w)]-2n^{-1}\langle A(w)\mu,P(w)\epsilon\rangle}{R_n(w)}\right|\xrightarrow{p}0.
$$

Thus Equation (S1.6) holds if, as $n \to \infty$, we have

$$\sup_{w \in \mathcal{W}} \left| \frac{\langle A(w)\mu, P(w)\epsilon \rangle}{nR_n(w)} \right| \xrightarrow{p} 0, \tag{S1.8}$$

and

$$\sup_{w \in \mathcal{W}} \left| \frac{\|P(w)\epsilon\|^2 - \text{trace}[\Omega P(w)'P(w)]}{nR_n(w)} \right| \xrightarrow{p} 0. \tag{S1.9}$$

By Equation (S1.3), we have

$$\begin{aligned}
&Pr\left\{ \sup_{w \in \mathcal{W}} \left| \frac{\langle A(w)\mu, P(w)\epsilon \rangle}{nR_n(w)} \right| > \delta \right\} \\
&\leq Pr\left\{ \sup_{w \in \mathcal{W}} \sum_{m=1}^{S_n} \sum_{s=1}^{S_n} w_t w_s \left| \epsilon' P_{(s)}(I - P_{(m)})\mu \right| > \delta \xi_n \right\} \\
&\leq Pr\left\{ \max_{1 \leq m \leq S_n} \max_{1 \leq s \leq S_n} \left| \epsilon' P_{(s)}(I - P_{(m)})\mu \right| > \delta \xi_n \right\} \\
&\leq \sum_{t=1}^{S_n} \sum_{s=1}^{S_n} E\left[\frac{\langle P(w_t^o)\epsilon, A(w_s^o)\mu \rangle^{2N}}{\delta^{2N} \xi_n^{2N}} \right] \\
&\leq C_3 \delta^{-2N} \xi_n^{-2N} \sum_{m=1}^{S_n} \sum_{s=1}^{S_n} \left\| P(w_m^o)\Omega^{1/2} A(w_s^o)\mu) \right\|^{2N} \\
&\leq C_3 \lambda[\Omega^{1/2} P(w_m^o)' P(w_m^o) \Omega^{1/2}]^N \delta^{-2N} \xi_n^{-2N} \sum_{m=1}^{S_n} \sum_{s=1}^{S_n} \|A(w_s^o)\mu)\|^{2N} \\
&\leq C_3 S_n \lambda(\Omega)^N \lambda[P(w_m^o)]^{2N} \delta^{-2N} \xi_n^{-2N} \sum_{s=1}^{S_n} [nR_n(w_s^o)]^N \to 0, \quad \text{as n} \to \infty,
\end{aligned}$$

where C_3 is a constant, and where the last inequality follows from Equation (S1.3). Also,

$$
\begin{aligned}
&Pr\left\{\sup_{w\in\mathcal{W}}\left|\frac{\|P(w)\epsilon\|^2-\operatorname{trace}[\Omega P(w)'P(w)]}{nR_n(w)}\right|>\delta\right\}\\
&\quad\le Pr\left\{\sup_{w\in\mathcal{W}}\sum_{t=1}^{S_n}\sum_{s=1}^{S_n}w_tw_s|\epsilon'P'_{(t)}P_{(s)}\epsilon-\operatorname{trace}[\Omega P'_{(s)}P_{(t)}]|>\delta\xi_n\right\}\\
&\quad\le Pr\left\{\max_{1\le t\le S_n}\max_{1\le s\le S_n}|\epsilon'P'_{(t)}P_{(s)}\epsilon-\operatorname{trace}[\Omega P'_{(s)}P_{(t)}]|>\delta\xi_n\right\}\\
&\quad\le\sum_{t=1}^{S_n}\sum_{s=1}^{S_n}E\left\{\frac{[\langle\Omega^{-1/2}\epsilon,\Omega^{1/2}P(w_t^o)'P(w_s^o)\Omega^{1/2}\Omega^{-1/2}\epsilon\rangle-\operatorname{trace}(\Omega P(w_t^o)'P(w_s^o))]^{2N}}{\delta^{2N}\xi_n^{2N}}\right\}\\
&\quad\le C_4\lambda(\Omega)^N\delta^{-2N}\xi_n^{-2N}\sum_{t=1}^{S_n}\sum_{s=1}^{S_n}\operatorname{trace}(P(w_t^o)'P(w_s^o)\Omega P(w_s^o)'P(w_t^o))^N\\
&\quad\le C_5S_n\lambda(\Omega)^N\lambda[P(w_t^o)]^{2N}\delta^{-2N}\xi_n^{-2N}\sum_{s=1}^{S_n}\operatorname{trace}(\Omega P(w_s^o)'P(w_s^o))^N\\
&\quad\le C_5S_n\lambda(\Omega)^N\lambda[P(w_t^o)]^{2N}\delta^{-2N}\xi_n^{-2N}\sum_{s=1}^{S_n}[nR_n(w_s^o)]^N\to 0,\quad\text{as n}\to\infty,
\end{aligned}
$$

where C_4 and C_5 are constants. Thus we obtain Equation (S1.8) and Equation (S1.9). □

Proof of Theorem 2. Obviously,

$$\widehat{C}_n(w)=C_n(w)+2n^{-1}\operatorname{trace}[P(w)\widehat{\Omega}(w)]-2n^{-1}\operatorname{trace}[P(w)\Omega].$$

Therefore, Equation (2.20) holds if

$$\sup_{w\in\mathcal{W}} |\operatorname{trace}[P(w)\widehat{\Omega}(w)] - \operatorname{trace}[P(w)\Omega]|/[nR_n(w)] = o_p(1). \tag{S1.10}$$

Let $H_{(s)} = \operatorname{diag}(\rho_{11}^{(s)}, \ldots, \rho_{nn}^{(s)})$ and $H(w) = \sum_{s=1}^{S_n} w_s H_{(s)}$. Then we obtain that

$$\begin{aligned}
&\sup_{w\in\mathcal{W}} |\operatorname{trace}[P(w)\widehat{\Omega}(w)] - \operatorname{trace}[P(w)\Omega|/[nR_n(w)] \\
&= \sup_{w\in\mathcal{W}} |[Y - P(w)Y]'H(w)[Y - P(w)Y] - \operatorname{trace}[H(w)\Omega]|/[nR_n(w)] \\
&= \sup_{w\in\mathcal{W}} |[\epsilon + \mu - P(w)Y]'H(w)[\epsilon + \mu - P(w)Y] - \operatorname{trace}[H(w)\Omega]|/[nR_n(w)] \\
&\le \sup_{w\in\mathcal{W}} \frac{|\epsilon'H(w)\epsilon - \operatorname{trace}[H(w)\Omega]|}{[nR_n(w)]} + 2\sup_{w\in\mathcal{W}} \frac{|\epsilon'H(w)[P(w)Y - \mu]|}{[nR_n(w)]} \\
&+ \sup_{w\in\mathcal{W}} \frac{|[P(w)Y - \mu]'H(w)[P(w)Y - \mu]|}{[nR_n(w)]} \\
&\le \sup_{w\in\mathcal{W}} \frac{|\epsilon'H(w)\epsilon - \operatorname{trace}[H(w)\Omega]|}{[nR_n(w)]} \\
&+ 2\sup_{w\in\mathcal{W}} \frac{|\epsilon'H(w)[P(w)\mu - \mu]|}{[nR_n(w)]} \\
&+ 2\sup_{w\in\mathcal{W}} \frac{|\epsilon'H(w)P(w)\epsilon - \operatorname{trace}[H(w)P(w)\Omega]|}{[nR_n(w)]} \\
&+ 2\sup_{w\in\mathcal{W}} \frac{|\operatorname{trace}[H(w)P(w)\Omega]|}{[nR_n(w)]} \\
&+ \sup_{w\in\mathcal{W}} \frac{|[P(w)Y - \mu]'H(w)[P(w)Y - \mu]|}{[nR_n(w)]} \\
&\equiv D_1 + D_2 + D_3 + D_4 + D_5,
\end{aligned}$$

where the definitions of D_i ($i \in 1, \ldots, 5$) should be apparent. By following the proof of Equation (A.5) in Zhao, Zhang, and Gao (2016), we have $D_1 + D_2 + D_3 + D_4 + D_5 = o_p(1)$ by noting that the counterpart of $\mathbf{Q}(\mathbf{w})$ in Zhao, Zhang, and Gao (2016) is $H(w)$, and the definition of $R_n(\mathbf{w})$ in Zhao, Zhang, and Gao (2016) is the same as our $nR_n(w)$. The proof is complete. □

Proof of Corollary 1. Note that

$$\widehat{C}_n^*(w) = n^{-1}\|P(w)Y - Y\|^2 + 2n^{-1}\,\mathrm{trace}[P(w)\widehat{\Omega}_{(s^*)}].$$

Therefore, Equation (2.22) holds if

$$\sup_{w\in\mathcal{W}} |\,\mathrm{trace}[P(w)\widehat{\Omega}_{(s^*)}] - \mathrm{trace}[P(w)\Omega]|/[nR_n(w)] = o_p(1).$$

Let $H_{(s)} = \mathrm{diag}(\rho_{11}^{(s)}, \ldots, \rho_{nn}^{(s)})$ and $H(w) = \sum_{s=1}^{S_n} w_s H_{(s)}$. Then we obtain

that

$$
\begin{aligned}
&\sup_{w\in\mathcal{W}} |\operatorname{trace}[P(w)\widehat{\Omega}_{(s^*)}] - \operatorname{trace}[P(w)\Omega|/[nR_n(w)] \\
&= \sup_{w\in\mathcal{W}} \frac{|(Y-P_{(s^*)}Y)'H(w)(Y-P_{(s^*)}Y) - \operatorname{trace}(P(w)\Omega)|}{[nR_n(w)]} \\
&= \sup_{w\in\mathcal{W}} \frac{|(\epsilon+\mu-P_{(s^*)}\mu-P_{(s^*)}\epsilon)'H(w)(\epsilon+\mu-P_{(s^*)}\mu-P_{(s^*)}\epsilon) - \operatorname{trace}(P(w)\Omega)|}{[nR_n(w)]} \\
&\le \sup_{w\in\mathcal{W}} \frac{|\epsilon'(I_n-P_{(s^*)})'H(w)(I_n-P_{(s^*)})\epsilon - \operatorname{trace}[(I_n-P_{(s^*)})'H(w)(I_n-P_{(s^*)})\Omega]|}{[nR_n(w)]} \\
&+ 2\sup_{w\in\mathcal{W}} \frac{|\epsilon'(I_n-P_{(s^*)})'H(w)(I_n-P_{(s^*)})\mu|}{[nR_n(w)]} \\
&+ \sup_{w\in\mathcal{W}} \frac{|\mu'(I_n-P_{(s^*)})'H(w)(I_n-P_{(s^*)})\mu|}{[nR_n(w)]} \\
&+ \sup_{w\in\mathcal{W}} \frac{\left|\operatorname{trace}\left[P'_{(s^*)}H(w)P_{(s^*)}\Omega\right]\right|}{[nR_n(w)]} \\
&+ 2\sup_{w\in\mathcal{W}} \frac{\left|\operatorname{trace}\left[P'_{(s^*)}H(w)\Omega\right]\right|}{[nR_n(w)]} \\
&\equiv \widetilde{D}_1 + \widetilde{D}_2 + \widetilde{D}_3 + \widetilde{D}_4 + \widetilde{D}_5,
\end{aligned}
$$

where the definitions of $\widetilde{D}_i$ $(i \in 1,\ldots,5)$ should be apparent. By following the proof of $(A.7)$ in Zhang and Wang (2015), we have $\widetilde{D}_1+\widetilde{D}_2+\widetilde{D}_3+\widetilde{D}_4+\widetilde{D}_5 = o_p(1)$ by noting that the counterpart of $\mathbf{Q}(\mathbf{w})$ in Zhang and Wang (2015) is $H(w)$, and the definition of $R_n(\mathbf{w})$ in Zhang and Wang (2015) is the same as our $nR_n(w)$. The proof is complete. □

S2 R Code (NOT FOR PUBLICATION)

The R code to replicate the Monte Carlo simulations is provided below. It can be run in serial mode or in parallel on a laptop, desktop, or cluster environment.

```
## Monte Carlo simulation and functions for model averaging and model
## selection for the varying coefficient specification. This requires
## that the latest version of the R package 'np' from github
## (https://github.com/JeffreyRacine/R-Package-np) along with the
## quadprog, foreach, and doParallel packages from CRAN.

rm(list=ls())

## To process Monte Carlo simulations in parallel (e.g. using multiple
## cores), modify the integer in makeCluster() to the desired number
## of cores, then uncomment/comment the appropriate foreach() command
## at or around lines 133-136. Can also use makeCluster(detectCores())
## to automate the number of cores used.

library(foreach)
library(doParallel)
cl<-makeCluster(detectCores())
registerDoParallel(cl)

library(np)
library(quadprog)
options(np.messages=FALSE)
clusterExport(cl,"options")

## alpha determines the rate of coefficient decline, larger alpha
## implies coefficients decline more quickly with j

alpha <- scan("alpha.dat")

## sigma determines R^2=1/(1+sigma^2)
## R^2 = (0.941, 0.80, 0.50, 0.20) when (c = 0.25, 0.50, 1.0, 2.0)

sigma <- scan("sigma.dat")

## Generate X matrix with large number of N(0,1) columns for the DGP
## (infinite order regression with decaying weights)

num.cols <- 1000

## Sample size, number of Monte Carlo replications, and ability to
## restart if halted

n <- scan("num_obs.dat")
Monte <- scan("num_monte.dat")
system("if test -e mse.out;then wc -l mse.out | awk '{print $1}' > num_monte_exist.dat;else echo 0 > num_monte_exist.dat;fi")
M.exist <- scan("num_monte_exist.dat")

## Number of candidate models is a function of the sample size (Hansen
## (2007))

M <- round(3*n**(1/3))

## Create headers (can be restarted and, when so, the headers are not
## written nor is the seed set)

if(M.exist==0) {
    M.remain <- Monte
    system("rm *.out")
    write(c("JMA","MMA","SAIC","SBIC","AIC","BIC","CV","Cp"),file="mse.out",ncol=8)
    write(c("DGP","JMA","MMA","SAIC","SBIC","AIC","BIC","CV","Cp"),file="r_squared.out",ncol=9)
} else {
    Monte <- Monte+1-M.exist
}

## Functions for computing R^2 and MSE

r.sq <- function(yhat,ybar) {
    sum((yhat-ybar)^2)/sum((y-ybar)^2)
}

mse <- function(yhat,dgp) {
    mean((yhat-dgp)^2)
}
```

```
## Function for computing the hat matrix of the varying coefficient
## specification (the diagonal matrix with 1e-10 proceeds with the
## inversion without throwing an error if the singular case is
## encountered)

hat.mat.npscoef <- function(x,z,bw) {
    W <- cbind(1,x)
    IM <- diag(1e-10,ncol(W),ncol(W))
    K <- npksum(txdat=z,bws=bw,return.kernel.weights=TRUE)$kw
    P <- sapply(1:NROW(x),function(i){tWK <- t(W*K[,i]);W[i,,drop=FALSE]%*%chol2inv(chol(tWK%*%W+IM))%*%tWK})
}

## Function to compute the trace of the hat matrix times \hat\Omega

trace.hat.mat.Omega.npscoef <- function(x,z,bw,resid) {
    sum(diag(hat.mat.npscoef(x,z,bw)*resid^2))
}

## Function for Cp model selection criterion

Cp.npscoef <- function(n,trace.hat.mat,resid) {
    sum(resid^2)*(1 + 2*trace.hat.mat/(n-trace.hat.mat))
}

## Function for BIC model selection

BIC.npscoef <- function(n,trace.hat.mat,sigmasq.ml) {
    log(sigmasq.ml)+trace.hat.mat*log(n)/n
}

## Function for AIC model selection

AIC.npscoef <- function(n,trace.hat.mat,sigmasq.ml) {
    log(sigmasq.ml)+2*trace.hat.mat/n
}

## Storage matrices and vectors (loo = 'leave-one-out')

yhat.loo.mat <- matrix(nrow=n,ncol=M)
yhat.mat <- matrix(nrow=n,ncol=M)
residual.mat <- matrix(nrow=n,ncol=M)

trace.Omega.vec <- numeric(M)
mse.vec <- numeric(M)
aic.vec <- numeric(M)
bic.vec <- numeric(M)
cv.vec <- numeric(M)
cp.vec <- numeric(M)
bw.vec <- numeric(M)
r.sq.vec <- numeric(M)

## Monte Carlo begins

## NB - foreach() is _not_ a loop, and the index i cannot be used in
## return vectors though it is available inside the loop it
## appears... it simply chunks things up... so don't make the mistake
## of expecting it to work like for() or you will be sorely
## disappointed (it is list based, for like 'apply')

## Uncomment for serial processing
##if(Monte>0) foreach(i=1:Monte,.combine='c') %do% {
## Uncomment for parallel processing
if(Monte>0) foreach(i=1:Monte,.packages=c('np','quadprog'),.combine='c',.inorder=FALSE) %dopar% {

    set.seed(i+M.exist)

    z <- runif(n,min=-1,max=1)
    X <- matrix(rnorm(n*num.cols),nrow=n,ncol=num.cols)

    ## Generate parameter vector theta of length num.cols

    theta <- sqrt(2*alpha)*seq(1,num.cols)**(-alpha-1/2)

    ## Generate the DGP, convert to unit variance, add heteroskedastic error with
    ## expected variance sigma^2 (expected r-squared will therefore be
    ## 1/(1+sigma^2))

    dgp <- as.numeric(X%*%theta)*exp(z)
    dgp <- dgp/sd(dgp)

    y <- dgp + rnorm(n,sd=sigma*abs(z)*sqrt(3))

    ybar <- mean(y)
    dgp.r.sq <- r.sq(dgp,ybar)

    for(j in M:1) {

        ## Compute cross-validated bandwidths, the model fit, residuals,
        ## and delete-one fits and residuals

        bws <- npscoefbw(ydat=y,zdat=z,xdat=X[,1:j])
```

```
    model <- npscoef(bws=bws,residuals=TRUE)

    ## Need residuals from the 'largest' model for computing
    ## \hat\Omega (the full matrix X)

    if(j==M) model.largest <- model
    model.loo <- npscoef(bws=bws,delete.one=TRUE,residuals=TRUE)

    yhat <- fitted(model)
    r.sq.vec[j] <- r.sq(yhat,ybar)
    mse.vec[j] <- mse(fitted(model),dgp)
    bw.vec[j] <- bws$bw[1]

    ## For JMA

    yhat.loo.mat[,j] <- fitted(model.loo)
    yhat.mat[,j] <- fitted(model)

    ## For MMA

    residual.mat[,j] <- residuals(model)
    trace.Omega.vec[j] <- trace.hat.mat.Omega.npscoef(X[,1:j],z,model$bws$bw,residuals(model.largest))

    ## For model selection

    trace.hat.mat <- sum(diag(hat.mat.npscoef(X[,1:j],z,model$bws$bw)))

    aic.vec[j] <- AIC.npscoef(n,trace.hat.mat,model$MSE)
    bic.vec[j] <- BIC.npscoef(n,trace.hat.mat,model$MSE)
    cv.vec[j] <-  mean(residuals(model.loo)^2)
    cp.vec[j] <- Cp.npscoef(n,trace.hat.mat,residuals(model))

}

## Grab the mse and r-squared from the AIC, BIC, CV and Cp optimal
## models

aic.mse <- mse.vec[which.min(aic.vec)]
bic.mse <- mse.vec[which.min(bic.vec)]
cv.mse <- mse.vec[which.min(cv.vec)]
cp.mse <- mse.vec[which.min(cp.vec)]

aic.r.sq <- r.sq.vec[which.min(aic.vec)]
bic.r.sq <- r.sq.vec[which.min(bic.vec)]
cv.r.sq <- r.sq.vec[which.min(cv.vec)]
cp.r.sq <- r.sq.vec[which.min(cp.vec)]

## Now compute the JMA-optimal model. Compute weights, impose
## restriction of summing to one and being non-negative

## The w'Dmat w matrix (M x M)

Dmat <- t(yhat.loo.mat)%*%yhat.loo.mat
if(qr(Dmat)$rank<M) Dmat <- Dmat + diag(1e-10,M,M)

## The -2 dvec'w vector (1 X M)

dvec <- t(y)%*%yhat.loo.mat

## The constraint matrix.  Amat has row one the adding up
## constraint, the following num.model rows the non-negativity,
## finally the following num.model the less than one constraints.

Amat <- t(rbind(rep(1,M),diag(x=1,M,M),diag(x=-1,M,M)))

## The constraint vector

bvec <- c(1,rep(0,M),rep(-1,M))

## meq tells us to treat the first constraint as an equality
## constraint, the rest as inequality ones

## JMA weight vector, fitted model, MSE and r-squared

w.hat.jma <- solve.QP(Dmat,dvec,Amat,bvec=bvec,meq=1)$solution

yhat.jma <- yhat.mat%*%w.hat.jma

jma.mse <- mse(yhat.jma,dgp)
jma.r.sq <- r.sq(yhat.jma,ybar)

## Mallows model average (can reuse constraint matrix/vector)

## The w'Dmat w matrix (M x M)

Dmat <- t(residual.mat)%*%residual.mat
if(qr(Dmat)$rank<M) Dmat <- Dmat + diag(1e-10,M,M)

## The 2 dvec'w vector (1 x M) (opposite sign from JMA dvec)

dvec <- -trace.Omega.vec
```

```
    ## MMA weight vector, fitted model, MSE and r-squared

    w.hat.mma <- solve.QP(Dmat,dvec,Amat,bvec=bvec,meq=1)$solution

    yhat.mma <- yhat.mat%*%w.hat.mma

    mma.mse <- mse(yhat.mma,dgp)
    mma.r.sq <- r.sq(yhat.mma,ybar)

    ## SAIC, SBIC weight vectors, fitted model, MSE and r-squared

    w.hat.aic <- exp(-aic.vec/2)/sum(exp(-aic.vec/2))
    w.hat.bic <- exp(-bic.vec/2)/sum(exp(-bic.vec/2))

    yhat.saic <- yhat.mat%*%w.hat.aic
    yhat.sbic <- yhat.mat%*%w.hat.bic

    saic.mse <- mse(yhat.saic,dgp)
    sbic.mse <- mse(yhat.sbic,dgp)

    saic.r.sq <- r.sq(yhat.saic,ybar)
    sbic.r.sq <- r.sq(yhat.sbic,ybar)

    ## Write results to files as the Monte Carlo progresses (can compute
    ## summaries before experiment is completed).

    write(c(jma.mse,mma.mse,saic.mse,sbic.mse,aic.mse,bic.mse,cv.mse,cp.mse),"mse.out",ncol=8,append=TRUE)
    write(c(dgp.r.sq,jma.r.sq,mma.r.sq,saic.r.sq,sbic.r.sq,aic.r.sq,bic.r.sq,cv.r.sq,cp.r.sq),"r_squared.out",ncol=9,append=TRUE)
    write(mse.vec,"mse_models.out",ncol=M,append=TRUE)
    write(w.hat.jma,"jma_weights.out",ncol=M,append=TRUE)
    write(w.hat.mma,"mma_weights.out",ncol=M,append=TRUE)
    write(w.hat.aic,"saic_weights.out",ncol=M,append=TRUE)
    write(w.hat.bic,"sbic_weights.out",ncol=M,append=TRUE)
    write(bw.vec,"bw.out",ncol=length(bw.vec),append=TRUE)

}

stopCluster(cl)
```

S3 R Code for the Illustrative Application (NOT FOR PUBLICATION)

The R code to replicate the illustrative application is provided below. It can be run in serial mode or in parallel on a laptop, desktop, or cluster environment. When the number of candidate models is large (e.g. 500+) the benefits of running in parallel can be substantial.

```
num.reps <- 1000

set.seed(42)

library(foreach)
library(doParallel)
cl<-makeCluster(detectCores())
registerDoParallel(cl)

library(np)
library(quadprog)
options(np.messages=FALSE)

## Functions for computing R^2 and MSE

r.sq <- function(yhat,ybar) {
    sum((yhat-ybar)^2)/sum((y-ybar)^2)
}

## Function for computing the hat matrix of the varying coefficient
## specification (the diagonal matrix with 1e-10 proceeds with the
## inversion without throwing an error if the singular case is
## encountered)

hat.mat.npscoef <- function(x,z,bw) {
    W <- cbind(1,x)
    IM <- diag(1e-10,ncol(W),ncol(W))
    K <- npksum(txdat=z,bws=bw,return.kernel.weights=TRUE)$kw
    P <- sapply(1:NROW(x),function(i){tWK <- t(W*K[,i]);W[i,,drop=FALSE]%*%chol2inv(chol(tWK%*%W+IM))%*%tWK})
}

## Function to compute the trace of the hat matrix times \hat\Omega

trace.hat.mat.Omega.npscoef <- function(x,z,bw,resid) {
    sum(diag(hat.mat.npscoef(x,z,bw)*resid^2))
}

## Function for Cp model selection criterion

Cp.npscoef <- function(n,trace.hat.mat,resid) {
    sum(resid^2)*(1 + 2*trace.hat.mat/(n-trace.hat.mat))
}

## Function for BIC model selection

BIC.npscoef <- function(n,trace.hat.mat,sigmasq.ml) {
    log(sigmasq.ml)+trace.hat.mat*log(n)/n
}

## Function for AIC model selection

AIC.npscoef <- function(n,trace.hat.mat,sigmasq.ml) {
    log(sigmasq.ml)+2*trace.hat.mat/n
}

## Data

data(wage1)
attach(wage1)
y <- lwage
z <- data.frame(factor(female),factor(married))
## X must be a matrix...
```

```
d <- scan("poly_order.dat")
X <- cbind(poly(educ,d),poly(exper,d),poly(tenure,d))
K <- ncol(X)
M <- 0
for(k in 1:K) M <- M + ncol(combn(K,k))
indices.mat <- matrix(0,K,M)
j <- 1
for(k in 1:K) {
  indices.mat[1:nrow(combn(K,k)),j:(j+ncol(combn(K,k))-1)] <- combn(K,k)
  j <- j + ncol(combn(K,k))
}

n <- nrow(X)

## foreach() returns a list (or list of lists), it is not a
## replacement for for(), rather a kindof 'apply' replacement albeit
## in parallel (thanks Revolution R programmers!)

n.train <- 500
n.eval <- n - n.train

## Write results to files
write(c("JMA","MMA","SAIC","SBIC","AIC","BIC","CV","Cp"),
      file="r_squared.out",ncol=8)
write(c("AIC","BIC","CV","Cp"),
      file="model_selection.out",ncol=4)
write(c("JMA","MMA","SAIC","SBIC","AIC","BIC","CV","Cp"),
      file="mse.out",ncol=8)

for(m in 1:num.reps) {

    ii <- sample(n)
    ii.train <- ii[1:n.train]
    ii.eval <- ii[(1+n.train):n]

    y.train <- y[ii.train]
    ybar.train <- mean(y.train)
    y.eval <- y[ii.eval]

    z.train <- z[ii.train,]
    z.eval <- z[ii.eval,]

    ## Overhead for computing the largest model (done in serial, might be
    ## folded into post processing but might have to pass back big hat
    ## matrices)

    bws <- npscoefbw(ydat=y.train,zdat=z.train,xdat=X[ii.train,indices.mat[,M]])
    model.largest <- npscoef(bws=bws,residuals=TRUE)

    output <- foreach(i=1:M,.packages=c('np','quadprog')) %dopar% {

        X.train <- X[ii.train,indices.mat[,i]]
        X.eval <- X[ii.eval,indices.mat[,i]]

        bws <- npscoefbw(ydat=y.train,zdat=z.train,xdat=X.train)
        model <- npscoef(bws=bws,residuals=TRUE)
        model.loo <- npscoef(bws=bws,delete.one=TRUE,residuals=TRUE)
        trace.hat.mat <- sum(diag(hat.mat.npscoef(X.train,z.train,model$bws$bw)))

        list(r.sq.train.vec=r.sq(fitted(model),ybar.train),
             yhat.train.loo.mat=fitted(model.loo),
             yhat.train.mat=fitted(model),
             residual.train.mat=residuals(model),
             trace.Omega.train.vec=trace.hat.mat.Omega.npscoef(X.train,z.train,model$bws$bw,residuals(model.largest)),
             aic.train.vec=AIC.npscoef(n.train,trace.hat.mat,model$MSE),
             bic.train.vec=BIC.npscoef(n.train,trace.hat.mat,model$MSE),
             cv.train.vec=mean(residuals(model.loo)^2),
             cp.train.vec=Cp.npscoef(n.train,trace.hat.mat,residuals(model)),
             yhat.eval.mat=fitted(npscoef(ydat=y.train,zdat=z.train,xdat=X.train,ezdat=z.eval,exdat=X.eval,bws=bws$bw)),
             residual.eval.mat=y.eval-fitted(npscoef(ydat=y.train,zdat=z.train,xdat=X.train,ezdat=z.eval,exdat=X.eval,bws=bws$bw)))

    }

    ## Storage matrices and vectors (loo = 'leave-one-out')

    yhat.train.loo.mat <- matrix(nrow=n.train,ncol=M)
    yhat.train.mat <- matrix(nrow=n.train,ncol=M)
    residual.train.mat <- matrix(nrow=n.train,ncol=M)
    residual.eval.mat <- matrix(nrow=n.eval,ncol=M)
    yhat.eval.mat <- matrix(nrow=n.eval,ncol=M)

    trace.Omega.train.vec <- numeric(M)
    aic.train.vec <- numeric(M)
    bic.train.vec <- numeric(M)
    cv.train.vec <- numeric(M)
    cp.train.vec <- numeric(M)
    r.sq.train.vec <- numeric(M)

    ## Extract elements of list into storage matrices and vectors

    for(i in 1:M) {
        yhat.train.loo.mat[,i] <- output[[i]]$yhat.train.loo.mat
```

```
    yhat.train.mat[,i] <- output[[i]]$yhat.train.mat
    residual.train.mat[,i] <- output[[i]]$residual.train.mat
    residual.eval.mat[,i] <- output[[i]]$residual.eval.mat
    yhat.eval.mat[,i] <- output[[i]]$yhat.eval.mat

    trace.Omega.train.vec[i] <- output[[i]]$trace.Omega.train.vec
    aic.train.vec[i] <- output[[i]]$aic.train.vec
    bic.train.vec[i] <- output[[i]]$bic.train.vec
    cv.train.vec[i] <- output[[i]]$cv.train.vec
    cp.train.vec[i] <- output[[i]]$cp.train.vec
    r.sq.train.vec[i] <- output[[i]]$r.sq.train.vec
}

## Grab the r-squared from the AIC, BIC, CV and Cp optimal
## models

aic.r.sq <- r.sq.train.vec[which.min(aic.train.vec)]
bic.r.sq <- r.sq.train.vec[which.min(bic.train.vec)]
cv.r.sq <- r.sq.train.vec[which.min(cv.train.vec)]
cp.r.sq <- r.sq.train.vec[which.min(cp.train.vec)]

aic.mse <- mean((yhat.eval.mat[,which.min(aic.train.vec)]-y.eval)^2)
bic.mse <- mean((yhat.eval.mat[,which.min(bic.train.vec)]-y.eval)^2)
cv.mse <- mean((yhat.eval.mat[,which.min(cv.train.vec)]-y.eval)^2)
cp.mse <- mean((yhat.eval.mat[,which.min(cp.train.vec)]-y.eval)^2)

## Now compute the JMA-optimal model. Compute weights, impose
## restriction of summing to one and being non-negative

## The w'Dmat w matrix (M x M)

Dmat <- t(yhat.train.loo.mat)%*%yhat.train.loo.mat
if(qr(Dmat)$rank<M) Dmat <- Dmat + diag(1e-10,M,M)

## The -2 dvec'w vector (1 X M)

dvec <- t(y.train)%*%yhat.train.loo.mat

## The constraint matrix.  Amat has row one the adding up
## constraint, the following num.model rows the non-negativity,
## finally the following num.model the less than one constraints.

Amat <- t(rbind(rep(1,M),diag(x=1,M,M),diag(x=-1,M,M)))

## The constraint vector

bvec <- c(1,rep(0,M),rep(-1,M))

## meq tells us to treat the first constraint as an equality
## constraint, the rest as inequality ones

## JMA weight vector, fitted model and r-squared

w.hat.jma <- solve.QP(Dmat,dvec,Amat,bvec=bvec,meq=1)$solution

yhat.train.jma <- yhat.train.mat%*%w.hat.jma
yhat.eval.jma <- yhat.eval.mat%*%w.hat.jma

jma.r.sq <- r.sq(yhat.train.jma,ybar.train)
jma.mse <- mean((yhat.eval.jma-y.eval)^2)

## Mallows model average (can reuse constraint matrix/vector)

## The w'Dmat w matrix (M x M)

Dmat <- t(residual.train.mat)%*%residual.train.mat
if(qr(Dmat)$rank<M) Dmat <- Dmat + diag(1e-10,M,M)

## The 2 dvec'w vector (1 x M) (opposite sign from JMA dvec)

dvec <- -trace.Omega.train.vec

## MMA weight vector, fitted model and r-squared

w.hat.mma <- solve.QP(Dmat,dvec,Amat,bvec=bvec,meq=1)$solution

yhat.train.mma <- yhat.train.mat%*%w.hat.mma
yhat.eval.mma <- yhat.eval.mat%*%w.hat.mma

mma.r.sq <- r.sq(yhat.train.mma,ybar.train)
mma.mse <- mean((yhat.eval.mma-y.eval)^2)

## SAIC, SBIC weight vectors, fitted model and r-squared

w.hat.aic <- exp(-aic.train.vec/2)/sum(exp(-aic.train.vec/2))
w.hat.bic <- exp(-bic.train.vec/2)/sum(exp(-bic.train.vec/2))

yhat.train.saic <- yhat.train.mat%*%w.hat.aic
yhat.train.sbic <- yhat.train.mat%*%w.hat.bic

yhat.eval.saic <- yhat.eval.mat%*%w.hat.aic
yhat.eval.sbic <- yhat.eval.mat%*%w.hat.bic
```

```
    saic.r.sq <- r.sq(yhat.train.saic,ybar.train)
    sbic.r.sq <- r.sq(yhat.train.sbic,ybar.train)

    saic.mse <- mean((yhat.eval.saic-y.eval)^2)
    sbic.mse <- mean((yhat.eval.sbic-y.eval)^2)

    write(c(jma.r.sq,mma.r.sq,saic.r.sq,sbic.r.sq,aic.r.sq,bic.r.sq,cv.r.sq,cp.r.sq),
          file="r_squared.out",ncol=8,append=TRUE)
    write(c(which.min(aic.train.vec),which.min(bic.train.vec),which.min(cv.train.vec),which.min(cp.train.vec)),
          file="model_selection.out",ncol=4,append=TRUE)
    write(c(jma.mse,mma.mse,saic.mse,sbic.mse,aic.mse,bic.mse,cv.mse,cp.mse),
          file="mse.out",ncol=8,append=TRUE)

}

stopCluster(cl)
```