Estimating partially linear functional-coefficient panel data models with Stata

Kerui Du
School of Management, Xiamen University
Xiamen, Fujian/China
kerrydu@xmu.edu.cn
Yonghui Zhang
School of Economics, Renmin University of China
Beijing/China
yonghui.zhang@hotmail.com
Qiankun Zhou
Department of Economics, Louisiana State University
Baton Rouge/USA
qzhou@lsu.edu

Abstract. In this article, we describe Stata implementation of estimating partially linear functional-coefficient panel models with fixed effects proposed by An et al. (2016) and Zhang and Zhou (2020). Three new commands xtplfc, ivxtplfc and xtdplfc are introduced and illustrated through Monte Carlo simulations to exemplify the effectiveness of these estimators.

Keywords: xtplfc, ivxtplfc, xtdplfc, functional coefficients, fixed effects, sieve, spline

1 Introduction

A partial linear functional-coefficient regression model allows for linearity in some regressors and nonlinearity in other regressors, where the effects of these covariates on the dependent variable vary according to a set of low-dimensional variables nonparametrically (Cai et al. 2017), thereby showing distinct advantages in capturing nonlinearity and heterogeneity. Since the seminal work of Chen and Tsay (1993), the functional-coefficient models have drawn much attention in the literature. To name a few, Cai et al. (2000b; 2000a; 2009) study functional-coefficient models under the time series framework. Huang et al. (2004) estimate a functional-coefficient panel data model without fixed effects via the series method. Cai and Li (2008) study functional-coefficient dynamic panel data models without fixed effects based on the kernel method. Sun et al. (2009) consider functional-coefficient panel data models with fixed effects which are removed via the least square dummy variable (LSDV) approach. Alternatively, An et al. (2016) deal with the fixed effects via the first time difference and estimate the models using the series method. Zhang and Zhou (2020) propose to use a sieve-2SLS procedure to estimate functional-coefficient panel dynamic models with fixed effects and develop a model specification test for the constancy

of slopes.

In empirical studies, functional-coefficient models have been widely used. For example, they are applied to explore whether working experience matters to the impact of eduction on wage (Su et al. 2013; Cai et al. 2017), to analyze the heterogeneous effect of FDI on economic growth (Delgado et al. 2014; Cai et al. 2017), to examine the nonlinear relationship between income level and democracy (Lundberg et al. 2017; Zhang and Zhou 2020), to compare the returns to scale of the US commercial banks across different regimes (Feng et al. 2017), and to investigate the role of marketization in China's energy rebound effect (Li et al. 2019).

The objective of this article is to present a new Stata module to estimate partially linear functional-coefficient panel data models with fixed effects. The remainder of the paper is organized as follows. Section 2 briefly describes the model and the estimation procedure. Sections 3-5 explain the syntax and options of the new commands. Section 6 provides some Monte Carlo simulations. Section 7 concludes the paper.

2 The model

Consider a partially linear functional-coefficient panel data model of the form

$$Y_{it} = Z_{it}' g(U_{it}) + X_{it}'\beta + a_i + \varepsilon_{it}, i = 1, \ldots, N, t = 1, \ldots, T, \tag{1}$$

where the subscript i and t denote individual i and time t, respectively; Y_{it} is the scalar dependent variable; $U_{it} = (U_{1,it}, \ldots, U_{l,it})'$ is a vector of continuous variables; $Z_{it} = (Z_{1,it}, \ldots, Z_{l,it})'$ is a vector of covariates with functional coefficients $g(U_{it}) = (g_1(U_{1,it}), \ldots, g_l(U_{l,it}))'$; X_{it} is a $k \times 1$ vector of covariates with constant slopes β; a_i is the individual fixed effects which might be correlated with Z_{it}, U_{it} and X_{it}. ε_{it} represents the idiosyncratic error. Moreover, part of the elements in Z_{it} and X_{it} are allowed to be endogenous variables which are correlated with ε_{it}, and they could also include lagged dependent variables as in Zhang and Zhou (2020)[1]. In the latter case, model (1) becomes a dynamic panel model with functional coefficients.

Recently, An et al. (2016) and Zhang and Zhou (2020) propose using the series method to estimate model (1). The estimation procedure is sketched as follows.

First, one can use a linear combination of sieve basis functions to approximate the unknown functional coefficients in (1). Let $h^p(\cdot) = (h_{p,1}(\cdot), \ldots, h_{p,L_p}(\cdot))'$ be a $L_p \times 1$ sequence of basis functions where the number of sieve basis functions $L_p \equiv L_{NL}$ increases as either N or T increases. We have $g_p(\cdot) \approx h^p(\cdot)'\gamma_p$ for $p = 1, \ldots, l$, where $\gamma_p = (\gamma_{p1}, \ldots, \gamma_{pLp})$. Then model (1) can be re-written as

$$Y_{it} = H_{it}'\Gamma + X_{it}'\beta + a_i + v_{it}, \tag{2}$$

[1]In this paper, we restrict that all elements in U_{it} are exogenous. Thus, for the case of dynamic panel data models, the first lagged dependent variable should not enter U_{it}

where $H_{it} \equiv (Z_{1,it}h^1(U_{1,it})', \ldots, Z_{l,it}h^l(U_{l,it})')'$, $\Gamma \equiv (\gamma_1', \ldots, \gamma_l')'$ and $v_{it} = \varepsilon_{it} + r_{it}$,

$$r_{it} = Z_{it}'g(U_{it}) - H_{it}'\Gamma, \tag{3}$$

denoting the sieve approximation error which becomes asymptotic negligible as $L_p \to \infty$ for $p = 1, \ldots, l$ when $(N, T) \to \infty$.

Then, taking the first time difference of model (2) to eliminate the fixed effects yields

$$\Delta Y_{it} = \Delta H_{it}'\Gamma + \Delta X_{it}'\beta + \Delta v_{it}, \tag{4}$$

where Δ represents first difference operator, i.e., $\Delta A_t \equiv A_t - A_{t-1}$.

If all the variables are exogenous, model (4) can be estimated through the least square(LS) method as in An et al. (2016)

$$\left(\hat{\Gamma}', \hat{\beta}'\right)' = \left(\Delta\tilde{X}'\Delta\tilde{X}\right)^{-}\left(\Delta\tilde{X}'\Delta Y\right), \tag{5}$$

where $\tilde{X} = (X, H)$;$X = (X_1', ..., X_N')'$, $X_i = (X_{i1}, \ldots, X_{iT})'$; $H = (H_1, ..., H_N)'$, $H_i = (H_{i1}, \ldots, H_{iT})'$, and $^-$ denotes the generalized inverse.

If part of variables in Z_{it} and X_{it} are endogenous, Eq. (4) can be estimated via the two-step least square(2SLS) method as in Zhang and Zhou (2020). Suppose we have a $d \times 1$ vector of instrumental variables with $d \geq (\sum_{p=1}^{l} L_p + k)$. The 2SLS estimator is given by

$$\left(\hat{\Gamma}', \hat{\beta}'\right)' = \left(\Delta\tilde{X}'P_W\Delta\tilde{X}\right)^{-}\left(\Delta\tilde{X}'P_W\Delta Y\right), \tag{6}$$

where $P_W = W(W'W)^{-}W'$ is a projection matrix with W being the instrumental variables matrix.

Once $\hat{\Gamma}$ is obtained, the functional coefficients $g(U_{it})$ can be estimated

$$\hat{g}_p(U_{p,it}) = h^p(U_{p,it})'\hat{\gamma}_p, p = 1, \ldots, l. \tag{7}$$

Under certain regular assumptions, Zhang and Zhou (2020) establish the consistency and asymptotic normality of the above estimators when sample size N and T go to infinity jointly or only N tends to infinity.

In practice, there are several sieve methods to approximate the unknown functions. We follow Libois et al. (2013) to employ the B-splines. More technical details on the B-splines can be found in Newson (2001).

3 The xtplfc command

`xtplfc` estimates An et al.'s (2016) partially linear functional-coefficient panel data models with exogenous variables.

3.1 Syntax

`xtplfc` *varlist*, `zvars(`*varlist*`)` `uvars(`*varlist*`)` `generate(`*string*`)` [

`power(`***numlist***`)` `nknots(`***numlist***`)` `quantile` `maxnknots(`***numlist***`)`

`minnknots(`*numlist*`) grid(`*string*`) pctile(#) brep(#) wild`
`predict(`*prspec*`) nodots level(#) fast tenfoldcv`]

3.2 Options

`zvars(`*varlist*`)` specifies variables that have functional coefficients. It is required.

`uvars(`*varlist*`)` specifies (continuous) variables that enter the functional coefficients interacted with variables specified by zvars() in order. It is required.

`generate(`*string*`)` specifies a prefix for the names to store fitted values of functional coefficients. It is required.

`te` specifies including time fixed effects.

`power(`*numlist*`)` specifies the power (or degree) of the splines; default is power(3).

`nknots(`*numlist*`)` specifies the number of knots used for the spline interpolation in order specified by uvars(). If absent, 2 is assumed for all the functions.

`quantile` specifies creating knots based on empirical quantiles. By default, the knots are generated by the rule of equal space.

`maxnknots(`*numlist*`)` specifies the maximum number of knots used for conducting least-squares cross-validation(LSCV). If present, LSCV is employed to determine the optimal number of knots. In our practice, we perform the Leave-One-Out cross-validation (CV) across the panelvar. That is to say, we leave one individual (with all observations during the sample period) out each time.

`minnknots(`*numlist*`)` specifies the minimun number of knots used for performing LSCV. If absent, 2 is assumed.

`grid(`*string*`)` specifies a prefix for the names to store the grid points of the variable specified by uvars(*varlist*).

`pctile(#)` specifies the domain of the generating grid points; default is pctile(0).

`brep(#)` specifies the number of bootstrap replications. The default is bootstrap(200).

`wild` specifies using the wild bootstrap. By default, residual bootstrap with cluster(panelvar) is performed.

`predict(`*prspec*`)` stores predicted values of the conditional mean and fixed effects using variable names specified in *prspec*. Specifically, the expression is predict(*varlist*| *stub** [, *replace noai*]). The option takes a variable list or a stub. The first variable name corresponds to the predicted conditional mean. The second name corresponds to fixed effects.When *replace* is used, variables with the names in *varlist* or *stub** are replaced by those in the new computation. If *noai* is specified, only a variable for the mean is created.

`nodots` suppresses iteration dots.

`level(#)` sets confidence level; default is level(95).

`fast` speeds up using mata functions.

`tenfoldcv` specifies using ten-fold CV. It is done by dividing the sample into ten pieces and conduct LSCV across these ten pieces. Specifically, given the number of knots, leave one piece out, run the regression using the left pieces and predict the dependent variable for the leaving piece.

3.3 Stored results

`xtplfc` stores the following in `e()`:

Scalars

`e(N)`	number of individuals
`e(df_m)`	model degrees of freedom
`e(df_r)`	residual degrees of freedom
`e(r2)`	within R-squared
`e(r2_a)`	adjusted within R-squared
`e(rmse)`	root mean squared error
`e(mss)`	model sum of squares
`e(rss)`	residual sum of squares

Matrix

`e(b)`	coefficient vector in the linear part
`e(V)`	variance-covariance matrix of the estimators in the linear part
`e(bs)`	coefficient vector in the approximating model
`e(Vs)`	variance-covariance matrix of the estimators in the approximating model
`e(knots)`	number of knots
`e(power)`	number of power (or degree) of splines

Macros

`e(cmd)`	xtplfc
`e(depvar)`	name of dependent variable
`e(title)`	title in estimation output
`e(estfun)`	variables storing the estimated functional coefficients
`e(vcetype)`	type of variance-covariance
`e(model)`	Fixed-effect Series Semiparametric Estimation
`e(k#)`	list of knots for the #th function

3.4 Dependency of xtplfc

`xtplfc` depends on the moremata and bspline packages.

4 The ivxtplfc command

`ivxtplfc` estimates Zhang and Zhou's (2020) partially linear functional-coefficient panel data models with endogenous variables using the sieve 2SLS method.

4.1 Syntax

`ivxtplfc` *varlist*, `uvars(`*varlist*`)` `generate(`*string*`)` [`zvars(`*varlist*`)`
`endox(`*varlist*`)` `endozflag(`*numlist*`)` `ivx(`*varlist*`)` `ivz(`*varlist*`,`
`uflag(`*numlist*`)` `[ivtype(#)])` `power(`*numlist*`)` `nknots(`*numlist*`)`

quantile(*numlist*) maxnknots(*numlist*) maxnknots(*numlist*)
grid(*string*) pctile(#) brep(#) wild predict(*prspec*) nodots
level(#) fast tenfoldcv]

4.2 Options

uvars(*varlist*) specifies (continuous) variables that enter the functional coefficients interacted with variables specified by zvars() in order. It is required.

generate(*string*) specifies a prefix for the names to store fitted values of functional coefficients. It is required.

zvars(*varlist*) specifies variables that have functional coefficients.

endox(*varlist*) specifies endogeneous variables which enter the model linearly.

endozflag(*numlist*) specify the orders of variables in zvars(*varlist*) which are endogeneous variables. For example, endozflag(1 3) indicates that the first and third variables specified in zvars(*varlist*) are endogeneous.

ivx(*varlist*) specifies instrumental variables which enter the model linearly.

ivz(*varlist*, uflag(*numlist*) [ivtype(*numlist*)]) specify instrumental variables entering the model nonlinearly which interacts with the functions specified by the orders in uflag(numlist). Optionally, one may specify the type of nonlinear instrumental variables to be constructed. ivtype(#) means using the #th lag of the basis functions and the final IVs are formed from "ivz×L#.S(u)" (where S(u) are basis functions of u). By default, the first lag of the basis functions are used.

te specifies including time fixed effects.

power(*numlist*) specifies the power (or degree) of the splines; default is power(3).

nknots(*numlist*) specifies the number of knots used for the spline interpolation in order specified by uvars(). If absent, 2 is assumed for all the functions.

quantile specifies creating knots based on empirical quantiles. By default, the knots are generated by the rule of equal space.

maxnknots(*numlist*) specifies the maximun number of knots used for conducting least-squares cross-validation (LSCV). If present, LSCV is employed to determine the optimal number of knots. In our practice, we perform the Leave-One-Out CV across the panelvar. That is to say, we leave one individual (with all observations during the sample period) out each time.

minnknots(*numlist*) specifies the minimun number of knots used for performing LSCV. If absent, 2 is assumed.

grid(*string*) specifies a prefix for the names to store the grid points of the variable specified by uvars(*varlist*).

pctile(#) specifies the domain of the generating grid points; default is pctile(0).

brep(#) specifies the number of bootstrap replications. The default is bootstrap(200).

wild specifies using the wild bootstrap. By default, residual bootstrap with cluster(panelvar) is performed.

predict(*prspec*) stores predicted values of the conditional mean and fixed

effects using variable names specified in *prspec*. Specifically, the expression is predict(*varlist*| *stub** [, *replace noai*]). The option takes a variable list or a stub. The first variable name corresponds to the predicted conditional mean. The second name corresponds to fixed effects.When *replace* is used, variables with the names in *varlist* or *stub** are replaced by those in the new computation. If *noai* is specified, only a variable for the mean is created.

`nodots` suppresses iteration dots.
`level(#)` sets confidence level; default is level(95).
`fast` speeds up using mata functions.
`tenfoldcv` specifies using ten-fold CV.

4.3 Stored results

`ivxtplfc` stores the following in `e()`:

Scalars

`e(N)`	number of individuals
`e(df_m)`	model degrees of freedom
`e(df_r)`	residual degrees of freedom
`e(r2)`	within R-squared
`e(r2_a)`	adjusted within R-squared
`e(rmse)`	root mean squared error
`e(mss)`	model sum of squares
`e(rss)`	residual sum of squares

Matrix

`e(b)`	coefficient vector in the linear part
`e(V)`	variance-covariance matrix of the estimators in the linear part
`e(bs)`	coefficient vector in the approximating model
`e(Vs)`	variance-covariance matrix of the estimators in the approximating model
`e(knots)`	number of knots
`e(power)`	number of power (or degree) of splines

Macros

`e(cmd)`	ivxtplfc
`e(depvar)`	name of dependent variable
`e(title)`	title in estimation output
`e(estfun)`	variables storing the estimated functional coefficients
`e(vcetype)`	type of variance-covariance
`e(model)`	Fixed-effect Series Semiparametric Estimation
`e(k#)`	list of knots for the #th function
`e(ivlist)`	instrumental variables used for estimation.

4.4 Dependency of ivxtplfc

`ivxtplfc` depends on the moremata and bspline packages.

5 The xtdplfc command

`xtdplfc` estimates Zhang and Zhou's (2020) partially linear functional-coefficient panel dynamic data models using the sieve 2SLS method.

5.1 Syntax

`xtdplfc` *varlist*, `uvars(`*varlist*`)` `generate(`*string*`)` [`zvars(`*varlist*`)`
`lag(`#`)` `lagyinz(`*numlist*`)` `endox(`*varlist*`)` `endozflag(`*numlist*`)`
`ivx(`*varlist*`)` `ivz(`*varlist*`,` `uflag(`*numlist*`)` `[ivtype(`#`)])` `onlyivxz`
`ivtype(`*numlist*`)` `power(`*numlist*`)` `nknots(`*numlist*`)`
`quantile(`*numlist*`)` `maxnknots(`*numlist*`)` `grid(`*string*`)` `pctile(`#`)`
`brep(`#`)` `wild` `predict(`*prspec*`)` `nodots` `level(`#`)` `fast` `tenfoldcv`]

5.2 Options

`uvars(`*varlist*`)` specifies (continuous) variables that enter the functional coefficients interacted with variables specified by zvars() in order. It is required.

`generate(`*string*`)` specifies a prefix for the names to store fitted values of functional coefficients. It is required.

`zvars(`*varlist*`)` specifies variables that have functional coefficients.

`lag(`#`)` specifies using #lags of dependent variable as covariates; default is lags(1).

`lagyinz(`*numlist*`)` speficies lags of dependent variable that have functional coefficients. When this option is used, the specified lags of depvar are automatically added in the front of variables in zvars(*varlist*).

`endox(`*varlist*`)` specifies endogeneous variables which enter the model linearly.

`endozflag(`*numlist*`)` specify the orders of variables in zvars(*varlist*) which are endogeneous variables. For example, endozflag(1 3) indicates that the first and third variables specified in zvars(*varlist*) are endogeneous.

`ivx(`*varlist*`)` specifies instrumental variables which enter the model linearly.

`ivz(`*varlist*`,` `uflag(`*numlist*`)` `[ivtype(`*numlist*`)])` specify instrumental variables entering the model nonlinearly which interacts with the functions specified by the orders in uflag(numlist). Optionally, one may specify the type of nonlinear instrumental variables to be constructed. ivtype(#) means using the #th lag of the basis functions and the final IVs are formed from "ivz×L#.S(u)" (where S(u) are basis functions of u). By default, the first lag of the basis functions are used.

`onlyivxz` only uses instruments specified by ivx() and ivz(). By default, addtional instruments are automatically constructed using lags of dependent variables, variables specified by endox() and endozflag(), and the generating

splines.

`ivtype(`*`numlist`*`)` speficies the lag of the basis functions to be used for constructing the IVs. Suppose Z is an endogeneous variable interacting with g(U); S(U) are basis functions for g(U). ivtype(1) indicates constructing IVs from "L2.Z×L.S(U)".

`te` specifies including time fixed effects.

`power(`*`numlist`*`)` specifies the power (or degree) of the splines; default is power(3).

`nknots(`*`numlist`*`)` specifies the number of knots used for the spline interpolation in order specified by uvars(). If absent, 2 is assumed for all the functions.

`quantile` specifies creating knots based on empirical quantiles. By default, the knots are generated by the rule of equal space.

`maxnknots(`*`numlist`*`)` specifies the maximun number of knots used for conducting least-squares cross-validation (LSCV). If present, LSCV is employed to determine the optimal number of knots. In our practice, we perform the Leave-One-Out CV across the panelvar. That is to say, we leave one individual (with all observations during the sample period) out each time.

`minnknots(`*`numlist`*`)` specifies the minimun number of knots used for performing LSCV. If absent, 2 is assumed.

`grid(`*string*`)` specifies a prefix for the names to store the grid points of the variable specified by uvars(*varlist*).

`pctile(#)` specifies the domain of the generating grid points; default is pctile(0).

`brep(#)` specifies the number of bootstrap replications. The default is bootstrap(0).

`wild` specifies using the wild bootstrap. By default, residual bootstrap with cluster(panelvar) is performed.

`predict(`*`prspec`*`)` stores predicted values of the conditional mean and fixed effects using variable names specified in *prspec*. Specifically, the expression is predict(*varlist*| *stub** [, *replace noai*]). The option takes a variable list or a stub. The first variable name corresponds to the predicted conditional mean. The second name corresponds to fixed effects.When *replace* is used, variables with the names in *varlist* or *stub** are replaced by those in the new computation. If *noai* is specified, only a variable for the mean is created.

`nodots` suppresses iteration dots.

`level(#)` sets confidence level; default is level(95).

`fast` speeds up using mata functions.

`tenfoldcv` specifies using ten-fold CV.

5.3 Stored results

`xtdplfc` stores the following in `e()`:

Scalars

`e(N)`	number of individuals
`e(df_m)`	model degrees of freedom
`e(df_r)`	residual degrees of freedom
`e(r2)`	within R-squared
`e(r2_a)`	adjusted within R-squared
`e(rmse)`	root mean squared error
`e(mss)`	model sum of squares
`e(rss)`	residual sum of squares

Matrix

`e(b)`	coefficient vector in the linear part
`e(V)`	variance-covariance matrix of the estimators in the linear part
`e(bs)`	coefficient vector in the approximating model
`e(Vs)`	variance-covariance matrix of the estimators in the approximating model
`e(knots)`	number of knots
`e(power)`	number of power (or degree) of splines

Macros

`e(cmd)`	xtdplfc
`e(depvar)`	name of dependent variable
`e(title)`	title in estimation output
`e(estfun)`	variables storing the estimated functional coefficients
`e(vcetype)`	type of variance-covariance
`e(model)`	Fixed-effect Series Semiparametric Estimation
`e(k#)`	list of knots for the #th function
`e(ivlist)`	instrumental variables used for estimation.

5.4 Dependency of xtdplfc

`xtdplfc` depends on the moremata and bspline packages.

6 Monte Carlo simulation

In this section, we investigate the finite sample performance of estimators discussed above through Monte Carlo simulations. For all data generating processes (DGPs) to be considered, we set up a standard fixed effect panel including 50 individuals over 40 time periods, i.e., $N = 50, T = 40$. Five hundred replications are carried out.

We first consider the static panel data model as follows (DGP1):

$$Y_{it} = X_{1,it} - X_{2,it} + g(X_{3,it})Z_{it} + a_i + \varepsilon_{it} \tag{8}$$

$$g(X_{3,it}) = X_{3,it} + 2(X_{3,it})^2 - 0.25(X_{3,it})^3 \tag{9}$$

Following Libois et al. (2013), we generate $X_{1,it}, X_{2,it}, X_{3,it}$ and a_i via a

two-step procedure[2] as follows:

$$X_{1,it} = X^e_{1,it} \tag{10}$$

$$X_{2,it} = (X^f_{2,i} + X^e_{2,it})/\sqrt{2} \tag{11}$$

$$X_{3,it} = (X^f_{3,i} + X^e_{3,it})/\sqrt{2} \tag{12}$$

We draw $X^f_{2,it}, X^f_{3,it}$ and a_i from the multivariate normal distribution with mean $\mu = (0, 0, 0)$ and covariance matrix

$$\begin{array}{c} \\ X^f_{2,it} \\ X^f_{3,it} \\ a_i \end{array} \begin{array}{c} \begin{array}{ccc} X^f_{2,it} & X^f_{3,it} & a_i \end{array} \\ \begin{pmatrix} 1 & & \\ 0 & 1 & \\ 0.42 & 0.85 & 1 \end{pmatrix} \end{array}$$

Similarly, $X^e_{1,it}, X^e_{2,it}$ and $X^e_{3,it}$ are drawn from the multivariate normal distribution with mean $\mu = (0, 0, 0)$ and covariance matrix

$$\begin{array}{c} \\ X^e_{1,it} \\ X^e_{2,it} \\ X^e_{3,it} \end{array} \begin{array}{c} \begin{array}{ccc} X^e_{1,it} & X^e_{2,it} & X^e_{3,it} \end{array} \\ \begin{pmatrix} 1 & & \\ 0.2 & 1 & \\ 0.8 & 0 & 1 \end{pmatrix} \end{array}$$

For comparison, we consider the following three regression models.

Model 1: xtplfc, considering that $X_{1,it}$ and $X_{2,it}$ enter the model linearly, whereas Z_{it} is included with a functional coefficient of $X_{3,it}$.

Model 2: xtreg, regressing Y_{it} on $X_{1,it}$, $X_{2,it}$,$X_{3,it}$, and Z_{it} with fixed effects.

Model 3: xtreg, regressing Y_{it} on $X_{1,it}$, $X_{2,it}$ $X_{3,it}$, Z_{it} and $c.X_{3,it}\#c.Z_{it}$ with fixed effects.

The simulation codes for DGP 1 are presented as follows [3]:

```
. clear all
. set matsize 2000
. set obs 50
number of observations (_N) was 0, now 50
.
. set seed 123456
. matrix C = (1,0,0.42\0,1,0.85\0.42,0.85,1)
. drawnorm x2f x3f d, corr(C)
. gen id = _n
```

[2]The generating covariates are consisted of two components. The first one is generated for each individual and then duplicated by T times, indicating that they are fixed for each individual. The second one is a random realization for each time.

[3]To run the simulation code, Baum & Azevedo's "outtable" package and UCLA Statistical Consulting Group's "graph2tex" packge should be installed in advance. "outtable" can be installed by "ssc install outtable". "graph2tex" can be installed by "net install graph2tex, from(https://stats.idre.ucla.edu/stat/stata/ado/analysis)."

```
. expand 40
(1,950 observations created)
. bysort id : gen t = _n
. xtset id t
panel variable:  id (strongly balanced)
time variable:  t, 1 to 40
delta:  1 unit
. gen y = 0
. matrix D = (1,0.2,0.8\0.2,1,0\0.8,0,1)
. drawnorm x1 x2e x3e, corr(D)
. gen x2 = (x2f+x2e)/sqrt(2)
. gen x3 = (x3f+x3e)/sqrt(2)
. gen z=rnormal()
. gen gf=1*x3+ 2*x3^2 - 0.25*(x3)^3
.
. forv j=1/500{
2.          cap drop e
3.          cap drop *_sd
4.          qui drawnorm e
5.          qui replace y = x1 -x2 + z*gf+ d + e
6.          qui xtplfc y x1 x2, z(z) u(x3) maxnk(20) gen(fit`j´) fast brep(500)
7.          matrix B=e(b)
8.          matrix B=B[1,1..2]
9.          matrix B1=(nullmat(B1)\B)
10.         matrix V=e(Vs)
11.         matrix V=vecdiag(V)
12.         matrix V=V[1,1..2]
13.         matrix V1=(nullmat(V1)\V)
14.
.           qui xtreg y x1 x2 x3 z,fe
15.         matrix B=e(b)
16.         matrix B=B[1,1..2]
17.         matrix B2=(nullmat(B2)\B)
18.         matrix V=e(V)
19.         matrix V=vecdiag(V)
20.         matrix V=V[1,1..2]
21.         matrix V2=(nullmat(V2)\V)
22.
.
.           qui xtreg y x1 x2 x3 z c.x3#c.z,fe
23.         matrix B=e(b)
24.         matrix B=B[1,1..2]
25.         matrix B3=(nullmat(B3)\B)
26.         matrix V=e(V)
27.         matrix V=vecdiag(V)
28.         matrix V=V[1,1..2]
29.         matrix V3=(nullmat(V3)\V)
30.
.
. }
.
.
. * Fig.1
.
. egen av_fit = rowmean(fit*)
.
. egen sd_fit = rowsd(fit*)
```

```
.
.
. gen  c =  invnormal(1 - (100 - 95) / 200)
.
. gen low = av_fit - c * sd_fit
.
. gen up = av_fit + c * sd_fit
.
. twoway (rarea low up x3, sort(x3) color(gs7)) ///
>    (line av_fit x3, sort(x3) color(black) lpattern(solid)) ///
>    (line gf x3, color(gs10) sort lpattern(longdash)), ///
>    legend(label(1 confidence interval at 95%) label(3 DGP) label(2 average fit) ///
>    cols(3) order(3 1 2)) xtitle("X3",height(5)) ytitle("g(X3)",height(5)) sch(sj)
. graph2tex , epsfile(fig1) caption(Average fit of g(X3)) label(fig1)
% exported graph to fig1.eps
% We can see in Figure \ref{fig:fig1} that
\begin{figure}[h]
\begin{centering}
\includegraphics[height=3in]{fig1}
\caption{Average fit of g(X3)}
\label{fig:fig1}
\end{centering}
\end{figure}
.
.
. * Table 1
. clear
. set obs 500
number of observations (_N) was 0, now 500
. mat res1=J(3,8,.)
.
. forv k=1/3{
2.         qui svmat B`k´
3.         su B`k´1,meanonly
4.         mat res1[`k´,1]=r(mean)-1
5.         su B`k´2,meanonly
6.         mat res1[`k´,2]=r(mean)+1
7.
.          qui svmat V`k´
8.         qui replace V`k´1=sqrt(V`k´1)
9.         qui gen B`k´1_lb=B`k´1-invnormal(0.975)*V`k´1
10.        qui gen B`k´1_ub=B`k´1+invnormal(0.975)*V`k´1
11.
.          qui gen CIlen`k´1=B`k´1_ub-B`k´1_lb
12.        qui su CIlen`k´1,d
13.        mat res1[`k´,5]=r(p50)
14.
.          qui replace V`k´2=sqrt(V`k´2)
15.        qui gen B`k´2_lb=B`k´2-invnormal(0.975)*V`k´2
16.        qui gen B`k´2_ub=B`k´2+invnormal(0.975)*V`k´2
17.
.          qui gen CIlen`k´2=B`k´2_ub-B`k´2_lb
18.        qui su CIlen`k´2,d
19.        mat res1[`k´,6]=r(p50)
20.
.          qui gen cov`k´1=(B`k´1_lb<=1 & B`k´1_ub>=1)
21.        su cov`k´1,meanonly
```

```
22.         mat res1[`k´,7]=r(mean)
23.
.           qui gen cov`k´2=(B`k´2_lb<=-1 & B`k´2_ub>=-1)
24.         su cov`k´2,meanonly
25.         mat res1[`k´,8]=r(mean)
26.
.           qui replace B`k´1=(B`k´1-1)^2
27.         su B`k´1,meanonly
28.         mat res1[`k´,3]=r(mean)
29.         qui replace B`k´2=(B`k´2+1)^2
30.         su B`k´2,meanonly
31.         mat res1[`k´,4]=r(mean)
32.
.
. }
. outtable using res1, mat(res1)  format(%9.4f) replace
```

Following Burton et al. (2006) and White (2010), we report the bias, mean square error (MSE), median width of 95% confidence interval, and coverage of 95% confidence interval of the estimated coefficients associated with $X_{1,it}$ and $X_{2,it}$ in Table 1. We can find that the fixed-effect sieve estimator outperforms the fixed-effect estimator in terms of both bias and MSE. As expected, the bias are relatively large in Models 2 and 3 since they suffer from endogeneity due to omitting the nonlinear relationship of $X_{3,it}$ and Z_{it}. In terms of the 95% confidence interval, the fixed-effect sieve estimator gives rise to a relatively small interval width. Moreover, coverage probabilities of 95% confidence interval generated by the fixed-effect sieve estimator are quite close to the nominal value(0.95). The average fit of $g(X_{it})$ alongside with the corresponding 95% confidence band in the simulations is presented in Fig. 1. It can be seen that the average fit is very close to the true function $g(X_{it})$ and the 95% confidence band is relatively small except for the edges.

Table 1: Simulation results for the parametric parts in DGP 1

	Bias		MSE		95% CI width		Coverage	
	$X_{1,it}$	$X_{2,it}$	$X_{1,it}$	$X_{2,it}$	$X_{1,it}$	$X_{2,it}$	$X_{1,it}$	$X_{2,it}$
Model 1	0.0016	0.0026	0.0007	0.0014	0.1084	0.1559	0.9540	0.9560
Model 2	-0.0547	-0.0769	0.0045	0.0071	0.5667	0.4891	1.0000	1.0000
Model 3	-0.0830	-0.0605	0.0084	0.0048	0.5595	0.4828	1.0000	1.0000

Next, we consider the case of dynamic panel data (DGP2):

$$Y_{it} = g(U_{it})Y_{it-1} + 0.3Y_{it-2} + 0.3X_{it} + a_i + \varepsilon_{it} \tag{13}$$

$$g(U_{it}) = sin(\frac{\pi}{3}U_{it}) \tag{14}$$

$$X_{it} = W_{it} + 0.5a_i \tag{15}$$

We assume ε_{it} are IID $N(0,1)$ across both i and t and a_i are IID $N(0,1)$. W_{it} and U_{it} are drawn from IID $U(0,10)$ and IID $U(-9,9)$, respectively.

We compare the performance of ivxtplfc and ivregress 2sls via the following models:

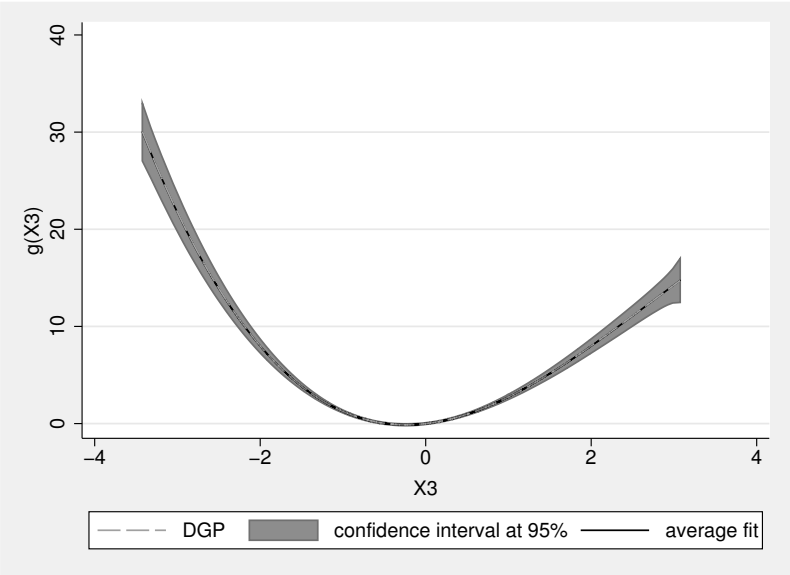

Figure 1: Average fit of $g(X_3)$ across replications

Model 4: ivxtplfc, considering that X_{it} and Y_{it-2} enter the model linearly, whereas Y_{it-1} is included with a functional coefficient of U_{it}.

Model 5: ivregress 2sls, taking first difference to remove fixed effects and regressing ΔY_{it} on ΔY_{it-1}, ΔY_{it-2}, and ΔX_{it}.

Model 6: ivregress 2sls, taking first difference to remove fixed effects and regressing ΔY_{it} on ΔY_{it-1}, ΔY_{it-2}, ΔX_{it} and ΔU_{it}.

It should be noted that $L2.Y_{it}$ (the second lag of Y_{it}) is used as the instrumental variable in "ivregress 2sls", and the interaction of $L2.Y_{it}$ and the first lag of the basis functions of U_{it} are constructed as the instrumental variables in "ivxtplfc". The simulation codes for DGP 2 are presented as follows:

```
. clear all
. set matsize 2000
. set obs 50
number of observations (_N) was 0, now 50
. set seed 789
. gen a=rnormal()
. gen id=_n
. expand 80
(3,950 observations created)
. bys id: gen year=_n
. gen x=10*runiform()+0.5*a
. gen u=-9+18*runiform()
. gen gf=sin(_pi/3*u)
```

```
. gen y=0
. xtset id year
panel variable:  id (strongly balanced)
time variable:  year, 1 to 80
delta:  1 unit
. mata: gfmat=J(2000,500,.)
. forv j=1/500{
2.
.       preserve
3.      qui replace y=a+0.3*x+0.3*L2.y+L1.y*gf+sqrt(2)*rnormal() if year>2
4.      qui drop if year<41
5.      qui gen L_y=L1.y
6.      qui gen L2_y=L2.y
7.      qui ivxtplfc y L2_y x, zvars(L_y) uvar(u) gen(g) endoz(1) ///
>                                 maxnknots(20) ivz(L2_y,uflag(1)) fast brep(500)
8.      qui putmata gfhat=g_1,replace
9.      mata: gfmat[.,`j´]=gfhat
10.     matrix B=e(b)
11.     matrix B=B[1,1..2]
12.     matrix B1=(nullmat(B1)\B)
13.     matrix V=e(Vs)
14.     matrix V=vecdiag(V)
15.     matrix V=V[1,1..2]
16.     matrix V1=(nullmat(V1)\V)
17.
.
.       qui ivregress 2sls D.y D.L2.y D.x (D.L.y=L2.y), noconstant
18.     matrix B=e(b)
19.     matrix B=B[1,2..3]
20.     matrix B2=(nullmat(B2)\B)
21.     matrix V=e(V)
22.     matrix V=vecdiag(V)
23.     matrix V=V[1,1..2]
24.     matrix V2=(nullmat(V2)\V)
25.
.       qui ivregress 2sls D.y D.L2.y D.x D.u (D.L.y=L2.y), noconstant
26.     matrix B=e(b)
27.     matrix B=B[1,2..3]
28.     matrix B3=(nullmat(B3)\B)
29.     matrix V=e(V)
30.     matrix V=vecdiag(V)
31.     matrix V=V[1,1..2]
32.     matrix V3=(nullmat(V3)\V)
33.
.       restore
34.
. }
.
.
. * Fig 2
.
. qui drop if year<41
. qui getmata (gfs*)=gfmat
. egen av_fit = rowmean(gfs*)
. egen sd_fit = rowsd(gfs*)
. gen  c =  invnormal(1 - (100 - 95) / 200)
. gen low = av_fit - c * sd_fit
```

```
. gen up = av_fit + c * sd_fit
.
.         twoway (rarea low up u, sort(u) color(gs7)) ///
>            (line av_fit u, sort(u) color(black) lpattern(solid)) ///
>            (line gf u, color(gs10) sort lpattern(longdash)), ///
>            legend(label(1 confidence interval at 95%) label(3 DGP) label(2 average fit) ///
>            cols(3) order(3 1 2)) xtitle("U", height(5)) ytitle("g(U)", height(5)) sch(sj)
. graph2tex , epsfile(fig2) caption(Average fit of g(U)) label(fig2)
% exported graph to fig2.eps
% We can see in Figure \ref{fig:fig2} that
\begin{figure}[h]
\begin{centering}
\includegraphics[height=3in]{fig2}
\caption{Average fit of g(U)}
\label{fig:fig2}
\end{centering}
\end{figure}
.
.
. * Table 2
. clear
. set obs 500
number of observations (_N) was 0, now 500
. mat res2=J(3,8,.)
. forv k=1/3{
2.      qui svmat B`k´
3.      su B`k´1,meanonly
4.      mat res2[`k´,1]=r(mean)-0.3
5.      su B`k´2,meanonly
6.      mat res2[`k´,2]=r(mean)-0.3
7.
.       svmat V`k´
8.      qui replace V`k´1=sqrt(V`k´1)
9.      qui gen B`k´1_lb=B`k´1-invnormal(0.975)*V`k´1
10.     qui gen B`k´1_ub=B`k´1+invnormal(0.975)*V`k´1
11.
.       qui gen CIlen`k´1=B`k´1_ub-B`k´1_lb
12.     qui su CIlen`k´1,d
13.     mat res2[`k´,5]=r(p50)
14.
.       qui replace V`k´2=sqrt(V`k´2)
15.     qui gen B`k´2_lb=B`k´2-invnormal(0.975)*V`k´2
16.     qui gen B`k´2_ub=B`k´2+invnormal(0.975)*V`k´2
17.
.       qui gen CIlen`k´2=B`k´2_ub-B`k´2_lb
18.     qui su CIlen`k´2,d
19.     mat res2[`k´,6]=r(p50)
20.
.       qui gen cov`k´1=(B`k´1_lb<=0.3 & B`k´1_ub>=0.3)
21.     su cov`k´1,meanonly
22.     mat res2[`k´,7]=r(mean)
23.
.       qui gen cov`k´2=(B`k´2_lb<=0.3 & B`k´2_ub>=0.3)
24.     su cov`k´2,meanonly
25.     mat res2[`k´,8]=r(mean)
26.
.       qui replace B`k´1=(B`k´1-0.3)^2
27.     su B`k´1,meanonly
28.     mat res2[`k´,3]=r(mean)
```

```
29.     qui replace B`k´2=(B`k´2-0.3)^2
30.     su B`k´2,meanonly
31.     mat res2[`k´,4]=r(mean)
32.
. }
. outtable using res2, mat(res2)  format(%9.4f) replace
```

Table 2 presents the bias, mean square error (MSE), median width of 95% confidence interval, and coverage of 95% confidence interval of the estimated coefficients associated with Y_{it-2} and X_{it}. Similar to DGP1, the simulation results show that the fixed-effect sieve 2SLS estimator performs much better than the fixed-effect 2SLS estimator in terms of bias, MSE, width of 95% confidence interval, and coverage of 95% confidence interval. Fig. 2 displays the average fit of $g(U_{it})$ alongside with the corresponding 95% band in the simulations. Although $g(U_{it})$ is fluctuated greatly in Model 4, the proposed method estimates it quite well.

Table 2: Simulation results for the parametric parts in DGP 2

	Bias		MSE		95% CI width		Coverage	
	Y_{it-2}	X_{it}	Y_{it-2}	X_{it}	Y_{it-2}	X_{it}	Y_{it-2}	X_{it}
Model 4	-0.0009	-0.0005	0.0002	0.0002	0.0542	0.0527	0.9360	0.9400
Model 5	0.0302	0.0357	0.0017	0.0018	0.3623	0.2555	1.0000	1.0000
Model 6	0.0266	0.0385	0.0015	0.0020	0.3603	0.2539	1.0000	1.0000

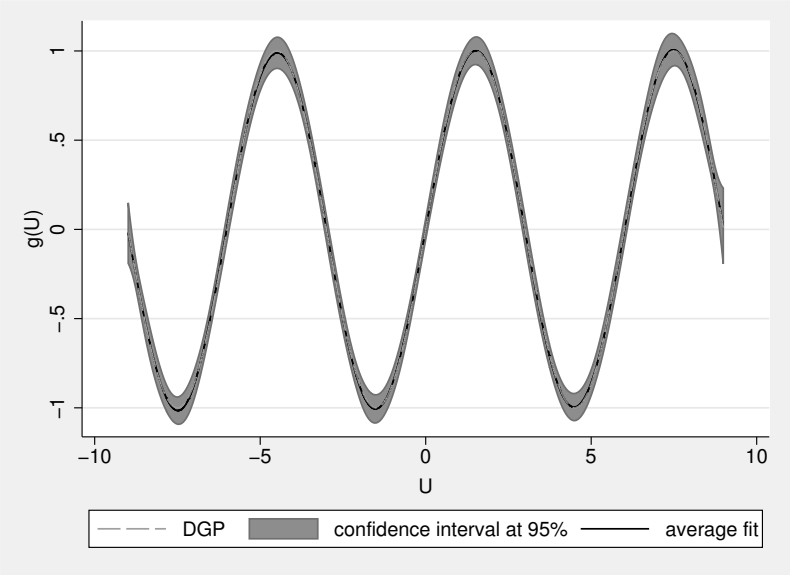

Figure 2: Average fit of g(U) across replications

7 Conclusion

Partial linear functional-coefficient models are flexible enough to accommodate the nonlinear structure and capture the heterogeneity over individuals and times which have been popularly in academic research. This article briefly introduces the newly development of functional-coefficient panel data models, and provides three new Stata commands for implementing estimation. Additionally, we illustrate the usefulness of our proposed commands via some simple simulations.

8 Acknowledgments

Kerui Du acknowledges financial support from the National Natural Science Foundation of China (No. 71603148). Qiankun Zhou acknowledges financial support from the National Natural Science Foundation of China (No. 71431006). We are grateful to the anonymous reviewer for the helpful comments and suggestions which led to an improved version of this paper.

9 References

An, Y., C. Hsiao, and D. Li. 2016. Semiparametric Estimation of Partially Linear Varying Coefficient Panel Data Models. In *Essays in Honor of Aman Ullah (Advances in Econometrics, Volume 36)*, 47–65. Emerald Group Publishing Limited.

Burton, A., D. G. Altman, P. Royston, and R. L. Holder. 2006. The design of simulation studies in medical statistics. *Statistics in Medicine* 25(24): 4279–4292.

Cai, Z., J. Fan, and R. Li. 2000a. Efficient Estimation and Inferences for Varying-Coefficient Models. *Journal of the American Statistical Association* 95(451): 888–902.

Cai, Z., J. Fan, and Q. Yao. 2000b. Functional-coefficient regression models for nonlinear time series. *Journal of the American Statistical Association* 95(451): 941–956.

Cai, Z., Y. Fang, M. Lin, and J. Su. 2017. Inferences for a Partially Varying Coefficient Model With Endogenous Regressors. *Journal of Business & Economic Statistics* 0(0): 1–13.

Cai, Z., and Q. Li. 2008. Nonparametric estimation of varying coefficient dynamic panel data models. *Econometric Theory* 24(5): 1321–1342.

Cai, Z., Q. Li, and J. Y. Park. 2009. Functional-coefficient models for nonstationary time series data. *Journal of Econometrics* 148(2): 101–113.

Chen, R., and R. S. Tsay. 1993. Functional-coefficient autoregressive models. *Journal of the American Statistical Association* 88(421): 298–308.

Delgado, M. S., N. McCloud, and S. C. Kumbhakar. 2014. A generalized empirical model of corruption, foreign direct investment, and growth. *Journal of Macroeconomics* 42: 298 – 316.

Feng, G., J. Gao, B. Peng, and X. Zhang. 2017. A varying-coefficient panel data model with fixed effects: Theory and an application to US commercial banks. *Journal of Econometrics* 196(1): 68 – 82.

Huang, J. Z., C. O. Wu, and L. Zhou. 2004. Polynomial spline estimation and inference for varying coefficient models with longitudinal data. *Statistica Sinica* 763–788.

Li, J., H. Liu, and K. Du. 2019. Does market-oriented reform increase energy rebound effect? Evidence from China's regional development. *China Economic Review* 56: 101304.

Libois, F., V. Verardi, et al. 2013. Semiparametric fixed-effects estimator. *Stata journal* 13(2): 329–336.

Lundberg, A. L., K. P. Huynh, and D. T. Jacho-Chávez. 2017. Income and democracy: A smooth varying coefficient redux. *Journal of Applied Econometrics* 32(3): 719–724.

Newson, R. 2001. B-splines and splines parameterized by their values at reference points on the x-axis. *Stata Technical Bulletin* 10(57).

Su, L., I. Murtazashvili, and A. Ullah. 2013. Local linear GMM estimation of functional coefficient IV models with an application to estimating the rate of return to schooling. *Journal of Business & Economic Statistics* 31(2): 184–207.

Sun, Y., R. J. Carroll, and D. Li. 2009. Semiparametric estimation of fixed-effects panel data varying coefficient models. In *Nonparametric econometric methods*, 101–129. Emerald Group Publishing Limited.

White, I. R. 2010. simsum: Analyses of simulation studies including Monte Carlo error. *Stata Journal* 10(3): 369–385.

Zhang, Y., and Q. Zhou. 2020. Partially Linear Functional-Coefficient Dynamic Panel Data Models: Sieve Estimation and Specification Testing. *Forthcoming at Econometric Reviews* .

About the authors

Kerui Du is an associate professor at School of Management, Xiamen University. His primary research interests are applied econometrics, energy and environmental economics.

Yonghui Zhang (corresponding author) is an assistant professor of Economics at School of Economics, Renmin University of China. His primary research interests are both theoretical and applied econometrics, with a focus on nonparametric econometrics.

Qiankun Zhou is an associate professor of Economics at the Department of Economics of Louisiana State University. His primary research interests are both theoretical and applied econometrics, with a focus on panel data econometrics.