Guide to Replication Files for Cunha, Heckman, and Schennach (2010)

1 Introduction

These notes are a guide to the replication files in Cunha, Heckman, and Schennach (2010). The following table lists the folders that are part of the replication material:

Table

Folder Name	Description of Material in the Folder
Code	The code used in the estimation
Data	The dataset in stata format
Simulation	Codes used in the simulation to replicate Figures
Table I&II	Material to replicate Tables I & II
Table III	Material to replicate Table III
Table IV	Material to replicate Table IV
Table V	Material to replicate Table V

We start the note with a review of state space models. This will allow readers interested in replicating the results or learning how to use the code by setting the notation that is used in the software. After reading these notes and working through the examples, readers should have a familiarity with entering information to use the code for their own purposes, replicate the tables in the paper, and run the simulations that generate the figures.

2 Review of State Space Models

2.1 Gaussian Linear State Space Models

Let $t = 1, ..., T$. Let $y_t \in M(p_t \times 1)$ denote the observed measurements. Let $\alpha_t \in M(m \times 1)$ denote the state vector (unobserved). Let $\varepsilon_t \in M(p_t \times 1)$ denote the measurement noise, and $\eta_t \in M(m \times 1)$ the transition disturbances. The multivariate Gaussian linear state space model is

given by:

$$y_t = X\beta_t + Z_t\alpha_t + \varepsilon_t \text{ where } \varepsilon_t \sim N(0, H_t) \tag{1}$$

$$\alpha_{t+1} = G_t\alpha_t + \eta_t \text{ where } \eta_t \sim N(0, Q_t) \tag{2}$$

$$\alpha_1 \sim N(a_1, P_1). \tag{3}$$

Equation (1) is called the measurement equation. It relates the measured observable variables that provide information on α_t. We use $Z_t \in M(p_t \times m)$ to denote the matrix of factor loadings. The $H_t \in M(p_t \times p_t)$ matrix is the variance-covariance matrix of the measurement noise vector, ε_t. Equation (2) is called the transition equation. We use $G_t \in M(m \times m)$ to denote the matrix of factor coefficients. The transition equation models how the state vector α_t evolves over time. In Cunha and Heckman (2007), the technology of skill formation is modeled as a transition equation. It tells us how current child's skills and parental investments today increase the child's skills tomorrow. Finally, (3) is the initial condition of the system.

The matrices Z_t, H_t, G_t, Q_t are called system matrices. The state space model is said to be time invariant when $Z_t = Z$, $H_t = H$, $G_t = G$, and $Q_t = Q$ for $t = 1, ..., T$. In many practical situations the state space model can be set up as time invariant.

2.2 The Kalman Filter

Let $p(y) = p(y_1, ..., y_T)$ denote the likelihood of the model (1),(2), and (3). It is possible to estimate the matrices Z_t, H_t, G_t, Q_t by maximizing this likelihood. However, it is easier to work with the conditional likelihood:

$$p(y) = p(y_1)\prod_{t=2}^{T} p(y_t|\, y_{t-1}, ..., y_1). \tag{4}$$

One can obtain the conditional likelihood by using the Kalman Filter. The idea is that, because of the linearity and normality of the model we are using, the likelihood $p(y)$ is the likelihood of a normal random vector. Consequently, each term $p(y_t|\, y_{t-1}, ..., y_1)$ is also the likelihood of a normal random variable. So, to obtain this function we only need to characterize its mean and variance. The Kalman filter generates a set of recursions that given the mean and variance of $y_t|y_{t-1}, ..., y_1$ we obtain the mean and variance of $y_{t+1}|y_{t,}..., y_1$.

Define

$$a_{t+1} = E[\alpha_{t+1}|\, Y_t]$$
$$P_{t+1} = Var[\alpha_{t+1}|\, Y_t]$$

Note that

$$E[y_{t+1}|\, X, Y_t] = X\beta_t + Z_t a_t$$
$$Var[y_{t+1}|\, X, Y_t] = Z_t P_{t+1} Z_t' + H_t.$$

The Kalman filter for the state space model (1)-(3) can be written in the form (Harvey, 1989):

$$\begin{aligned} v_t &= y_t - Z_t a_t \\ F_t &= Z_t P_t Z_t' + H_t \\ K_t &= P_t Z_t' \\ a_{t+1} &= G_t \left(a_t + K_t F_t^{-1} v_t \right) \\ P_{t+1} &= G_t \left(P_t - K_t F_t^{-1} K_t' \right) G_t' + Q_t \end{aligned} \tag{5}$$

2.3 Mixture of Gaussian Nonlinear State Space Models

More generally, consider the model:

$$y_t = g\left(X, \alpha_t\right) + \varepsilon_t \text{ where } \varepsilon_t \sim N\left(0, H_t\right) \tag{6}$$

$$\alpha_{t+1} = f\left(\alpha_t\right) + \eta_t \text{ where } \eta_t \sim N\left(0, Q_t\right) \tag{7}$$

$$\alpha_1 \sim N\left(a_1, P_1\right). \tag{8}$$

The conceptual solution of the nonlinear filtering is simple. We break the problem into a prediction and update step and then proceed recursively. The prediction step generates $p\left(\alpha_t | \, y^{t-1}\right)$ given knowledge of $p\left(\alpha_{t-1} | \, y^{t-1}\right)$. This is accomplished by applying the Chapman-Kolmogorov equation:

$$p\left(\alpha_t | \, y^{t-1}\right) = \int p\left(\alpha_t | \, \alpha_{t-1}\right) p\left(\alpha_{t-1} | \, y^{t-1}\right) d\alpha_{t-1}.$$

where $p\left(\alpha_t | \, \alpha_{t-1}\right)$ is the density of α_t conditional on α_{t-1}. The update step computes $p\left(\alpha_t | \, y^t\right)$ given $p\left(\alpha_t | \, y^{t-1}\right)$ via the Bayes' rule:

$$p\left(\alpha_t | \, y^t\right) = \frac{p\left(y_t | \, \alpha_t\right) p\left(\alpha_t | \, y^{t-1}\right)}{p\left(y_t | \, y^{t-1}\right)}.$$

A simple solution to the filtering problem exists when the functions g and f are linear and separable in each of their arguments, the unobserved state α_t is Gaussian, and the noise terms ε_t, η_t are Gaussian, independent random variables. In this case, one can use the Kalman Filter to derive the equations used in the prediction and update steps analytically. However, simple departures of this framework (e.g. f is nonlinear) makes the Kalman Filter unsuitable. It is possible to adapt this approach by considering the first-order Taylor series approximation of the function f and then apply the standard Kalman Filter prediction and update rules. This is known in the filtering literature as the Extended Kalman Filter (EKF). The problem with this approach is that the EKF generally generates biased expressions for means and variances.

More recently, researchers have used general Particle Filtering techniques.[1] However, in the context of panel data with a large cross-section dimension, the Particle Filter can be computationally

[1] See, for example, [?]; [?]

costly. Furthermore, the Particle Filter may not be a good tool if the goal of the researcher is to estimate the (parameters of the) functions f or g especially when these functions are time invariant.

Another approach is to consider the Unscented Kalman Filter (UKF). The crucial assumption in this algorithm is that both $p\left(\alpha_t|\, y^t\right)$ and $p\left(\alpha_{t+1}|\, y^t\right)$ can be accurately approximated by the density of a normal random variable with mean:

$$a_{t+k,t} = E\left(\alpha_{t+k}|\, y^t\right)$$

and variance

$$P_{t+k,t} = Var\left(\alpha_{t+k}|\, y^t\right)$$

for $k \in \{0, 1\}$. Because of this assumption, the only objects that have to undergo the prediction and update steps are the means and variances of the approximating normal distribution, just as in the standard Kalman Filter algorithm.

Obviously, in some situations the normal approximation may not be a good one. It is possible that nonlinear functions of normally distributed random variables generate random variables that have densities are not symmetric around their means or with many modes, which would be inconsistent with a normal approximation. We introduce a more flexible approach which considers approximations that use mixture of normals:

$$p\left(\alpha_{t+k}|\, y^t\right) \simeq \sum_{l=1}^{L} \tau_{l,t}\phi\left(\alpha_t; a_{l,t+k,t}, P_{l,t+k,t}\right)$$

where $\phi\left(\alpha_t; a_{l,t+k,t}, P_{l,t+k,t}\right)$ is the probability density function of a normal random variable with mean $a_{l,t+k,t}$ and variance $P_{l,t+k,t}$, for $k \in \{0, 1\}$. The weights $\tau_{l,t}$ are such that $\tau_{l,t} \in [0, 1]$ and $\sum_l^L \tau_{l,t} = 1$. Under this formulation, within each stage, we break the filtering problem into parallel problems and obtain the final result at the end.

2.4 The Update Step

If the measurement equations are linear, then the update step is just like in the Kalman Filter.

First, we compute the update density for each element of the mixture. Namely, let $\hat{y}_{l,t}$ denote the predicted measurement by the l^{th} element of the mixture:

$$\hat{y}_{l,t} = E_l\left(y_t|\, X, Y^{t-1}\right) = E_l\left[g\left(X, \alpha_t\right)|\, X, Y^{t-1}\right] + E_l\left[\varepsilon_t|\, Y^{t-1}\right] = E_l\left[g\left(X, \alpha_t\right)|\, X, Y^{t-1}\right] \quad (9)$$

Below, we show how to compute the moment above. For now, consider the following updating equations:

$$a_{l,t,t} = a_{l,t,t-1} + K_{l,t}\left(y_t - \hat{y}_{l,t}\right) \quad (10a)$$

$$P_{l,t,t} = P_{l,t,t-1} - K_{l,t}F_{l,t}K_{l,t}' \quad (10b)$$

where:

$$K_{l,t} = Cov\left[\alpha_t, y_t | X, y^{t-1}\right] F_{l,t}^{-1} \tag{10c}$$

and

$$F_{l,t} = Var\left[g\left(X, \alpha_t\right) | X, Y^{t-1}\right] + H_t. \tag{10d}$$

We can then approximate the posterior density $p\left(\alpha_t | Y^t\right)$ with a linear combination of densities $\phi\left(\alpha_t; a_{l,t,t}, P_{l,t,t}\right)$ with weights given by:

$$\tau_{r,t} = \frac{\tau_{r,t-1}\phi\left(y_t; \hat{y}_{r,t}, F_{r,t}\right)}{\sum_{l=1}^{L} \tau_{l,t-1}\phi\left(y_t; \hat{y}_{l,t}, F_{l,t}\right)},\ r \in \{1, \ldots, L\}. \tag{10e}$$

2.5 The Prediction Step

With knowledge of a good approximation for the density $p\left(\alpha_t | Y^t\right)$ expressed as the mixture of normals and knowledge of the transition equation (**??**) one can approximately compute the one-step-ahead prediction density $p\left(\alpha_{t+1} | Y^t\right)$ also expressed as a mixture of normals. More precisely, let:

$$a_{l,t+1,t} = E_l\left(\alpha_{t+1} | Y^t\right) = E_l\left(f\left(\alpha_t\right) + \eta_{t+1} \middle| Y^t\right) = E_l\left(f\left(\alpha_t\right) | Y^t\right) \tag{11}$$

$$P_{l,t+1,t} = Var_l\left[\alpha_{t+1} | Y^t\right] = Var\left(f\left(\alpha_t\right) + \eta_{t+1} \middle| Y^t\right) = Var\left(f\left(\alpha_t\right) | Y^t\right) + Q_{t+1} \tag{12}$$

Then, an approximation to $p\left(\alpha_{t+1} | Y^t\right)$ is given by:

$$p\left(\alpha_{t+1} | Y^t\right) \approx \sum_{l=1}^{L} \tau_{l,t}\phi\left(y_t; a_{l,t+1,t}, P_{l,t+1,t}\right).$$

2.6 Unscented Transform

A difficulty arises in the implementation of the filtering because in the prediction and update stages one has to compute integrals that involve nonlinear transformations of random variables whose distributions are approximated by mixtures of normals. The Unscented Transform (UT) is a convenient tool to compute the mean and variance of a random variable that undergoes a nonlinear transformation. For example, consider computing the expressions (11) and (12). Then, by definition:

$$a_{l,t+1,t} = \int f\left(\alpha_t\right) \phi\left(\alpha_t; a_{l,t,t}, P_{l,t,t}\right) d\alpha_t \tag{13}$$

$$P_{l,t+1,t} = \int \left(f\left(\alpha_t\right) - a_{l,t+1,t}\right)\left(f\left(\alpha_t\right) - a_{l,t+1,t}\right)' \phi\left(\alpha_t; a_{l,t,t}, P_{l,t,t}\right) d\alpha_{t-1} + Q_{t+1} \tag{14}$$

The expressions (13) and (14) involve the computation of m integrals. One way to proceed is to consider the product of quadrature rules. The difficulty with this approach is that as m becomes larger, the number of evaluations increases exponentially.

Another approach is to consider monomial rules. The Unscented Transform is a monomial rule

that approximates the expressions (13) and (14). To do so, one picks deterministically $2m+1$ points $\bar{\alpha}_{N,l,t,t}$ and corresponding weights $w_{N,l,t}$, $n = 0, 1, ..., 2m$. Let $P_{l,t,t}(n,:)$ denote the n^{th} row of the $(m \times m)$ matrix $P_{l,t,t}$. Let $\kappa \in \mathbb{R}$ such that $\kappa + m \neq 0$. The UT proposes the following points $x_{N,l,t,t}$:

$$\begin{aligned}
\bar{\alpha}_{N,l,t,t} &= a_{l,t,t} && \text{for } n = 0. \\
\bar{\alpha}_{N,l,t,t} &= a_{l,t,t} + \sqrt{(N_\alpha + \kappa) P_{l,t,t}(n,:)} && \text{for } n = 1, ..., m. \\
\bar{\alpha}_{N,l,t,t} &= a_{l,t,t} - \sqrt{(N_\alpha + \kappa) P_{l,t,t}(n,:)} && \text{for } n = N_\alpha + 1, ..., 2m.
\end{aligned} \tag{15}$$

and the following weights $w_{N,l,t}$:

$$\begin{aligned}
w_{l,N,t} &= \tfrac{\kappa}{m+\kappa} && \text{for } n = 0. \\
w_{l,N,t} &= \tfrac{1}{2(m+\kappa)} && \text{for } n = 1, ..., m. \\
w_{l,N,t} &= \tfrac{1}{2(m+\kappa)} && \text{for } n = N_\alpha + 1, ..., 2m.
\end{aligned}$$

We approximate $E_l[f(\alpha_t)|Y^t]$ and $Var_l[f(\alpha_t)|Y^t]$ by computing:

$$a_{l,t+1,t} = E_l\left[f(\alpha_t)|Y^t\right] \approx \sum_{n=0}^{2m} w_{l,N,t} f(\bar{\alpha}_{N,l,t,t})$$

and

$$P_{l,t,t} = Var\left[f(\alpha_t)|Y^t\right] + Q_t \approx \sum_{n=0}^{2N_\alpha} w_{N,l,t}\left[f(\bar{\alpha}_{N,l,t,t}) - a_{l,t+1,t}\right]\left[f(\bar{\alpha}_{N,l,t,t}) - a_{l,t+1,t}\right]' + Q_t$$

2.7 Implementation of Nonlinear Filtering

Let $p(y)$ denote the likelihood (4):

$$p(y) = p(y_1) \prod_{t=2}^{T} p\left(y_t|y^{t-1}\right). \tag{16}$$

The idea is to use the nonlinear filtering to obtain a recursive algorithm which we can use to calculate $p(y_{t+1}|y^t)$. To see how, note that we assume that

$$p(\alpha_1) \approx \sum_{l=1}^{L} \tau_{l,0} \phi(\alpha_1; a_{l,1,0}, P_{l,1,0})$$

It follows that:

$$p(y_1) \approx \sum_{l=1}^{L} \tau_{l,0} \phi(y_1; \hat{y}_1, F_{l,1})$$

where $\hat{y}_1$ and $F_{l,1}$ are defined in (9) and (10d). Now, by applying (10a), (10b), (10c), and (10e) allow us to obtain $a_{l,1,1}$, $P_{l,1,1}$, and $\tau_{l,1}$ which is really helpful because now we can characterize the

posterior density as:

$$p\left(\alpha_1 | Y_1\right) \approx \sum_{l=1}^{L} \tau_{l,1} \phi\left(\alpha_1; a_{l,1,1}, P_{l,1,1}\right).$$

We then apply the prediction steps to obtain $a_{l,2,1}$ and $\Sigma_{l,2,1}$. With knowledge of these quantities, we can approximate the predicted density as:

$$p\left(\alpha_2 | Y_1\right) \approx \sum_{l=1}^{L} \tau_{l,1} \phi\left(\alpha_2; a_{l,2,1}, P_{l,2,1}\right).$$

And now we complete the cycle, because by using (9)-(10d) we can compute $\hat{y}_2$ and $F_{l,2}$, with which we can compute:

$$p\left(y_2 | y_1\right) \approx \sum_{l=1}^{L} \tau_{l,1} \phi\left(y_2; \hat{y}_2, F_{l,2}\right).$$

Furthermore, we use (10e) to update the weights $\tau_{l,2}$. By proceeding in a recursive manner, we can construct the right-hand side of (16), which is just equal to the likelihood.

3 Code Instructions

The code is written in Fortran/90. It contains the following routines:

Name	Function
globvar.f90	Here you define the basics of the model: Number of periods, individuals, etc...
initialize.f90	Here you describe the organization of the data set you want to estimate.
mappings.f90	Here you inform the likelihood the parameters to be estimated.
stdeviation.f90	This routine computes standard errors.
utilities.f90	This module has the UT Algorithm and specification of the functions f and g
write_results.f90	Routine that prints out estimates and standard errors.
likelihood.f90	It contains the routine that will call the likelihood.
like_aux.f90	Here you find the recursions of the Kalman Filter.
matrix.f90	Auxiliary routine: Matrix Inversion, etc...
minimization.f90	Routine that minimizes the negative of the likelihood.
probability.f90	Routine that is used to evaluate the density function of the normal random variable.
statistics.f90	Routine that computes means, standard errors, etc...
main.f90	"The Code"

3.1 The Module globvar.f90

The globvar.f90 routine has the information you need to enter. You will say the number of individuals ($nind$), the number of time periods ($ntime$), the number of factors ($nfac$), the number of

measurement equations ($nmea$), the dimension of X_t (nx)[2], and the number of stages that you want to estimate ($nstage$). The matrices G and Q can vary over time, t. If you prefer some periods to have the same matrix G, then you can set those to be the same stages. The remaining objects in this routine should not be changed.

3.2 The Module initialize.f90

3.2.1 Reading the data set for the estimation

This routine is used to inform the code how your data is organized. However, it is important that you supply the data to the code in the following "long" format:

person id	period	$y_{1,t}$	$x_{3,t}$	$y_{2,t}$	... $y_{n,t}$	$dy_{1,t}$	$dy_{2,t}$	... $dy_{n,t}$
1	1	12	1	1	5	1	1	1
1	2	−100	1	1	7	0	1	1
1	3	11	2	−100	15	1	0	1
1	4	14	3	3	12	1	1	1
2	1	−100	4	2	9	0	1	1
2	2	12	3	2	12	1	1	1
2	3	13	3	−100	22	1	0	1
2	4	−100	5	0	8	0	1	1

You supply that total number of variables in your data set (including the variable person id and period) in nvar. This is the total number of columns in your data set. The name of the data set is supplied in "character(len=100) :: datafile=filename.raw".

Next, you supply the number of measurement equation per period. This is done so because some measurements may be available at some periods, but not at others. In the example that comes ready with the code, there are six equations at each period. Then, you inform the stage number at each period. Here we have only one stage, which is the same for all periods.

The information for the code to find the location of each measurement variable, that is, the vector y_t in (1) should be entered in the vector ly, which has dimension $nmea$. The vector ldy has the location of the dy variables, which inform the code whether the value for y_t for a given person is missing (in which case $dy = 0$) or not ($dy = 1$).

The integer-valued matrix $eqindex\,(t, nequation(t))$ informs the code the measurement equations you want to estimate in period t. This information may change with time periods, so you should adjust it accordingly.

The integer-valued matrix $mx\,(t, nequation(t))$ informs the code the number of explanatory variables that you want to include in period t and equation $nequation(t)$. The tensor $lx(t, j, mx\,(t, j))$

[2]The number of observable variables in each equation at each time period can change. Here you have to inform the maximum number of observable equations you will have. For example, if you have two measurement equations, the first has 4 observable variables and the second has 5 observable variables, you should set $nx = 5$.

informs the location of each explanatory variable you want to include in period t, equation j, variable $mx(t,j)$.

3.2.2 Imposing the normalizations in the parameter vector

The second part of the routine is devoted to initializing the parameters and imposing the normalizations you may wish. It is here that you will specify which factor loadings (elements of the matrix Z_t) that will be normalized.

> Tip: When you are running a large model, it is usually a good idea to constrain the factor loadings you want to estimate, Z_t, the elements of the transition matrix, G_t, and the initial variance matrix, P_0 to some value and let the code find good initial values for the vector β_t. If you start with bad initial values, this may make the code "crash". The reason is that the last iteration of the Kalman Filter:
>
> $$P_{t+1} = G_t \left(P_t - K_t F_t^{-1} K_t'\right) G_t' + Q_t$$
>
> The object on the left-hand side is the variance matrix (or, if it is scalar, a variance, which is a positive scalar). On the RHS, you have a subtraction. There is no guarantee that the resulting number will be positive. This is very likely to occur when you have bad initial values. This should not happen when you have good values for β_t.

3.3 The Modules mappings.f90 and stdeviation.f90

There are two routines in this module. The first one, whose name is *dimtheta*, simply counts the number of parameters to be estimated. It then informs the code the number *dimtheta* which is the dimension of the parameter vector *theta* to be estimated.

The second routine, called *getpar*, is the routine that transforms the vector *theta* in the parameters as written in (1),(2), and (3), that is, the elements of the system matrices $\beta_t, Z_t, H_t, G_t, Q_t$.

The stdeviation.f90 module is very parallel to the mappings.f90. The difference is that mappings.f90 is used for the point estimates, while the stdeviation.f90 is used, obviously, for the computation of standard errors. There are two routines here. The first is called *bootstrap* and it stores the bootstrap simulation results. The second is called *standard_error* and computes the standard error of the point estimates.

4 Examples

4.1 Example 1

To show how to use the code, consider the following example. Suppose that there are two factors, so $m = 2$. Assume that there are two independent variables x_1 and x_2. At each period t there are

three measures exclusively on factor α_{1t} :

$$y_{k,t} = \beta_{k,1} x_1 + \beta_{k,2} x_2 + Z_{k,t,1} \ln \alpha_{1,t} + \varepsilon_{k,t} \text{ for } k = 1, 2, 3 \tag{17}$$

At each period t there are three measures exclusively on factor α_{2t} :

$$y_{k,t} = \beta_{k,1} x_1 + \beta_{k,2} x_2 + Z_{k,t,2} \ln \alpha_{2,t} + \varepsilon_{k,t} \text{ for } k = 4, 5, 6. \tag{18}$$

As in Cunha, Heckman, and Schennach (2008) we assume that factor one needs an anchor. Let Q_1 denote such anchor:

$$Q_1 = \delta_1 x_1 + \delta_2 x_2 + \lambda_1 \alpha_{1,T} + \nu_1$$

Factor one involves according to a nonlinear function:

$$\ln \alpha_{1,t+1} = \frac{1}{\phi \lambda_1} \ln \left\{ \gamma_{1,1} e^{\phi \lambda_1 \ln \alpha_{1,t}} + \gamma_{1,2} e^{\phi \ln \alpha_{2,t}} \right\} + \eta_{1,t+1},$$

subject to $\gamma_{1,1}, \gamma_{1,2} \geq 0$ and $\gamma_{1,1} + \gamma_{1,2} = 1$.

Factor two involves according to a linear function:

$$\ln \alpha_{2,t+1} = \gamma_{2,2} \ln \alpha_{2,t} + \eta_{2,t+1}.$$

This model is implemented in three folders. We generate fake data using the STATA file data.do. The reader can read that file to see the true parameters used for the model above. Folder "LinearAnchor-V0" applies the algorithm above as stated with one exception: The anchoring parameter, λ_1, is set at is true value. As the reader can see, the algorithm recovers the true parameters relatively well. Folder "LinearAnchor-V1" estimates λ_1. The fact that λ_1 is estimated has a small effect on the estimated parameters of the technology, both point estimates as well as standard errors. Folder "LinearAnchor-V2" implements the algorithm above without computing the square root of the variance-covariance matrix, which is computationally intensive. As the reader can see, there is little effect on the estimates of the technology as well as standard errors.

4.2 Example 2

To show how to use the code, consider the following example. Suppose that there are three factors, so $m = 3$, which follow exactly the equations (17) and (18) in Example 1. Factor 3 is a static one (as parental skills in Cunha, Heckman, and Schennach, 2008). We assume that there are measurements for factor 3, α_3, only in period one:

$$y_{k,1} = \beta_{k,1} x_1 + \beta_{k,2} x_2 + Z_{k,1} \ln \alpha_3 + \varepsilon_{k,1} \text{ for } k = 1, 2, 3$$

As in Cunha, Heckman, and Schennach (2008) we assume that factor one needs an anchor. Let Q_1 denote such anchor:

$$Q_1 = \delta_1 x_1 + \delta_2 x_2 + \lambda_1 \alpha_{1,T} + \nu_1$$

Factor one evolves according to a nonlinear function:

$$\ln \alpha_{1,t+1} = \frac{1}{\phi \lambda_1} \ln \left\{ \gamma_{1,1} e^{\phi \lambda_1 \ln \alpha_{1,t}} + \gamma_{1,2} e^{\phi \ln \alpha_{2,t}} + \gamma_{1,3} e^{\phi \ln \alpha_3} \right\} + \eta_{1,t+1},$$

subject to $\gamma_{1,k} \geq 0$ for $k = 1, 2, 3$ and $\sum_{k=1}^{3} \gamma_{1,k} = 1$.

Factor two evolves according to a linear function:

$$\ln \alpha_{2,t+1} = \gamma_{2,2} \ln \alpha_{2,t} + \eta_{2,t+1}.$$

Since Factor 3 is static, it evolves according to:

$$\alpha_{3,t+1} = \alpha_{3,t}$$