`FCVARmodel.m`: A Matlab software package for estimation and testing in the fractionally cointegrated VAR model[*]

version 1.0

Morten Ørregaard Nielsen[†]
Queen's University and CREATES
Email: `mon@econ.queensu.ca`

Lealand Morin
Queen's University
Email: `morinl@econ.queensu.ca`

August 22, 2011

Abstract

This manual describes the usage of the accompanying freely available software package, written by the authors for estimation and testing in the fractionally cointegrated vector autoregressive (VAR) model.

JEL Codes: C22, C32

Keywords: cofractional process, cointegration rank, fractional autoregressive model, fractional cointegration, fractional unit root, VAR model

[*]We are grateful to Søren Johansen and James MacKinnon for comments and to the Social Sciences and Humanities Research Council of Canada (SSHRC grant 410-2009-0183) and the Center for Research in Econometric Analysis of Time Series (CREATES, funded by the Danish National Research Foundation) for research support.

[†]Corresponding author

Contents

1 Obtaining and using the software package

1.1 Obtaining the software package

The software package can be downloaded from the first author's website at Queen's University:

`http://www.econ.queensu.ca/faculty/nielsen/`

It is freely available for non-commercial, academic use.

1.2 Citation

If you use this software, or any program or computer codes based on this software, we ask that you please cite this document. For example, you could add "The results were obtained using the computer software by Nielsen and Morin (2011)" in the main text of your paper and then add the citation

Nielsen, M. Ø. and L. Morin (2011). "FCVARmodel.m: A Matlab software package for estimation and testing in the fractionally cointegrated VAR model." QED working paper xxxx, Queen's University.

to your list of references.

1.3 Using the software package

The use of this software package requires a functioning installation of Matlab. Any recent version should work.

The next section describes the fractionally cointegrated VAR model and the restricted models that can be estimated with this software package. Section 3 describes the functioning of the main program, and section 4 describes the options file and the various settings. With these programs, both the unrestricted and restricted models can be estimated. It is then straightforward to calculate likelihood ratio test statistics and perform the tests. The cointegration rank tests can be calculated automatically using the program in section 5.1 and the model parameters can be estimated using the program in section 5.2. Section 5.3 describes how to obtain residuals for model diagnostics and section 6 describes auxilliary files.

1.4 Version history

v1.0 (August 2011): First publicly available version.

2 The fractionally cointegrated VAR model

The fractionally cointegrated vector autoregressive (VAR) model was proposed in Johansen (2008) and analyzed by Johansen and Nielsen (2010, 2011), henceforth JN. For a time series X_t of dimension p, the fractionally cointegrated VAR model is given in error correction form as

$$\Delta^d X_t = \Delta^{d-b} L_b \alpha(\beta' X_t - \rho') + \sum_{i=1}^{k} \Gamma_i \Delta^d \ L_b^i X_t + \varepsilon_t, \quad t = 1, \dots, T, \tag{1}$$

where ε_t is p-dimensional $i.i.d.(0, \Omega)$, $d \geq b > 0$, Δ^d is the fractional difference operator, and $L_b = 1 - \Delta^b$ is the fractional lag operator.

The model (1) includes the Johansen (1996) cointegrated VAR model as the special case $d = b = 1$, and the interpretation of the model parameters is similar. Thus, we can write $\Pi = \alpha\beta'$, where the $p \times r$ matrices α and β, with $r \leq p$, are assumed to have full column rank r. Then the columns of β are the r cointegrating (cofractional) relations that determine the long-run equilibria, and α holds the adjustment coefficients. The parameter $\Gamma = (\Gamma_1, \dots, \Gamma_k)$ governs the short-run dynamics. The parameter ρ is the so-called restricted constant term (since the constant term in the model is restricted to be of the form $\mu = \alpha\rho'$), which is interpreted (asymptotically) as the mean level of the long-run equilibria, $E\beta' X_t = \rho'$. The rank r is termed the cointegrating, or cofractional, rank. The expansion of $L_b = 1 - \Delta^b$ has no term in L^0, and thus only lagged disequilibrium errors appear in (1).

In the next three subsections we briefly describe estimation and testing within the model (1). For details we refer to JN (2011).

2.1 Maximum likelihood estimation

The model (1) is estimated by conditional maximum likelihood (given initial values) by minimizing the function

$$\log L_T(\lambda) = -\frac{T}{2} \log\det(T^{-1} \sum_{t=1}^{T} \varepsilon_t(\lambda)\varepsilon_t(\lambda)'\}, \tag{2}$$

$$\varepsilon_t(\lambda) = \Delta^d X_t - \Delta^{d-b} L_b \alpha(\beta' X_t - \rho') - \sum_{i=1}^{k} \Gamma_i \Delta^d \ L_b^i X_t, \tag{3}$$

where $\lambda = (d, b, \alpha, \beta, \rho, \Gamma)$. It is shown in JN (2011) how, for fixed (d, b) the estimation reduces to regression and reduced rank regression as in Johansen (1996). In this way the parameters $(\alpha, \beta, \rho, \Gamma)$ can be concentrated out of the likelihood function, and numerical optimization is only needed to optimize the profile likelihood function over the two fractional parameters, d and b.

JN (2011) shows that asymptotic theory is standard when $b < 0.5$, and for the case $b > 0.5$ asymptotic theory is non-standard and involves fractional Brownian motion of type II. Specifically, when $b > 0.5$, JN (2011) shows that under i.i.d. errors with suitable moment conditions, the conditional maximum likelihood parameter estimates $(\hat{d}, \hat{b}, \hat{\alpha}, \hat{\Gamma}_1, \dots, \hat{\Gamma}_k)$ are asymptotically Gaussian, while $(\hat{\beta}, \hat{\rho})$ are locally asymptotically mixed normal. These results allow asymptotically standard (chi-squared) inference on all parameters of the mode, including the cointegrating relations and orders of fractionality, using quasi-likelihood ratio tests.

2.2 Cointegration rank tests

The non-standard asymptotic distributions appear for the likelihood ratio (LR) test statistic of the hypothesis $\mathcal{H}_r$: rank$(\Pi, \mu) = r$ against $\mathcal{H}_p$: rank$(\Pi, \mu) = p$. Let $L(d, b, r)$ be the profile likelihood function, where $\alpha, \beta, \rho, \Gamma_1, \ldots, \Gamma_k, \Omega$ have been concentrated out by regression and reduced rank regression; see JN (2011) for details. To calculate the LR statistic, we maximize the profile likelihood function both under the hypothesis $\mathcal{H}_r$ and under $\mathcal{H}_p$, see the following sections. Letting $L(d, b, r)$ denote the profile likelihood function given rank r, we then have $\mathrm{LR}_\mathrm{T}(q) = 2\log(L(\hat{d}_p, \hat{b}_p, p)/L(\hat{d}_r, \hat{b}_r, r))$, where $q = p - r$,

$$L(\hat{d}_p, \hat{b}_p, p) = \max_{d,b} L(d, b, p), \text{ and } L(\hat{d}_r, \hat{b}_r, r) = \max_{d,b} L(d, b, r).$$

The asymptotic distribution of this test statistic is derived in JN (2011), where it is shown that, when $0.5 < b \leq d$,

$$\mathrm{LR}_\mathrm{T}(q) \xrightarrow{D} \mathrm{Tr}\left\{\int_0^1 dW(s)F'(s)\left(\int_0^1 F(s)F'(s)ds\right)^{-1}\int_0^1 F(s)dW'(s)\right\} = \mathrm{LR}(q), \qquad (4)$$

where the vector process dW is the increment of ordinary $q = p - r$-dimensional vector Brownian motion. The vector process F depends on the deterministics similar to the cointegrated VAR model in Johansen (1996). When no deterministic term is in the model, $F(u) = W_b(u)$, and when the restricted constant term is included, $F(u) = (W_b'(u), u^{-(d-b)})'$, where $W_b(u)$ is vector fractional Brownian motion of type II defined as

$$W_b(u) = \Gamma(b)^{-1}\int_0^u (u - s)^{b-1} dW(s). \qquad (5)$$

The asymptotic distribution $\mathrm{LR}(q)$ of the test statistic $\mathrm{LR}_\mathrm{T}(q)$ defined in (4) depends on both (d, b) and $q = p - r$, although it does not depend on d when no deterministic terms are included. The dependence on the unknown parameters d and b complicates empirical analysis compared to the cointegrated VAR model. However, computer programs for computing asymptotic critical values and asymptotic P values for tests of cointegration rank in the fractional model (1), based on numerical distribution functions, are made available by MacKinnon and Nielsen (2011).

2.3 Restricted models

Note that a reduced rank restriction has already been imposed on model (1), where the coefficient matrix $\Pi = \alpha\beta'$ has been restricted to rank $r \leq p$. Other restrictions on the model parameters can be considered as in Johansen (1996). The most interesting restrictions from an economic point of view would likely be restrictions on the adjustment parameters α and cointegration vectors β.

We consider restrictions on α and β of two forms. The parameters can be partitioned into sets of columns, each of which can be restricted either to known values or to a linear subspace. Partition α and β into $\alpha = [\alpha_1, \alpha_2, \ldots]$ and $\beta = [\beta_1, \beta_2, \ldots]$ such that each α_i and

β_i are $p \times r_i$ matrices with $\sum_i r_i = r$. Restrictions to known values a_i and b_i can be specified for each α_i and β_i, respectively. Linear restrictions can be imposed by specifying matrices A_i and H_i so that $\alpha_i = A_i\psi_i$ and $\beta_i = H_i\phi_i$. Any submatrices without such restrictions provided are estimated unrestrictedly.

The estimation is performed using a switching algorithm as in Johansen (1996). The parameters for each pair of submatrices α_i and β_i are estimated in turn by taking the other submatrices as fixed. The switching algorithm repeats the estimation through the sequence of submatrices until convergence.

In the case of deterministics represented by the restricted constant term ρ, restrictions must be of the same form as those for corresponding columns of β. This is because the estimation is performed by estimating the two variables jointly through the matrix $\beta^* = (\beta', \rho)'$. For columns of β specified as known, ρ must also be restricted to known values. For submatrices β_i with linear restrictions, each ρ_i can be estimated separately.

For the remaining parameters, any linear restrictions can be imposed, with the exception of the Γ_i matrices, for which only exclusion restrictions are permitted.

The restrictions described above can be implemented individually or simultaneously by the software package. The restrictions are imposed by setting the appropriate options, and the implementation is described in section 4 below.

3 Main program: `FCVARmodel.m`

MORTEN: Following this choice of title for the main program, I decided to change the names of all functions with names like fracARXXX to FCVARXXX. Also note that this section is extended to reflect the new use of RestrictEstnOptions separate from DefaultEstnOptions.

This program demonstrates the estimation of the fractionally cointegrated VAR model. There are four main function calls, each of which are described in detail in subsequent sections.

- The function `DefaultEstnOptions(db0, k, r, p)` is called to set estimation options, see section 4.
- The function `RankTests(x, k, db0, FCVARoptions)` is called to perform tests to determine cointegrating rank, see section 5.1. Note that the array `FCVARoptions` is the output from the function `DefaultEstnOptions` called above.
- The function `FCVARestn(x, k, r, db0, FCVARoptions)` is called to estimate the model, see section 5.2.
- The function `GetResiduals(x, k, r, dbHat, alphaHat, betaHat, rhoHat, GammaHat, EstnOptions)` is called to compute the residuals, see section 5.3. These could then be used for model diagnostics. The array `EstnOptions` is output from the function `DefaultEstnOptions`.
- The function `RestrictEstnOptions(db0, k, r, p, defaultFCVARoptions)` is called to set estimation options to impose restrictions on parameter estimates, see section 4. This will provide the specifications to estimate a restricted model.
- The functions `FCVARestn(x, k, r, db0, FCVARoptions)` and `GetResiduals(x, k, r, dbHat, alphaHat, betaHat, rhoHat, GammaHat, EstnOptions)` are again called to estimate the restricted model and compute the residuals, see sections 5.2 and 5.3. The second estimation can be used to perform likelihood ratio tests of parameter restrictions.

4 Options files

4.1 Default estimation options: `DefaultEstnOptions.m`

The function `DefaultEstnOptions(db0, k, r, p)` in this file returns a set of preliminary setings corresponding to the default set of options specified in the file. These options are set by the user.

The output from this file is later processed automatically by the function `AdjustEstnOptions(db0, k, r, p, initialFCVARoptions)`, see section 6, which completes the specification of the options array. Note that the latter function does not need any action from the user. The options specified in `DefaultEstnOptions.m` govern all options with the exception of those for restrictions on parameter estimates, which are specified in the file `RestrictEstnOptions.m` discussed below.

LEE: Is the above a better description than before? Is it correct? MORTEN: I modified this to reflect separation of Default and RestrictEstnOptions. Now restrictions are imposed separately. To change options, I envision a user creating multiple DefaultEstnOptions files (ex: DefaultEstnOptions1, etc.) each time options are changed. Then they don't have to play with AdjustEstnOptions at all.

Inputs

- `db0`: vector or scalar of starting values (in the numerical optimization) for the fractional differencing parameters, d and b. The size of `db0` indicates whether or not restriction $d = b$ is imposed (if `db0` is scalar the restriction is imposed).
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `p`: scalar denoting the number of variables in the system.

Output

The output is the cell array `FCVARoptions = {` `dbEstnOptions, EstnOptions` `}`, an array of 2 arrays of options for use in many other routines.

`dbEstnOptions = {` `constrained, C_db, c_db, R_db, r_db, dbFminOptions` `}` is an array of options for the estimation of the fractional differencing parameters d and b.

- `constrained`: scalar indicating if the restrictions `dbMax` $\geq d \geq b \geq$ `dbMin` are imposed, given user inputs `dbMax` and `dbMin`.

  LEE: In which file (or files) can the value 0.01 be changed?

  MORTEN: The tolerance dbMin is introduced here to impose an inequality restriction using the equality restriction $dbMax >= d >= b >= dbMin$. Note the renaming of D as dbMax to match. These are now both set here in DefaultEstnOptions.

- `C_db` and `c_db` are matrices for imposing restrictions of the form `C_db` $* (d, b)' \leq$ `c_db`. These are produced automatically by the function `AdjustEstnOptions(db0, k, r, p, initialFCVARoptions)` from the user-defined inputs `dbMax` and `dbMin`.

  - `dbMax`: scalar denoting the upper bound for the estimation of d and b.
  - `dbMin`: scalar denoting the lower bound for the estimation of d and b. It is used to ensure positive d and b through the equality restriction `dbMax` $\geq d \geq b \geq$ `dbMin`.

- `R_db` and `r_db`: optional matrices for imposing restrictions of the form `R_db` $*\ (d, b)' =$ `r_db`. These are left blank in `DefaultEstnOptions(db0, k, r, p)` and only used in `RestrictEstnOptions(db0, k, r, p, restrictFCVARoptions)`.

- `dbFminOptions`: array of options for built-in Matlab (numerical optimization) functions for the optimization over d and b.

  LEE: I added "for the optimization over d and b", is that correct? MORTEN: Yes.

`EstnOptions = {` `N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, [], [], [], [], [], [], [], print2screen, printGammas, restrictFminOptions` `}` is a cell array of optional arguments for the estimation of the remaining parameters.

- `N`: scalar denoting the number of initial values of X that are to be conditioned upon in the estimation of the model.

- `M`: scalar denoting the truncation in the fractional differencing operator. If `M` > 0, the fractional differencing filter is truncated at lag `M`, which may speed up computations. If `M` $= 0$ there is no truncation.

- `deterministics`: string to determine the deterministics included in the model. The options are `'restricted constant'` and `' none '`. Note the spaces around the word 'none', for which a sample is commented out in `DefaultEstnOptions.m`. Anything other than `'restricted constant'` is interpreted as 'none'.

- `R_Gamma` and `r_Gamma` are optional matrices for imposing restrictions of the form `R_Gamma`$*$ $\Gamma =$ `r_Gamma`. These are left blank in `DefaultEstnOptions(db0, k, r, p)` and only used in `RestrictEstnOptions(db0, k, r, p, restrictFCVARoptions)`.

- `nCols, A, H, a, b, h` are optional matrices for use when estimating restricted models with Johansen's switching algorithm. These are also left blank in `DefaultEstnOptions(db0, k, r, p)` and only used in `RestrictEstnOptions(db0, k, r, p, restrictFCVARoptions)`.

  LEE: The following need to be added to the comments in the top of the file DefaultEstnOptions.m. MORTEN: You bet.

- `uGridMax, uGridStep, charPolyDegree`: ??

  LEE: I have no idea what this is. MORTEN: To be deleted.

- `print2screen`: scalar indicator to print results (from `RankTests.m` and `FCVARestn.m`) to screen and to generate graphs of characteristic roots.

- `printGammas`: scalar indicator to print coefficients of lagged differences as well.

  LEE: Did I get the above two bullets right? MORTEN: Yes.

- `restrictFminOptions`: array of options for the switching algorithm to optimize over all the other parameters besides d and b. This array has a format similar to that in `dbFminOptions`.

  LEE: Did I get this right? Is it really for the GetParamsSwitching? Then we should just write that. MORTEN: Yes, that's right. It used to be for the other restricted estn method as well. Well, actually it is of the same form as the built-in Matlab options and is used in the same way within the GetParamsSwitching function - but no matlab optimization is used there.

LEE: The comments at the top of this file need to be updated.

4.2 Restricted parameter estimation options: `RestrictEstnOptions.m`

The function `RestrictEstnOptions(db0, k, r, p, defaultFCVARoptions)` returns a set of preliminary setings corresponding to the restrictions on parameters. These restrictions are set by the user. It is to be run after `DefaultEstnOptions(db0, k, r, p)` is already used to set other options not directly related to restrictions on parameter estimates.

Inputs

- `db0`: vector or scalar of starting values (in the numerical optimization) for the fractional differencing parameters, d and b. The size of `db0` indicates whether or not restriction $d = b$ is imposed (if `db0` is scalar the restriction is imposed).
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `p`: scalar denoting the number of variables in the system.
- `defaultFCVARoptions`: array of options output from `DefaultEstnOptions(db0, k, r, p)`.

Output

The output is the cell array `restrictFCVARoptions =` $\{$ `dbEstnOptions, EstnOptions` $\}$, an array of 2 arrays of options for use in many other routines.

`dbEstnOptions =` $\{$ `constrained, C_db, c_db, R_db, r_db, dbFminOptions` $\}$ is an array of options for the estimation of the fractional differencing parameters d and b.

- `C_db` and `c_db`: optional matrices for imposing restrictions of the form `C_db` $*\,(d, b)' =$ `c_db`. These are generated to reflect the options `constrained`, `dbMax` and `dbMin` through the function `AdjustEstnOptions(db0, k, r, p, initialFCVARoptions)`, first called in `DefaultEstnOptions(db0, k, r, p)`. These matrices can be modified by the user to impose additional linear inequality restrictions on d and b.
- `R_db` and `r_db`: optional matrices for imposing restrictions of the form `R_db` $*\,(d, b)' =$ `r_db`.
- The options `constrained`, `dbMax`, `dbMin` and `dbFminOptions` are unchanged from those in `defaultFCVARoptions`.

`EstnOptions =` $\{$ `N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, [], [], [], [], [], [], [], print2screen, printGammas, restrictFminOptions` $\}$ is a cell array of optional arguments for the estimation of the remaining parameters.

- `R_Gamma` and `r_Gamma` are optional matrices for imposing restrictions of the form `R_Gamma`$*$ $\Gamma =$ `r_Gamma`. Note: This is intended for exclusion restrictions only and does not work with other types of restrictions!

- `nCols, A, H, a, b, h` are optional matrices for use when estimating restricted models with Johansen's switching algorithm.
  - nCols: vector of dimensions r_i of the partitioned matrices of α and β, where $\sum_i r_i = r$.
  - `A`: array of linear restrictions on each alpha such that alpha_i = A_i*psi_i.
  - `H`: array of linear restrictions on each beta such that beta_i = H_i*phi_i.
  - `a`: array of values of α considered known.
  - `b`: array of values of β considered known.
  - `h`: array of vectors specifying restrictions on ρ. Note that restrictions on ρ must be of the same type as those for β. For known β_i, ρ_i must be specified as known but for linear restriction of β_i to a subspace, elements of ρ_i can be estimated separately. MORTEN: Note the modification to the rule for restrictions on ρ. I would like to know your thoughts about which restrictions on rho would be economically interesting.
  - The options `N`, `M`, `deterministics`, `print2screen` and `printGammas` are unchanged from those in `defaultFCVARoptions`.

5 Other main files

5.1 Rank tests: `RankTests.m`

The function `RankTests(x, k, db0, RankTestOptions)` in this file performs a sequence of likelihood ratio tests for cointegrating rank.

Also prints the results to screen depending on the indicator `print2screen` set in the file `DefaultEstnOptions.m`, please see section 4.

LEE: This needs to be updated in the comments in the top of the file.

Inputs

- `x`: vector or matrix of data.
- `k`: scalar denoting the number of lags.
- `db0`: vector or scalar specifying starting values for the estimation of the fractional differencing parameters d and b, constrained to be equal if `db0` is a scalar.
- `RankTestOptions =` { `dbEstnOptions, EstnOptions` }: an array of 2 arrays of options for use in estimation routines, see sections 4 and 6.1 for details.
  - `dbEstnOptions =` { `constrained, C_db, c_db, R_db, r_db, dbFminOptions` } is an array of options for the estimation of the fractional differencing parameters d and b.
  - `EstnOptions =` { `N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, Abar, Aperp, replaceCols, keepCols, [], [], [], print2screen, printGammas, restrictFminOptions` } is an array of options for estimating the remaining parameters, please see sections 4 and 6.1 for details.

    LEE: This needs to be updated in the file.

Output

`rankTestStats`: vector of cointegrating rank test statistics.

Dependencies

Calls the function `FCVARlike`, see section 6.

5.2 Model estimation: `FCVARestn.m`

The function `FCVARestn(x, k, r, db0, FCVARoptions)` in this file estimates the fractional cointegration model under desired options.

Also prints the results to screen and creates a graph of the characteristic roots, depending on the indicator `print2screen`. Depending on the additional indicator `printGammas`, the estimates `GammaHat` are also printed. The indicators are set in `DefaultEstnOptions.m` and `RestrictEstnOptions.m`, please see section 4.

LEE: Did I get this right? This needs to be updated in the comments in the top of the file. MORTEN. Ok.

Inputs

- `x`: vector or matrix of data.
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `db0`: vector or scalar specifying starting values for the estimation of the fractional differencing parameters d and b, constrained to be equal if `db0` is a scalar.
- `FCVARoptions = {` `dbEstnOptions, EstnOptions` `}`: an array of 2 arrays of options for use in estimation routines, see sections 4 and 6.1 for details.
  - `dbEstnOptions = {` `constrained, C_db, c_db, R_db, r_db, dbFminOptions` `}` is an array of options for the estimation of the fractional differencing parameters d and b.
  - `EstnOptions = {` `N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, Abar, Aperp, replaceCols, keepCols, [], [], [], print2screen, printGammas, restrictFminOptions` `}` is an array of options for estimating the remaining parameters, please see sections 4 and 6.1 for details.

    LEE: This needs to be updated in the file.

Output

`estimates = {` `dbHat, alphaHat, betaHat, rhoHat, PiHat, OmegaHat, GammaHat` `}`: matrices of parameter estimates (those concentrated out of the likelihood function, see JN (2011)). Note that `GammaHat` is reported as the p x kp matrix [`GammaHat1`, . . ., `GammaHatk`].

Dependencies

Calls the functions `FCVARlike` and `GetParams` for estimation, the functions `FCVARhess`, `SEmat2vec`, and `SEvec2mat` for calculation of standard errors, and the functions `DegreesOfFreedom` and `CharPolyRoots` for display of the estimation results.

5.3 Obtain residuals: `GetResiduals.m`

The function `GetResiduals(x, k, r, db, alpha, beta, rho, Gamma, EstnOptions)` in this file obtains the residuals from the fractionally cointegrated VAR model.

Inputs

- `x`: vector or matrix of data.
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `db`: scalar or vector with parameter value(s) for d and b, at which to obtain residuals. If a scalar is given, then the restriction $d = b$ is imposed.
- `alpha, beta, rho, Gamma`: matrices with parameter values for α, β, ρ, and Γ, at which to obtain residuals.
- `EstnOptions = {N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, Abar, Aperp, replaceCols, keepCols, restrictFminOptions}`: cell array with options, see sections 4 and 6.1 for details.

Output

`epsilon`: matrix of residuals from fractionally cointegrated VAR model evaluated at the parameter values in the inputs.

Dependencies

Calls the function `TransformData`, see section 6.

6 Auxilliary files

These are files that are used by the main programs, but which do not require any action or modification by the user. They are stored in the folder `FCVARtools` and added to the path of functions available to Matlab using a command in the function `DefaultEstnOptions.m`.

LEE: We should move these files to a subfolder and make appropriate changes to the files. MORTEN: Maybe. Let me think about this. LEE: If it is not too hard, then it would really look much neater to have only the main files in the topmost folder together with the manual and the data files, and then hide away all the aux. files that the user doesn't have to work with in a subfolder. MORTEN: Done. Good idea.

6.1 `AdjustEstnOptions.m`

The function `AdjustEstnOptions(db0, k, r, p, initialFCVARoptions)` completes the specification of the options array, initialized by `DefaultEstnOptions(db0, k, r, p)` and `RestrictEstnOptions(db0, k, r, p, restrictFCVARoptions)`.

Inputs

- `db0`: vector or scalar of starting values (in the numerical optimization) for the fractional differencing parameters, d and b. The size of `db0` indicates whether or not restriction $d = b$ is imposed (if `db0` is scalar the restriction is imposed), see section 4.
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `p`: scalar denoting the number of variables in the system.
- `initialFCVARoptions = { dbEstnOptions0, EstnOptions0 }`: These are the arrays `dbEstnOptions0 = { constrained, C_db, c_db, R_db, r_db, dbFminOptions }` and `EstnOptions = { N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, [], [], [], [], restrictFminOptions }` from the function `DefaultEstnOptions`, see section 4.

Output

The output is the cell array `FCVARoptions = { dbEstnOptions, EstnOptions }`, an array of 2 arrays of options with the same form as `initialFCVARoptions`. Both arrays are transformations of the corresponding arrays in the input array `initialFCVARoptions`.

`dbEstnOptions = { constrained, C_db, c_db, R_db, r_db, dbFminOptions }`: array of options for the estimation of the fractional differencing parameters d and b.

- `C_db` and `c_db` are optional matrices for imposing restrictions of the form `C_db`$*(d,b)' \leq$ `c_db`. Note that `C_db` and `c_db` are automatically generated to impose $d \geq b \geq$ `dbMin` given the appropriate indicator `constrained`. However, the user can modify this to impose other or further restrictions on d and/or b within `RestrictEstnOptions.m`.

  LEE: Is that correct now? MORTEN: Yes.

- The rest are as defined in section 4.

`EstnOptions = { N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, Abar, Aperp, replaceCols, keepCols, restrictFminOptions }`: array of options for estimating the remaining parameters.

- `A, H, a, b and h`: defined as in section 4 but `A` and `H` are replaced with identities where no restrictions are specified.

- `Abar`: an array of matrices `Abar_i` such that `Abar_i' * A_i` = I, except in one case. When α is partitioned into 2 submatrices, the first of which is known to be a_1 (with no other restrictions), `Abar{2}` is set to equal $I_p - a_1[a_1'a_1]^{-1}a_1'$ with $A_2 = a_1[a_1'a_1]^{-1}$. This changes the eigenvalue problem for the switching algorithm in `GetParamsSwitching` to match that for this restriction considered in Johansen (1996). With this modification, the algorithm requires only one iteration.

  LEE: Please clarify the special cases. MORTEN: Later (but soon - tomorrow? - save this for the next version), ok? It is to estimate under the one-at-a-time type of restrictions covered in Johansen (96). LEE: Please add in next version. MORTEN: Done.

- `Aperp`: an array of matrices `Aperp_i` such that `Aperp_i' * A_i` $= 0$, except in one case. When β is partitioned into 2 submatrices, the first of which is known to be b_1 (with no other restrictions), `Aperp` is set to b_1 to force the optimization to search for β_2 in the space orthogonal to b_1. This requires only one iteration.

  LEE: Again, please clarify the special cases.

- `replaceCols and keepCols`: arrays of vectors specifying the column numbers that are replaced or assumed known in each stage of the switching algorithm.

- The rest are as defined in section 4.

Dependencies

None.

6.2 `CharPolyDet.m`

MORTEN: This function is no longer needed since the new `CharPolyRoots` (formerly `UnitCircleImage`) is now self contained. See next function.

The function `CharPolyDet(uVec, alpha, beta, Gamma)` calculates the determinant of the characteristic polynomial for the usual cointegration model ($d = b = 1, L_b = L$).

Inputs

- `uVec`: vector `uVec = [ uR uI ]` provides the real and complex parts of the candidate root of the characteristic polynomial.

- `alpha`, `beta`, `Gamma`: matrices of parameters. Note: The matrix Gamma is a $p \times pk$ matrix of coefficients in the lag polynomial.

Output

`charPolyDet` returns the determinant of the characteristic polynomial calculated at location `u = uR + uI*`i.

Dependencies

None.

6.3 `CharPolyRoots.m`

The function `CharPolyRoots(b, alpha, beta, Gamma, print2screen)` calculates the roots of the characteristic polynomial for the fractionally cointegrated VAR model and plots them with the transformed unit circle, see Johansen (2008).

Inputs

- `b`: scalar fractional differencing parameter.
- `alpha`, `beta`, `Gamma`: matrices of parameters. Note: The matrix Gamma is a $p \times pk$ matrix of coefficients in the lag polynomial.
- `print2screen`: scalar indicator to determine if a graph of the roots will be produced (to complement other statistics printed to screen).

Output

`charPolyRoots`: complex vector with the roots of the characteristic polynomial.

Dependencies

None.

6.4 `DegreesOfFreedom.m`

The function `DegreesOfFreedom(db0, k, r, p, EstnOptions)` calculates the degrees of freedom for the fractionally cointegrated VAR model.

When no restrictions are specified on α or β, the degrees of freedom are adjusted to restrict the top rows of beta to equal the identity matrix. For restricted estimation, it is assumed that the user imposes enough restrictions such that the parameters are identified.

Whenever there are restrictions on both *alpha* and β, a warning is given to indicate that the output `df` does not include the number of degrees of freedom in Π. This happens when there are restrictions placed on both `alpha` and `beta` together, in which case it is possible that some restrictions are redundant.

LEE: This could be clarified a bit. Perhaps it should also be part of the output? MORTEN: Is this better?

Inputs

- `db0`: scalar or vector of fractional differencing parameters d and b ($d = b$ is imposed if `db0` is scalar).
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `p`: scalar denoting the number of variables in the system.
- `EstnOptions = {` `N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, [], [], [], [], [], [], [], print2screen, printGammas, restrictFminO` `}`: cell array of options described in detail in the comments to the functions `DefaultEstnOptions` in section 4 and `AdjustEstnOptions` in section 6.1.

Output

- `df`: scalar with the degrees of freedom for the fractionally cointegrated VAR model.
- `dfWarning`: binary variable taking the value 1 if the value of `df` does not include the degrees of freedom in Π implied by the parameters `alpha` and `beta`.

Dependencies

None.

6.5 FCVARhess.m

The funcion `FCVARhess(x, k, r, db, alpha, beta, rho, Gamma, EstnOptions)` is the Hessian matrix for the fractionally cointegrated VAR model evaluated at `[ db, alpha, beta, rho, Gamma ]`.

Inputs

- `x`: vector or matrix of data.
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `db`: scalar or vector with parameter value(s) for d and b, at which to calculate the log-likelihood. If a scalar is given, then the restriction $d = b$ is imposed.
- `alpha, beta, rho, Gamma`: matrices of parameter values at which to calculate the log-likelihood.
- `EstnOptions` = { `N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, [], [], [], [], restrictFminOptions` }: cell array of options described in detail in the comments to the functions `DefaultEstnOptions` and `RestrictEstnOptions` in section 4 and `AdjustEstnOptions` in section 6.1.

Output

`hessian`: the Hessian matrix for the fractionally cointegrated VAR model evaluated at the parameter values in the inputs.

Dependencies

Calls the function `FullFCVARlike` to get the likelihood and also calls the functions `SEmat2vec` and `SEvec2mat` to convert parameter vector to matrices and vice versa.

6.6 `FCVARlike.m`

The function `FCVARlike(x, k, r, db, EstnOptions)` in this file is the log-likelihood function for the fractionally cointegrated VAR model.

Inputs

- `x`: vector or matrix of data.
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `db`: vector or scalar of fractional differencing parameters, d and b. The size of `db` indicates whether or not restriction $d = b$ is imposed (if `db` is scalar the restriction is imposed).
- `EstnOptions = { N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, Abar, Aperp, replaceCols, keepCols, restrictFminOptions }`: an array of options for estimating the remaining parameters. The elements of `EstnOptions` are described in detail in the comments to the functions `DefaultEstnOptions` and `RestrictEstnOptions` in section 4 and `AdjustEstnOptions` in section 6.1.

Output

`like`: scalar value of the profile (concentrated) log-likelihood function evaluated at parameter value `db`.

Dependencies

Calls the function `GetParams`.

6.7 `FracDiff.m`

The function `FracDiff(x,d,M)` calculates the fractional difference of `x`.

Inputs

- `x`: vector or matrix of data.
- `d`: scalar denoting the value of d at which to calculate the fractional difference.
- `M`: scalar denoting the truncation in the fractional differencing operator, see section 4 for details.

Output

`dx`: vector or matrix $(1-L)^d x$.

Dependencies

None.

6.8 `FullFCVARlike.m`

The function `FullFCVARlike(x, k, r, db, alpha, beta, rho, Gamma, EstnOptions)` in this file is the unconcentrated log-likelihood function for the fractionally cointegrated VAR model.

Inputs

- `x`: vector or matrix of data.
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `db`: scalar or vector with parameter value(s) for d and b, at which to calculate the log-likelihood. If a scalar is given, then the restriction $d = b$ is imposed.
- `alpha, beta, rho, Gamma`: matrices of parameter values at which to calculate the log-likelihood.
- `EstnOptions` = { `N, M, deterministics, R_Gamma, r_Gamma, [], [], [], nCols, A, H, a, b, h, Abar, Aperp, replaceCols, keepCols, restrictFminOptions` }: an array of options for estimating the remaining parameters. The elements of `EstnOptions` are described in detail in the comments to the functions `DefaultEstnOptions` and `RestrictEstnOptions` in section 4 and `AdjustEstnOptions` in section 6.1.

Output

`like`: scalar value of the log-likelihood function evaluated at the parameter values in the inputs.

Dependencies

Calls the function `GetResiduals`

6.9 `GetParams.m`

The function `GetParams(x, k, r, db, EstnOptions)` returns the parameter estimates of the fractionally cointegrated VAR model using regression and reduced rank regression.

Inputs

The inputs `x`, `k`, `r`, `db`, and `EstnOptions` are as defined for the function `FCVARlike`.

Output

`estimates = {` `dbHat, alphaHat, betaHat, rhoHat, PiHat, OmegaHat, GammaHat` `}`: matrices of parameter estimates (those concentrated out of the likelihood function, see JN (2011)) appended to the input `db`. Note that `GammaHat` is reported as the $p \times kp$ matrix `[ GammaHat1,..., GammaHatk ]`.

Dependencies

Calls the function `TransformData` and, if any restrictions are imposed on the estimation, also the function `GetParamsSwitching`.

6.10 `GetParamsSwitching.m`

The function `GetParamsSwitching(beta0, S00, S01, S11, T, p, r, EstnOptions)` returns the parameter estimates of the fractionally cointegrated VAR model using regression and reduced rank regression, by way of Johansen's switching algorithm, see Johansen (1996, chapters 7 and 9).

Inputs

- `beta0`: matrix of starting values for β in the switching algorithm. The unrestricted estimate of β is used and is then reordered within this function as the matrix closest to satisfying any linear restrictions on β as in Johansen (1996).
- `S00, S01 and S11`: matrices of product moments, see Johansen (1996), with obvious changes for the fractional case.
- `T`: sample size.
- `p`: scalar denoting the number of variables in the system.
- `r`: scalar denoting cointegrating rank.
- `EstnOptions`: array of options described in detail in the comments to the functions `DefaultEstnOptions` in section 4 and `AdjustEstnOptions` in section 6.1.

Output

`betaHat, alphaHat, rhoHat, and OmegaHat`: matrices of parameter estimates.

LEE: Should this instead be `estimates =` { `dbHat, alphaHat, betaHat, rhoHat, PiHat, OmegaHat, GammaHat` } as in `GetParams`? MORTEN: No. This function is used within GetParams only. It uses inputs partially processed from GetParams and outputs exactly what is skipped by not using the usual algorithm. PiHat is calculated immediately after and GammaHat has already been concentrated out.

Dependencies

None.

6.11 `Lbk.m`

The function `Lbk(x, b, k, M)` in this file calculates a lag polynomial in the fractional lag operator, L_b.

Inputs

- `x`: vector or matrix of data.
- `b`: scalar denoting the value of b at which to calculate the lag polynomial.
- `k`: scalar denoting the number of lags.
- `M`: scalar denoting the truncation in the fractional differencing operator, see section 4 for details.

Output

`Lbkx`: matrix $[\ L_b^1 X, L_b^2 X, \ldots, L_b^k X\ ]$, where $L_b = 1 - (1-L)^b$.

Dependencies

Calls the function `FracDiff`.

6.12 `SEmat2vec.m`

The function `SEmat2vec( db, alpha, Gamma, k, r, p )` transforms the vector `db` and the matrices `alpha` and `Gamma` into either a $(1 + pr + p^2k)$ or a $(2 + pr + p^2k)$ parameter vector, depending on the size of `db`. Note that `SEmat2vec( db, alpha, Gamma, k, r, p )` and `SEvec2mat( param, k, r, p )` are inverse functions of each other.

Inputs

- `db`: scalar or vector of d and b.
- `alpha`: matrix containing α.
- `Gamma`: matrix containing Γ.
- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `p`: scalar denoting the number of variables in the system.

Output

`param`: either a $(1 + pr + p^2k)$ or a $(2 + pr + p^2k)$ parameter vector, depending on the size of `db`.

LEE: Does that mean that it always outputs both a d and a b, even if they are restricted to d=b? MORTEN: No. It preserves the size of db. Note changes above.

Dependencies

None.

6.13 `SEvec2mat.m`

The function `SEvec2mat( param, k, r, p )` transforms either a $(2 + pr + p^2k)$ or a $(2 + pr + p^2k)$ parameter vector into the vector `db` and the matrices `alpha` and `Gamma`. Note that `SEmat2vec( db, alpha, Gamma, k, r, p )` and `SEvec2mat( param, k, r, p )` are inverse functions of each other.

Inputs

- `param`: either a $(1 + pr + p^2k)$ or a $(2 + pr + p^2k)$ parameter vector, depending on the size of `db`.

  LEE: How does the program know if there is only one d=b value or if there are two fractional parameters? By counting the total number of parameters it would think that there are two. MORTEN: It knows k, r and p and chops off those elements as it is constructing the other matrices. What is left is the vector or scalar db.

- `k`: scalar denoting the number of lags.
- `r`: scalar denoting cointegrating rank.
- `p`: scalar denoting the number of variables in the system.

Output

- `db`: scalar or vector of d and b.
- `alpha`: matrix containing α.
- `Gamma`: matrix containing Γ.

Dependencies

None.

6.14 `TransformData.m`

The function `TransformData(x, k, db, EstnOptions)` returns the transformed data required for regression and reduced rank regression.

Inputs

- `x`: vector or matrix of data.
- `k`: scalar denoting the number of lags.
- `db`: scalar of vector of fractional differencing parameters d and b. If this is a scalar, then the restriction $d = b$ is imposed.
- `EstnOptions`: array of options described in detail in the comments to the functions `DefaultEstnOptions` in section 4 and `AdjustEstnOptions` in section 6.1.

Output

`Z0, Z1, Z2`: matrices of transformed data, see Johansen (1996), with obvious changes for the fractional case.

Dependencies

Calls the functions `FracDiff` and `Lbk`.

6.15 `UnitCircleImage.m`

MORTEN: Note that this function is no longer necessary. It has been replaced with `CharPolyRoots`.

The function `UnitCircleImage(b, alpha, beta, Gamma, print2screen, uGrid, charPolyDegree` calculates the roots of the characteristic polynomial for the fractionally cointegrated VAR model and plots them with the transformed unit circle, see Johansen (2008).

Inputs

- `b`: scalar fractional differencing parameter.
- `alpha`, `beta`, `Gamma`: matrices of parameters. Note: The matrix Gamma is a $p \times pk$ matrix of coefficients in the lag polynomial.
- `print2screen`: scalar indicator to determine if a graph of the roots will be produced (to complement other statistics printed to screen).
- `uGrid`: is a grid over which to sample values of the characteristic polynomial. This will be used to estimate the coefficients of the characteristic polynomial.
- uGrid is generated as uGrid = (- uGridMax : uGridStep : uGridMax), where uGridMax and uGridStep are specified in DefaultEstnOptions.

  LEE: I don't understand this. I also don't see those options in DefaultEstnOptions.
- charPolyDegree is estimate of an upper bound on the degree of the characteristic polynomial.

  LEE: I don't understand this either. The degree of the polynomial can be explicitly calculated given k and p.

Output

`charPolyRoots`: complex vector with the roots of the characteristic polynomial.

Dependencies

Calls the function `CharPolyDet`.

Other comments

Note: This function calculates roots by generating a sample of determinants as data points and the coefficients of the characteristic polynomial are estimated by OLS regression. The problem of finding roots of a polynomial with given coefficients is easily calculated.

LEE: This I dont understand. Please explain? Why is this not just a problem of locating roots of a polynomial? What is this business with OLS?

MORTEN: Don't worry about this anymore. It's out.

7 List of references

1. Johansen, S. (1996). *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*, 2nd edition, Oxford University Press, Oxford.

2. Johansen, S. (2008). A representation theory for a class of vector autoregressive models for fractional processes. *Econometric Theory* **24**, 651–676.

3. Johansen, S. and Nielsen, M. Ø. (2010). Likelihood inference for a nonstationary fractional autoregressive model. *Journal of Econometrics* **158**, 51–66.

4. Johansen, S. and Nielsen, M. Ø. (2011). Likelihood inference for a fractionally cointegrated vector autoregressive model. QED working paper 1237, Queen's University.

5. MacKinnon, J. G. and Nielsen, M. Ø. (2011). Numerical distribution functions of fractional unit root and cointegration tests. QED working paper 1240, Queen's University.