4. EXAMPLES

The examples use the following data bases:

- *frfdem.asc* — Ascii file which contains daily quotations of the FRF/DEM exchange rate since 1987.
- *gnp.asc* — table B (page 515) of Harvey [1990]. The ascii file consists of US Real Gross National Product (GNP) (annual data for the period 1910-1970).
- *lutkepohl.asc* — table E.1 (page 498) of Lütkepohl [1991]. The ascii file consists of three series: German Fixed Investment, Disposable Income and Consumption Expenditures (quarterly, seasonally adjusted for the period 1960-1982).
- *lynx.asc* — series G (page 557) of Brockwell and Davis or data (appendix 3, page 470) of Tong [1990] or data (page 322) of Janacek and Swift [1993]. The ascii file consists of annual Canadian lynx trappings for the period 1821-1934.
- *purse*.asc — table C (page 516) of Harvey [1990] or data (page 324) of Janacek and Swift [1993]. The ascii file consists of Purses snatched in the Hyde Park area of Chicago (28-day-period from January 1968).
- *rainfall.asc* — table D (page 517) of Harvey [1990]. The ascii file consists of Rainfall in Fortaleza, North-East Brazil (annual data for the period 1849-1984).
- *reinsel.asc* — table A.2 (page 227) of Reinsel [1993]. The ascii file consists of two series: U.S. Fixed Investment and Changes in Business Inventories (quarterly, seasonally adjusted for the period 1947-1971).
- *sunspots.asc* — data (page 327) of Janacek and Swift [1993]. The ascii file consists of Wolfer sun spot numbers.

1. **arfima1.prg**
   We simulate an ARIMA(1,0,1) process
   $$(1 - 0.95L)\, y_t = (1 - 0.5L)\, \varepsilon_t \tag{4.1}$$
   with $\varepsilon_t \sim \mathcal{N}(0, 2)$. Then, we estimate the following ARFIMA model in the frequency domain
   $$(1 - \phi_1 L)\,(1 - L)^d\, y_t = (1 - \theta_1 L)\, \varepsilon_t \tag{4.2}$$

2. **arfima2.prg**
   We examine the problem of several maxima when a fractional process is estimated in the frequency domain. To do this, we use the simulated process (4.1) and estimate the model (4.2) using two algorithms: the scoring algorithm and the BFGS algorithm of **OPTMUM**.

3. **arfima3.prg**
In certain cases, the problem of several maxima comes from the estimation of the fractional d coefficient. If we impose the restriction $d = 0$, we notice that we get only one maximum. This suggests first using the Geweke-Porter Hudak (GPH) estimator and then fixing the fractional d coefficient to the GPH estimator to estimate completely the ARFIMA process.

4. **arfima4.prg**
We simulate the following ARFIMA process with the procedure `RND_arfima`

$$(1 - 0.8L)(1 - L)^{0.25} y_t = (1 - 0.4L)\varepsilon_t \tag{4.3}$$

with $\varepsilon_t \sim \mathcal{N}(0, 2)$. Then, we estimate the following ARFIMA model in the frequency domain

$$(1 - \phi_1 L)(1 - L)^d y_t = (1 - \theta_1 L)\varepsilon_t \tag{4.4}$$

Firstly, we estimate the unrestricted model. Secondly, we estimate the model under the restriction $d = 0.25$. Thirdly, we impose the restrictions $d = 0.25$ and $\theta_1 = 0.4$. Finally we test the two hypotheses $\mathrm{H}_1 : d = 0.25$ and $\mathrm{H}_2 : (d, \theta_1) = (0.25, 0.4)$ with the likelihood ratio statistic.

5. **arfima5.prg**
Simulation of fractional processes with $d = -0.25$ and $d = 0.75$.

6. **arma1a.prg**
Let $y_{1,t}$ be the variation in investment and $y_{2,t}$ the inventories level. We estimate the following vector ARMA(1,1) model

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix} - \Phi_1 \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \end{bmatrix} = \begin{bmatrix} \varepsilon_{1,t} \\ \varepsilon_{2,t} \end{bmatrix} - \Theta_1 \begin{bmatrix} \varepsilon_{1,t-1} \\ \varepsilon_{2,t-1} \end{bmatrix} \tag{4.5}$$

with $\varepsilon_t \sim \mathcal{N}(\mathbf{0}_2, \Sigma)$. We use the Newton-Raphson algorithm to obtain the estimates $\beta = \operatorname{vec} \begin{bmatrix} \Phi_1 & \Theta_1 \end{bmatrix}$. The external variables `_arma_sigma` and `_arma_epsilon` correspond to the estimate of Σ and to the residuals $\hat{\varepsilon}$ respectively.

7. **arma1b.prg**
We estimate the model (4.5) by exact maximum likelihood. For this, we use the Kalman Filter. To obtain the initial conditions, we use both the estimates of the Conditional Maximum Likelihood and the procedure `SSM_ic`. Given the procedure `KF_ml`, we construct the log-likelihood function. Then, we employ `TD_ml` to obtain the exact ML estimates. Note that the estimates θ correspond to the vector $\operatorname{vec} \begin{bmatrix} \beta & \mathbb{P}^\star \end{bmatrix}$ with $\mathbb{P}^\star = \operatorname{vech}(\mathbb{P})$ and $\mathbb{P}$ the Cholesky decomposition of Σ, that is $\Sigma = \mathbb{P}\mathbb{P}^\top$.

8. **arma1c.prg**
The model (4.5) is estimated by conditional maximum likelihood with linear restrictions of the form $\beta = R\gamma + r$. We impose $\beta_1 = 1$ (that is $\Phi_{1,11} = 1$).

We have

$$\begin{bmatrix}\beta_1\\\beta_2\\\beta_3\\\beta_4\\\beta_5\\\beta_6\\\beta_7\\\beta_8\end{bmatrix} = \begin{bmatrix}\mathbf{0}_{1\times 7}\\\mathbf{I}_7\end{bmatrix}\begin{bmatrix}\gamma_1\\\gamma_2\\\gamma_3\\\gamma_4\\\gamma_5\\\gamma_6\\\gamma_7\end{bmatrix} + \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}$$

To construct the matrix R, we employ the `design` procedure. Because the argument `sv` in `arma_CML` is 0, the procedure computes the starting values for the optimization algorithm.

9. **arma1d.prg**
   Estimates the model (4.5) by conditional maximum likelihood under the restriction $\Phi_{1,11} = \Phi_{1,21}$ (or $\beta_1 = \beta_2$). We put this linear restriction into the form

$$\begin{bmatrix}\beta_1\\\beta_2\\\beta_3\\\beta_4\\\beta_5\\\beta_6\\\beta_7\\\beta_8\end{bmatrix} = \begin{bmatrix}1 & \mathbf{0}_{1\times 6}\\1 & \mathbf{0}_{1\times 6}\\\mathbf{0}_{6\times 1} & \mathbf{I}_6\end{bmatrix}\begin{bmatrix}\gamma_1\\\gamma_2\\\gamma_3\\\gamma_4\\\gamma_5\\\gamma_6\\\gamma_7\end{bmatrix} + \mathbf{0}_{8\times 1}$$

10. **arma1e.prg**
   Estimates the model (4.5) by conditional maximum likelihood with the restrictions $\Phi_{1,12} = \Theta_{1,11} = \Theta_{1,21} = 0$ (or $\beta_3 = \beta_5 = \beta_6 = 0$). These restrictions are motivated because these coefficients are not significantly different from zero. We have

$$\begin{bmatrix}\beta_1\\\beta_2\\\beta_3\\\beta_4\\\beta_5\\\beta_6\\\beta_7\\\beta_8\end{bmatrix} = \begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&0&0&0\\0&0&1&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}\begin{bmatrix}\gamma_1\\\gamma_2\\\gamma_3\\\gamma_4\\\gamma_5\end{bmatrix} + \mathbf{0}_{8\times 1}$$

11. **arma1f.prg**
   In this example, we impose the restriction that the model (4.5) corresponds to two univariate ARMA(1,1) processes. That is, the matrices Φ_1 and Θ_1 are of the form

$$\begin{bmatrix}\cdot & 0\\0 & \cdot\end{bmatrix}$$

12. **arma1g.prg**
   With the command `vread`, we read from the external variables

`_ml_derivatives` the Jacobian and gradient vectors and the Hessian and information matrices of the log-likelihood function evaluated at the estimates $\hat{\beta}$ corresponding to the ML estimates under the preceding restrictions (*arma1f.prg*).

13. **arma1h.prg**
    We test whether the restrictions in the program *arma1f.prg* are accepted. To this end, we use the likelihood ratio, the Lagrange multiplier or the Wald test. Note that the Lagrange multiplier is evaluated by using the vectors and matrices given by `_ml_derivatives`.

14. **arma1i.prg**
    Stability analysis of the model (4.5).

15. **arma1j.prg**
    Forecast Error Variance Decomposition of the model (4.5).

16. **arma1k.prg**
    Impulse Responses of the model (4.5).

17. **arma2a.prg**
    We consider the univariate AR(1) model

$$y_t = 0.5 y_{t-1} + \varepsilon_t$$

In its state space form, the vector of state variables is $\begin{bmatrix} y_t & \varepsilon_t \end{bmatrix}^\top$. The covariance matrix corresponds to

$$\begin{bmatrix} E\left[y_t y_t\right] & E\left[y_t \varepsilon_t\right] \\ E\left[\varepsilon_t y_t\right] & E\left[\varepsilon_t \varepsilon_t\right] \end{bmatrix}$$

Computing this covariance can be achieved with the **SSM_ic** procedure.

18. **arma2b.prg**
    Same program as *arma2a.prg* but with a univariate MA(1) model.

19. **arma2c.prg**
    Same program as *arma2a.prg* but with the vector model (4.5).

20. **arma2d.prg**
    Exact maximum likelihood estimation of a univariate ARMA(1,1) model by Kalman filter (Kohn and Ansley [1983]). The results are compared with those obtained from Ansley's [1979] algorithm (**arima** library).

21. **arma2e.prg**
    Exact maximum likelihood estimation of the vector ARMA(1,1) model (4.5) by Kalman filter (Kohn and Ansley [1983]). The difference with the *arma1b.prg* program is that the initial conditions are computed at each iteration (`SSM_ic` is included in the `ml` procedure). *arma2e.prg* computes the Exact MLE (*arma1b.prg* computes an approximation of the Exact MLE).

22. **autocov1.prg**
Computes the theoretical autocovariances and autocorrelations of the following VAR(1) process

$$Y_t - \begin{bmatrix} .5 & 0 & 0 \\ .1 & .1 & .3 \\ 0 & .2 & .3 \end{bmatrix} Y_{t-1} = \varepsilon_t \tag{4.6}$$

with $\varepsilon_t \sim \mathcal{N}(\mathbf{0}_3, \Sigma)$ and

$$\Sigma = \begin{bmatrix} 2.25 & 0 & 0 \\ 0 & 1 & .5 \\ 0 & .5 & .74 \end{bmatrix}$$

To read the matrices, we use the `varget` procedure.

23. **autocov2.prg**
Computes the theoretical autocovariances and autocorrelations of the following VAR(2) process

$$Y_t - \begin{bmatrix} .5 & .1 \\ .4 & .5 \end{bmatrix} Y_{t-1} - \begin{bmatrix} 0 & 0 \\ .25 & 0 \end{bmatrix} Y_{t-2} = \varepsilon_t \tag{4.7}$$

with $\varepsilon_t \sim \mathcal{N}(\mathbf{0}_2, \Sigma)$ and

$$\Sigma = \begin{bmatrix} .09 & 0 \\ 0 & .04 \end{bmatrix}$$

24. **autocov3.prg**
Computes the theoretical autocovariances and autocorrelations of the vector ARMA model (4.5).

25. **autocov4.prg**
Same program as *autocov2.prg*, but autocovariances are computed with the `SSM_autocov` procedure.

26. **autocov5.prg**
Same program as *autocov3.prg*, but autocovariances are computed with the `SSM_autocov` procedure.

27. **autocov6.prg**
Computes autocovariances and autocorrelations matrices of a time-invariant state space model.

28. **band1a-1d.prg**
Ad hoc examples to show the use of the subband wavelet procedures: `split`, `extract`, `select` and `insert`.

29. **basis1.prg**
We use the procedure `isBasis` to verify that we have a wavelet packet basis.

30. **basis2.prg**
We use the procedure `BasisPlot` to obtain the Time-frequency plane tilings plot of several bases. We can also see the localization in time and in frequency. For the basis *base0*, we have a good localization in time, but not in frequency. For the basis *base9*, this is the opposite.

31. **basis3.prg**
We use the `BestLevel` procedure to select a basis with the log-energy cost function. Then, we verify that the selected basis has effectively the minimal cost value.

32. **basis4.prg**
Same program as *basis3.prg* but with the BestBasis procedure and different cost functions (entropy, ℓ^p norm and log-energy).

33. **boot1-3.prg**
Illustration of the `bootstrap_SMM` procedure.

34. **bsm1.prg**
We study the Basic Structural Model presented by Harvey [1990]. The measurement equation is

$$y_t = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \eta_t \\ \zeta_t \\ \omega_t \\ \omega_{t-1} \\ \omega_{t-2} \end{bmatrix} + \varepsilon_t$$

with $\varepsilon_t \sim \mathcal{N}(0, H)$ and the transition equation is

$$\begin{bmatrix} \eta_t \\ \zeta_t \\ \omega_t \\ \omega_{t-1} \\ \omega_{t-2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \eta_{t-1} \\ \zeta_{t-1} \\ \omega_{t-1} \\ \omega_{t-2} \\ \omega_{t-3} \end{bmatrix} + \begin{bmatrix} \mathbf{I}_3 \\ \mathbf{0}_{2\times 3} \end{bmatrix} \nu_t$$

with $\nu_t \sim \mathcal{N}(0, Q)$. We have

$$H = \sigma_\varepsilon^2$$

and

$$Q = \begin{bmatrix} \sigma_\eta^2 & 0 & 0 \\ 0 & \sigma_\zeta^2 & 0 \\ 0 & 0 & \sigma_\omega^2 \end{bmatrix}$$

Using the numerical values $\left(\sigma_\varepsilon^2, \sigma_\eta^2, \sigma_\zeta^2, \sigma_\omega^2\right) = (1, 0.25, 1.25, 3)$, we simulate the BSM model with the initial position $\begin{bmatrix} 100 & 4 & 4 & 2 & 3 \end{bmatrix}^\top$. Then we estimate the BSM model with ML in the frequency domain. In our case, we set s equal to 4.

35. **bsm2.prg**
We perform Monte Carlo experiments to investigate the power of the FDML of the BSM model.

36. **bsm3.prg**
In this example, we estimate the basic structural model in the frequency and in the time domains. Because the model is not stable, we cannot use the procedure `SSM_ic`. There are several ways to initialize the Kalman filter. The Kalman filter can be used to obtain unobservable components, for example the seasonal factor.

37. **canon1.prg**
Computes the moving average and autoregressive representations of the VAR(1) process described in equation (4.6).

38. **canon2.prg**
Computes the moving average and autoregressive representations of the VAR(2) process described in equation (4.7).

39. **canon3.prg**
Computes the infinite moving average and autoregressive representations of the vector ARMA process (4.5).

40. **canon4.prg**
For a univariate ARMA(p,q) model, we can use the `canonical_arfima` or `canonical_arma` procedures. The model is

$$y_t - 0.5y_{t-1} - 0.25y_{t-2} = u_t - 0.4u_{t-1} + 0.3u_{t-2} \tag{4.8}$$

41. **canon5.prg**
Computes the impulse responses and the accumulated impulse responses (or the interim multipliers) of the ARMA model (4.8).

42. **canon6.prg**
Computes the impulse responses and the accumulated impulse responses (or the interim multipliers) of the ARFIMA process

$$\left(1 - 0.5L + 0.25L^2\right)(1 - L)^d\, y_t = (1 - 0.3L)\, u_t \tag{4.9}$$

The fractional operator d takes different values: -0.5, -0.25, 0, 0.25, and 0.5.

43. **canon7.prg**
Computes the autocovariances, autocorrelations and partial autocorrelations of the ARFIMA process (4.9). The `AUTOCOV` procedure uses the fact that if the process allows an infinite moving average representation

$$y_t = \sum_{i=0}^{\infty} \theta_i u_{t-i}$$

then the autocovariances γ_i of the process (if we assume that $\operatorname{var}(u_t) = 1$) are equal to

$$\gamma_i = \sum_{j=0}^{\infty} \theta_j \theta_{j+i}$$

The autocorrelations correspond to

$$\rho_i = \frac{\gamma_i}{\gamma_0}$$

and the partial autocorrelations are obtained as the solution of the Toeplitz system

$$\begin{bmatrix} \gamma_0 & \gamma_1 & \gamma_2 & \cdots & \gamma_{i-1} \\ \gamma_{i-1} & \gamma_0 & \gamma_1 & \cdots & \gamma_{i-2} \\ & & & \ddots & \\ \gamma_{i-1} & \gamma_{i-2} & \gamma_{i-3} & \cdots & \gamma_0 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_i \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_i \end{bmatrix}$$

44. **chirp1a.prg**
We define the linear chirp $x_t = \sin\left(100\pi t^2\right)$. Using the Coiflet #2 filters, we compute the difference between the original signal and the reconstructed signal by the inverse wavelet transform.

45. **chirp1b.prg**
We use the precedent linear chirp. We plot the wavelet packet table of the signal. Using the basis $\mathcal{B} = (1, 2, 3, 3)$, we show that the reconstructed signal by applying the inverse wavelet packet transform is the same as the original signal.

46. **cml1-5.prg**
Illustration of the use of the **CML** package with **TSM**.

47. **cpdgm1.prg**
Illustration of the `CPDGM` procedure.

48. **cspect1-2.prg**
Illustration of the `CSpectrum` procedure.

49. **cspect3.prg**
Example 11.7.1 in Brockwell and Davis [1991].

50. **cusum1.prg**
Estimates the Local Level model (or random walk plus noise) with the data *purse*. The model corresponds to

$$\left\{ \begin{array}{lcl} y_t & = & \mu_t + \varepsilon_t \\ \mu_t & = & \mu_{t-1} + \eta_t \end{array} \right.$$

Using the Kalman filter, we can construct the standardized innovations

$$w_t = \frac{v_t}{\sqrt{f_t}}$$

Then we build the CUSUM statistic

$$W_t = \frac{1}{s}\sum_{i=1}^{t} w_i$$

with s the standard deviation of the standardized innovations w_t and the CUSUMsq statistic

$$W_t^{\bullet} = \frac{\sum_{i=1}^{t} w_i^2}{\sum_{i=1}^{T} w_i^2}$$

51. **cusum2.prg**
Computes the CUSUM and CUSUMsq statistics. The model is a MA(2) process and is estimated by exact maximum likelihood using the Kalman filter.

52. **cusum3.prg**
This is the same thing as the *cusum2.prg* example, except that a leverage point is introduced in the MA(2) process.

53. **cycle.src**
Spectral generating functions for the trend + cycle model and the cyclical trend model.

54. **cycle1.prg**
Represents the power spectra for stochastic cycles.

55. **cycle2.prg**
Harvey [1989] uses a stochastic cycle plus noise model to explain the *rainfall* data. The spectral generating function for a stochastic cycle plus noise model is the sum of the s.g.f. of the stochastic cycle and the s.g.f. of the noise. The s.g.f. of the stochastic cycle is given by the `_cycle_sgf` procedure. The model is estimated in the frequency domain with the `FD_ml` procedure. We observe that we obtain the same results as in Harvey [1989].

56. **cycle3.prg**
In this example, we compare the periodogram of the *Rainfall* data with the estimated spectral generating function.

57. **denois1a-1d.prg**
We consider the generated series

$$x_t = \sin(t) + \sin(2t) + u_t$$

with u_t a white noise process. Denoising a series could be done by using the wavelet shrinkage. In a first step, we calculate the wavelet coefficients with the `wt` procedure. In a second step, we use a thresholding technique. Finally, we reconstruct the series by applying the `iwt` procedure to the thresholding coefficients.

58. **denois2a-2b.prg**
In the examples below, the wavelet shrinkage is applied to all the coefficients. But, we can use thresholding techniques just for some coefficients, for example the coefficients of some subband of the wavelet transform or of the wavelet packet transform.

59. **fdml1a.prg**
We simulate an AR(2) process. Then, we use the Bloomfield exponential spectral density. The corresponding spectral generating function is given in Dzhaparidze [1986] on page 125:

$$g(\lambda_j) = \sigma^2 \exp\left(2\sum_{i=1}^{r} \gamma_i \cos(i\lambda_j)\right)$$

We may estimate the vector of parameters $\theta = \begin{bmatrix} \gamma_1 & \cdots & \gamma_r & \sigma \end{bmatrix}^\top$ with the `FD_ml` procedure. In this example, we have set r equal to 4.

60. **fdml1b.prg**
We test now $r = 4$ against $r = 5$. To compute the spectral LM test, we can employ the data buffer `_ml_derivatives` or the procedure `FDml_derivatives`.

61. **fdml2a.prg**
We consider the model z_t, defined by

$$\left\{ \begin{array}{lcl} z_t & = & x_t + y_t \\ x_t & = & \phi_1 x_{t-1} + u_t \\ y_t & = & v_t - \theta_1 v_{t-1} \end{array} \right.$$

with $u_t \sim N\left(0, \sigma_u^2\right)$ and $v_t \sim N\left(0, \sigma_v^2\right)$. The corresponding spectral generating function is

$$\begin{array}{rcl} g\left(\lambda_j\right) & = & \sigma_u^2 \frac{1}{\left|1-\phi_1 e^{i\lambda_j}\right|^2} + \sigma_v^2 \left|1 - \theta_1 e^{i\lambda_j}\right|^2 \\ & = & \sigma_u^2 \frac{1}{\left(1-2\phi_1 \cos\lambda_j + \phi_1^2\right)} + \sigma_v^2 \left(1 - 2\theta_1 \cos\lambda_j + \theta_1^2\right) \end{array}$$

The vector of parameters is set to $\begin{bmatrix} \phi_1 & \sigma_u & \theta_1 & \sigma_v \end{bmatrix}^\top$.

62. **fdml2b.prg**
To see if $\theta_1 = 0.7$ in the above model, we use the LM and LR tests in the frequency domain.

63. **fdml3.prg**
The model is

$$\left\{ \begin{array}{lcl} z_t & = & x_t + y_t \\ x_t & = & x_{t-1} + u_t - \theta_1 u_{t-1} \\ y_t & = & v_t \end{array} \right.$$

with $u_t \sim N\left(0, \sigma_u^2\right)$ and $v_t \sim N\left(0, \sigma_v^2\right)$. The stationary form is

$$z_t - z_{t-1} = \left(1 - \theta_1 L\right) u_t + \left(1 - L\right) v_t$$

The spectral generating function *for the stationary form* is

$$\begin{array}{rcl} g\left(\lambda_j\right) & = & \sigma_u^2 \left|1 - \theta_1 e^{i\lambda_j}\right|^2 + \sigma_v^2 \left|1 - e^{i\lambda_j}\right|^2 \\ & = & \left(1 - 2\theta_1 \cos\lambda_j + \theta_1^2\right) \sigma_u^2 + 2\left(1 - \cos\lambda_j\right) \sigma_v^2 \end{array}$$

Because the stationary form is $z_t - z_{t-1}$, the data used in the `FD_ml` procedure are `z-lag1(z)`.

64. **fdml4.prg**
Same example as *kalman4c.prg*, but the spectral generating function is computed by the `sgf_SSM` procedure.

65. **fdml5-6.prg**
Examples of Maximum Likelihood of multivariate processes in the frequency domain.

66. **fft.prg**
An example to illustrate the problem of the scaled factor. For any time series x_t, we must verify that the Fourier transform for the first frequency $\lambda_0 = 0$ equals the mean of x_t:
$$f(\lambda_0) = \bar{x}$$

67. **filter1a.prg**
Univariate ARMA process estimation with the `arma_Filter` procedure.

68. **filter1b-1c.prg**
Univariate ARMA-GARCH process estimation with the `arma_Filter` and `garch_Filter` procedures.

69. **filter2a.prg**
Estimation of a Fractional ARMA(2,1) process with the `fractional_Filter` procedure.

70. **fls1.prg**
We compare the FLS and OLS methods by applying them to the following model for $t = 1, \ldots, N$
$$y_t = \beta_{1,t}x_{1,t} + \beta_{2,t}x_{2,t} + \beta_{3,t}x_{3,t} + u_t$$
with
$$\beta_{1,t} = 1$$
$$\beta_{2,t} = \sin\left(\frac{2\pi}{N}t\right) + v_{2,t}$$
$$\beta_{3,t} = 0.9\beta_{3,t-1} + v_{3,t}$$
u_t, $v_{2,t}$ and $v_{3,t}$ are Gaussian processes. For the FLS regression, we pose
$$\mu = \begin{bmatrix} 10000 \\ 1 \\ 1 \end{bmatrix}$$

71. **fls2.prg**
We graph the residual efficiency frontier $\left\{\left(r_D^2(\mu), r_M^2(\mu)\right), \mu \in \mathbb{R}_+\right\}$ of the preceding model.

72. **fls3.prg**
We consider the model
$$y_t = x_t\beta_t + u_t$$
with
$$\beta_t = \begin{cases} z & \text{if } t \leq S \\ w & \text{if } t > S \end{cases}$$
We estimate β_t with the FLS, RLS and OLS methods. We show that FLS can detect an unanticipated shift from z to w.

73. **fls4.prg**
An example to illustrate the convergence of the FLS estimates to the OLS estimates as μ tends to $+\infty$. We consider different values for μ: 10^4, 10^6, 10^7, 10^8, 4×10^8 and 5×10^9.

74. **fractal1-4.prg — fractal.src**
Different examples to illustrate the estimation of the fractional parameter using wavelets. In *fractal1.prg*, we estimate the d parameter for a white noise process. The method proposed by Wornell and Oppenheim [1992] is based on the complete wavelet coefficients. But, we can use coefficients for just some levels (and not for all the scales). In *fractal2.prg*, we consider a fractional process with $d = 0.25$. The examples *fractal3.prg* and *fractal4.prg* compute the empirical density of the wavelet and GPH estimators.

75. **gfls1.prg**
We compare the GFLS and FLS methods with each other on the following model for $t = 1, \ldots, N$

$$y_t = \beta_{1,t} x_{1,t} + \beta_{2,t} x_{2,t} + \beta_{3,t} x_{3,t} + u_t$$

with $\beta_{1,t}$ a constant, $\beta_{2,t}$ a parameter with seasonal path and $\beta_{3,t}$ a time-varying parameter.

76. **gfls2.prg**
An example to show that the FLS method is a special case of the GFLS method.

77. **gfls3.prg**
We consider the following multi-dimensional process

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix} = \begin{bmatrix} x_{1,t} & 0 & x_{3,t} \\ 0 & x_{2,t} & x_{3,t} \end{bmatrix} \begin{bmatrix} \beta_{1,t} \\ \beta_{2,t} \\ \beta_{3,t} \end{bmatrix} + \begin{bmatrix} u_{1,t} \\ u_{2,t} \end{bmatrix}$$

with $u_{1,t}$ and $u_{2,t}$ two white noise processes and $\beta_{1,t}$, $\beta_{2,t}$ and $\beta_{3,t}$ three time-varying parameters. The corresponding approximately linear system is

$$\left\{ \begin{array}{ccc} \begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix} & \simeq & \begin{bmatrix} x_{1,t} & 0 & x_{3,t} \\ 0 & x_{2,t} & x_{3,t} \end{bmatrix} \begin{bmatrix} \beta_{1,t} \\ \beta_{2,t} \\ \beta_{3,t} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} \beta_{1,t+1} \\ \beta_{2,t+1} \\ \beta_{3,t+1} \end{bmatrix} & \simeq & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_{1,t} \\ \beta_{2,t} \\ \beta_{3,t} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{array} \right.$$

We can also estimate $\beta_{1,t}$, $\beta_{2,t}$ and $\beta_{3,t}$ with the GFLS filter. For the first estimation, we set $D_t = I_3$, $M_t = I_2$, $Q_0 = I_3$, $\mathbf{p}_0 = \mathbf{0}_3$ and $\mu = 1$. For the second estimation, D_t is equal to

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{10} & 0 \\ 0 & 0 & \frac{1}{10} \end{bmatrix}$$

and μ is set to 10.

78. **gfls4.prg**
In this example, we show the use of GFLS for the estimation of specific and common components. Suppose a two-dimensional process with

$$\left\{\begin{array}{lcl} y_t & = & s_t^y + c_t + u_t^y \\ x_t & = & s_t^x + c_t + u_t^x \end{array}\right.$$

with c_t the common component of y_t and x_t while s_t^y and s_t^x are the two specific components. Let us consider the approximately linear system

$$\left\{\begin{array}{rcl} \begin{bmatrix} y_t \\ x_t \end{bmatrix} & \simeq & \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{1,t} \\ \alpha_{2,t} \\ \alpha_{3,t} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} \alpha_{1,t+1} \\ \alpha_{2,t+1} \\ \alpha_{3,t+1} \end{bmatrix} & \simeq & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{1,t} \\ \alpha_{2,t} \\ \alpha_{3,t} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{array}\right.$$

Then, $\alpha_{1,t}$ and $\alpha_{2,t}$ can be wiewed as the specific components and we can interpret $\alpha_{3,t}$ as the common factor. Note that the choice of Q_0 and $\mathbf{p}_0$ are not very important in this example, because it does not affect the curve form of the estimates (we obtain the same estimates, but with a slight translation).

79. **gfls5.prg**
The local level model takes the approximately linear form:

$$\left\{\begin{array}{rcl} y_t & \simeq & \beta_t \\ \beta_{t+1} & \simeq & \beta_t \end{array}\right.$$

We compare the estimation of the state vector process obtained with the Kalman filter with that given by the GFLS filter (see *ll2.prg* example).

80. **gfls6.prg**
The local linear trend model takes the approximately linear form:

$$\left\{\begin{array}{rcl} y_t & \simeq & \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \delta_t \\ \beta_t \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} \delta_{t+1} \\ \beta_{t+1} \end{bmatrix} & \simeq & \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \delta_t \\ \beta_t \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{array}\right.$$

We compare the estimation of the state vector process obtained with the Kalman filter with the resulting one through the GFLS filter (see *llt2.prg* example).

81. **gmm1a.prg**
We consider the linear model

$$y_t = x_t\beta + u_t \tag{4.10}$$

with $u_t \sim \mathcal{N}(0, \sigma^2)$ and β a 4×1 vector. Let $\theta = \operatorname{vec}\begin{bmatrix} \beta & \sigma \end{bmatrix}$ be the vector of parameters. We estimate θ with GMM by considering the moment conditions

$$\left\{\begin{array}{l} E\left[u_t\right] = 0 \\ E\left[u_t^2 - \sigma^2\right] = 0 \\ E\left[u_t x_{t,i}\right] = 0 \qquad \forall\, i = 1, \ldots, 4 \end{array}\right.$$

Note that we use the analytical gradient to perform GMM.

82. **gmm1b.prg**
Constrained GMM of the model (4.10) with $\beta_1 = \beta_2$.

83. **gmm2a.prg**
The model is

$$\left\{ \begin{array}{lll} y_t & = & \beta_1 + \beta_2 x_t + u_t \\ u_t & \sim & \mathcal{N}(0, h_t^2) \\ h_t^2 & = & \alpha_0^2 + \alpha_1^2 u_{t-1}^2 \end{array} \right.$$

Let $\theta = \operatorname{vec} \left[\begin{array}{cccc} \beta_1 & \beta_2 & \alpha_0 & \alpha_1 \end{array} \right]$ be the vector of parameters. We estimate θ by the ML and GMM methods. For the GMM estimation, we consider the moment conditions

$$\left\{ \begin{array}{l} E_t \left[u_t \right] = 0 \\ E_t \left[u_t^2 - h_t^2 \right] = 0 \\ E_t \left[u_t x_t \right] = 0 \\ E_t \left[\left(u_t^2 - h_t^2 \right) u_{t-1}^2 \right] = 0 \end{array} \right.$$

84. **gmm2b.prg**
This is the same program as gmm2a.prg, but we impose that $\alpha_1 = 0$ (no ARCH effect).

85. **gmm3a.prg**
We consider a geometric Brownian motion process

$$\left\{ \begin{array}{lll} dx_t & = & \mu x_t \, dt + \sigma x_t \, dW_t \\ x\left(t_0\right) & = & x_0 \end{array} \right. \tag{4.11}$$

where W_t is a Wiener process. The solution of the stochastic differential equation (4.11) is

$$x\left(t\right) = x_0 \exp \left[\left(\mu - \frac{1}{2}\sigma^2 \right) \left(t - t_0\right) + \sigma \left(W\left(t\right) - W\left(t_0\right) \right) \right]$$

Let h be the sampling interval of the discrete-time data. We set

$$\varepsilon_t = \ln \frac{x_t}{x_{t-1}} - \left(\mu - \frac{1}{2}\sigma^2 \right) h$$

We can estimate the vector of parameters $\theta = \operatorname{vec} \left[\begin{array}{cc} \mu & \sigma \end{array} \right]$ by maximum likelihood or by the generalized method of moments. For the ML estimation, we have

$$\ell_t = -\frac{1}{2} \ln\left(2\pi\right) - \frac{1}{2} \ln\left(\sigma^2 h\right) - \frac{1}{2} \frac{\varepsilon_t^2}{\sigma^2 h}$$

For the GMM estimation, we consider the two moment conditions

$$\left\{ \begin{array}{l} E_{t-1} \left[\varepsilon_t \right] = 0 \\ E_{t-1} \left[\varepsilon_t^2 - \sigma^2 h \right] = 0 \end{array} \right.$$

86. **gmm3b.prg**
We consider an Ornstein-Uhlenbeck process

$$\left\{ \begin{array}{lll} dx_t & = & a\left(b - x_t\right) \, dt + \sigma \, dW_t \\ x\left(t_0\right) & = & x_0 \end{array} \right. \tag{4.12}$$

The solution of the stochastic differential equation (4.12) is

$$x(t) = x_0 e^{-a(t-t_0)} + b\left(1 - e^{-a(t-t_0)}\right) + \sigma \int_{t_0}^{t} e^{a(\theta - t)}\, dW(\theta)$$

We define

$$\varepsilon_t = x_t - e^{-ah} x_{t-1} - b\left(1 - e^{-ah}\right)$$

Let $\theta = \operatorname{vec}\left[\begin{array}{ccc} a & b & \sigma \end{array}\right]$ be the vector of parameters. The expression of the log-likelihood is

$$\ell_t = -\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln\left(\sigma^2\left(\frac{1 - e^{-2ah}}{2a}\right)\right) - \frac{1}{2}\frac{\varepsilon_t^2}{\sigma^2\left(\frac{1-e^{-2ah}}{2a}\right)}$$

GMM estimation of θ can be performed by considering the following moment conditions

$$\left\{\begin{array}{l} E_{t-1}[\varepsilon_t] = 0 \\ E_{t-1}\left[\varepsilon_t^2 - \sigma^2\left(\frac{1-e^{-2ah}}{2a}\right)\right] = 0 \\ E_{t-1}[\varepsilon_t x_{t-1}] = 0 \end{array}\right.$$

87. **gmm3c.prg**
Chan, Karolyi, Longstaff and Sanders [1992] consider the following stochastic differential equation

$$\left\{\begin{array}{rcl} dy_t & = & (\alpha + \beta y_t)\, dt + \sigma \left|y_t\right|^{\gamma} dW_t \\ y(t_0) & = & x_0 \end{array}\right. \tag{4.13}$$

To estimate the vector of parameters $\theta = \operatorname{vec}\left[\begin{array}{cccc} \alpha & \beta & \gamma & \sigma \end{array}\right]$, they use the discrete-time model

$$y_{t+1} - y_t = (\alpha + \beta y_t)\, h + \varepsilon_{t+1}$$

with $\varepsilon_{t+1} \sim \mathcal{N}(0, \sigma^2 \left|y_t\right|^{2\gamma} h)$. They consider the following moment conditions

$$\left\{\begin{array}{l} E_t[\varepsilon_{t+1}] = 0 \\ E_t\left[\varepsilon_{t+1}^2 - \sigma^2 \left|y_t\right|^{2\gamma} h\right] = 0 \\ E_t[\varepsilon_{t+1} y_t] = 0 \\ E_t\left[\left(\varepsilon_{t+1}^2 - \sigma^2 \left|y_t\right|^{2\gamma} h\right) y_t\right] = 0 \end{array}\right.$$

to estimate θ with GMM. In this example, we simulate an Ornstein-Ulhenbeck. Then, we estimate the parameters of the stochastic differential equation (4.13). The Ornstein-Uhlenbeck is a special case of the model (4.13) by imposing $\gamma = 0$. In this case, we have the following correspondence

$$\left\{\begin{array}{rcl} \alpha & = & ab \\ \beta & = & -a \end{array}\right.$$

88. **gmm4a-4i.prg**
Parameters estimation of the Bernoulli, Binomial, Negative Binomial, Poisson, Gamma, Beta, Laplace-Gauss, Log-normal and Exponential distributions.

89. **gmm5a-5b.prg**
Parameters estimation of univariate ARMA processes.

90. **gmm6a-6c.prg**
Parameters estimation of state space models.

91. **golay1.prg**
The program computes the coefficients of the Savitzky-Golay filter for different values of M, n_L and n_R. It replicates the table given on page 646 in Press, Teukolsky, Vetterling and Flannery [1992].

92. **golay2.prg**
An illustration of the `Savitzky_Golay` procedure applied to noisy data.

93. **gph1.prg**
Geweke and Porter-Hudak [1983] suggested the following regression to estimate the fractional integration order d of a time series

$$\ln I\left(\lambda_j\right) = c - d \sin^2 \frac{\lambda_j}{2} + u_t \tag{4.14}$$

We employ this method to estimate the fractional root of a white noise process. We test $d = 0$.

94. **gph2.prg**
Reisen [1994] proposes employing the smoothed periodogram in regression (4.14). The series used is a random walk process. We compare the estimates of d based on the periodogram and those based on the smoothed periodogram with the Parzen lag window generator. Then, we test the null hypothesis $d = 1$.

95. **gph3.prg**
We estimate the fractional root of a white noise process by using different smoothed periodograms and then we test if the hypothesis $d = 0$ cannot be rejected.

96. **gph4.prg**
This example is a Monte Carlo investigation of the power of the GPH estimator and the estimator based on a smoothed periodogram (Bartlett and Tukey with the parameter equal to 0.20). To obtain the density of the different estimators, we use the kernel estimator.

97. **gph5.prg**
In the frequency domain, we estimate an ARFIMA process in two ways. The first one consists of estimating all the parameters by maximizing the log-likelihood function. In the second method, we use the GPH estimator to estimate the fractional part of the ARFIMA model and we estimate the ARMA part of the ARFIMA model.

98. **gph6.prg**
Monte Carlo experiments of the standard errors of the GPH estimator and those based on the smoothing periodogram.

99. **hankel1.prg**
Hankel matrix of a univariate time series.

100. **hankel2.prg**
Hankel matrix of a multivariate time series.

101. **hankel3.prg**
Monte Carlo experiments of the singular value decomposition of the Hankel matrix for white noise and AR(1) processes.

102. **hankel4.prg**
Monte Carlo experiments of the singular value decomposition of the Hankel matrix for an AR(2) process.

103. **hankel5.prg**
McMillan order of a VAR process.

104. **hankel6.prg**
Computes the theoretical and the empirical Hankel matrices of the ARMA(1,1) model (4.5).

105. **hankel7.prg**
In this example, we check for the McMillan order of various state space models to be equal to the number of state variables.

106. **hurst1.prg — hurst.src**
R/S statistic and Hurst exponent with a white noise process.

107. **hurst2.prg — hurst.src**
R/S statistic and Hurst exponent with a fractional process.

108. **hurst3.prg — hurst2.src**
Estimates the Hurst exponent with the method described in Taqqu, Tererovsky and Willinger [1995].

109. **icss1-2.prg — icss.src**
Detection of changes of variance by the ICSS algorithm.

110. **impuls1a-2b.prg — impuls.txt**
Computes the standard errors of the impulse responses by simulation techniques.

111. **jump.prg**
An example of jump and sharp cust detection by wavelets.

112. **kalman1a.prg**
We consider the following state space model

$$\left\{ \begin{array}{l} \begin{bmatrix} y_{1,t} \\ y_2, t \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} + \begin{bmatrix} 10 \\ 0 \end{bmatrix} + \varepsilon_t \\ \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} = \begin{bmatrix} 0.5 & 0.3 \\ 0 & 0.2 \end{bmatrix} \begin{bmatrix} \alpha_{t-1} \\ \beta_{t-1} \end{bmatrix} + \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \eta_t \end{array} \right. \tag{4.15}$$

with

$$H = E\left[\varepsilon_t \varepsilon_t^\top\right] = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.16}$$

and $Q = E\left[\eta_t \eta_t\right] = 1$. We build the state space model in a time-invariant form.

113. **Kalman1b.prg**
We build the state space model (4.15) in a time-variant form.

114. **kalman1c.prg**
We simulate the state space model (4.15).

115. **kalman1d.prg**
Kalman filtering of the state space model (4.15) in its time-invariant form. $\mathbf{a}_0$ and P_0 are computed using the `SSM_ic` procedure.

116. **kalman1e.prg**
Kalman filtering of the state space model (4.15) in its time-variant form.

117. **kalman1f.prg**
Graphical representation of the estimated value of α_t with its 95% confidence interval.

118. **kalman1g.prg**
Graphical representation of the log-likelihood vector.

119. **kalman1h.prg**
Exact Maximum likelihood estimation of the model

$$\left\{ \begin{array}{l} \begin{bmatrix} y_{1,t} \\ y_2,t \end{bmatrix} = \begin{bmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{bmatrix} \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} + \begin{bmatrix} \theta_3 \\ 0 \end{bmatrix} + \varepsilon_t \\ \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} = \begin{bmatrix} \theta_6 & \theta_7 \\ 0 & \theta_8 \end{bmatrix} \begin{bmatrix} \alpha_{t-1} \\ \beta_{t-1} \end{bmatrix} + \begin{bmatrix} \theta_9 \\ \theta_{10} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \eta_t \end{array} \right. \tag{4.17}$$

with

$$H = \begin{bmatrix} \theta_4 & 0 \\ 0 & \theta_5 \end{bmatrix} \tag{4.18}$$

and $Q = \theta_{11}$.

120. **kalman1i.prg**
Conditional MLE with $\mathbf{a}_0 = \mathbf{0}$ and $P_0 = 0_{2\times 2}$ in the time-variant form. Note the use of external variables.

121. **kalman1j.prg**
Smoothing of the estimated model.

122. **kalman1k.prg**
Forecasting of the estimated model.

123. **kalman2a.prg**
Maximum likelihood estimation of the state space model

$$\left\{\begin{array}{l} y_t = \left[\begin{array}{ccc} 1 & x_t & t \end{array}\right] \left[\begin{array}{c} \beta_{0,t} \\ \beta_{1,t} \\ \beta_{2,t} \end{array}\right] + \varepsilon_t \\ \left[\begin{array}{c} \beta_{0,t} \\ \beta_{1,t} \\ \beta_{2,t} \end{array}\right] = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} \beta_{0,t} \\ \beta_{1,t} \\ \beta_{2,t} \end{array}\right] + \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} \eta_{0,t} \\ \eta_{1,t} \\ \eta_{2,t} \end{array}\right] \end{array}\right. \tag{4.19}$$

The unknown parameters θ are $H = \theta_1^2$ and

$$Q = \left[\begin{array}{ccc} \theta_2^2 & 0 & 0 \\ 0 & \theta_3^2 & 0 \\ 0 & 0 & \theta_4^2 \end{array}\right] \tag{4.20}$$

$\mathbf{a}_0$ and P_0 are set to the null vector and matrix respectively.

124. **kalman2b.prg**
Same program as *kalman2a.prg*, but $\mathbf{a}_0$ and P_0 are fixed differently. This program shows the initialization problem of the Kalman filter.

125. **kalman3a.prg**
We simulate a linear process with ARMA parameters.

126. **kalman3b.prg**
Conditional maximum likelihood of the corresponding state space model

$$\left\{\begin{array}{l} y_t = \left[\begin{array}{cc} x_t & 0 \end{array}\right] \left[\begin{array}{c} \beta_t \\ \eta_t \end{array}\right] + \varepsilon_t \\ \left[\begin{array}{c} \beta_t \\ \eta_t \end{array}\right] = \left[\begin{array}{cc} \phi_1 & -\theta_1 \\ 0 & 0 \end{array}\right] \left[\begin{array}{c} \beta_{t-1} \\ \eta_{t-1} \end{array}\right] + \left[\begin{array}{c} 1 \\ 1 \end{array}\right] \eta_t \end{array}\right. \tag{4.21}$$

The estimated parameters are ϕ_1, θ_1, σ_ε and σ_η.

127. **kalman3c.prg**
Conditional maximum likelihood of another representation of the above state space model

$$\left\{\begin{array}{l} y_t = \left[\begin{array}{ccc} x_t & 0 & 1 \end{array}\right] \left[\begin{array}{c} \beta_t \\ \eta_t \\ \varepsilon_t \end{array}\right] \\ \left[\begin{array}{c} \beta_t \\ \eta_t \\ \varepsilon_t \end{array}\right] = \left[\begin{array}{ccc} \phi_1 & -\theta_1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} \beta_{t-1} \\ \eta_{t-1} \\ \varepsilon_{t-1} \end{array}\right] + \left[\begin{array}{cc} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{array}\right] \left[\begin{array}{c} \eta_t \\ \varepsilon_t \end{array}\right] \end{array}\right.$$

This program is an illustration of the identification problem.

128. **kalman4a.prg**
Suppose that we observe a process y_t with a measurement error ε_t. We note z_t the observed process. We have

$$z_t = y_t + \varepsilon_t \tag{4.22}$$

We suppose that y_t is an ARMA(1,1) process

$$y_t = \phi_1 y_{t-1} + u_t - \theta_1 u_{t-1} \tag{4.23}$$

The state space form of this model is

$$\left\{ \begin{array}{l} z_t = \left[\begin{array}{cc} 1 & 0 \end{array} \right] \left[\begin{array}{c} y_t \\ u_t \end{array} \right] + \varepsilon_t \\ \left[\begin{array}{c} y_t \\ u_t \end{array} \right] = \left[\begin{array}{cc} \phi_1 & -\theta_1 \\ 0 & 0 \end{array} \right] \left[\begin{array}{c} y_{t-1} \\ u_{t-1} \end{array} \right] + \left[\begin{array}{c} 1 \\ 1 \end{array} \right] u_t \end{array} \right. \tag{4.24}$$

We simulate the ARMA plus noise process and then we use the Kalman filter to obtain the estimate of the unobserved component y_t.

129. **kalman4b.prg**
In this example, we estimate the coefficients ϕ_1, θ_1, σ_u and σ_ε of the model (4.24) by maximum likelihood in the time domain.

130. **kalman4c.prg**
We estimate the ARMA plus noise model by maximum likelihood in the frequency domain. The corresponding spectral generating function is

$$g(\lambda_j) = \sigma_u^2 \frac{1 - 2\theta_1 \cos\lambda_j + \theta_1^2}{1 - 2\phi_1 \cos\lambda_j + \phi_1^2} + \sigma_\varepsilon^2 \tag{4.25}$$

131. **kalman4d.prg**
We estimate the model (4.24) under the restriction $\theta_1 = 0$. This restriction can be written as:

$$\left[\begin{array}{c} \phi_1 \\ \theta_1 \\ \sigma_u \\ \sigma_\varepsilon \end{array} \right] = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right] \left[\begin{array}{c} \phi_1 \\ \sigma_u \\ \sigma_\varepsilon \end{array} \right] + \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \end{array} \right] \tag{4.26}$$

The restricted ML estimates are obtained both in the frequency domain (with the `FD_cml` procedure) and in the time domain (with the `TD_cml` procedure).

132. **kalman4e.prg**
Illustrate the `KForecasting` procedure to obtain forecasts of a process.

133. **kalman4f.prg**
In the model (4.24), we compute the smoothed component $\mathbf{a}_{t|T}$. If the variance of ε_t is zero, then we must verify that the first component of $\mathbf{a}_{t|T}$ is just equal to z_t or y_t.

134. **kalman4g.prg**
Smoothing the model (4.24) with the Kalman filter.

135. **kalman5a.prg**
Modelling the *lutkepohl* data as a VAR(2) process

$$\left[\begin{array}{c} y_{1,t} \\ y_{2,t} \\ y_{3,t} \end{array} \right] = \left[\begin{array}{c} \mu_1 \\ \mu_2 \\ \mu_2 \end{array} \right] + \Phi_1 \left[\begin{array}{c} y_{1,t-1} \\ y_{2,t-1} \\ y_{3,t-1} \end{array} \right] + \Phi_2 \left[\begin{array}{c} y_{1,t-2} \\ y_{2,t-2} \\ y_{3,t-2} \end{array} \right] + \left[\begin{array}{c} \varepsilon_{1,t} \\ \varepsilon_{2,t} \\ \varepsilon_{3,t} \end{array} \right] \tag{4.27}$$

with $\varepsilon_t \sim \mathcal{N}(\mathbf{0}, \Sigma)$. We estimate this model in the time domain with the Kalman filter. We use the Cholesky decomposition to define the matrix Q, that is $Q = \mathbb{P}\mathbb{P}^\top$. The estimated vector θ corresponds to

$$\begin{bmatrix} \operatorname{vec}(\Phi_1) \\ \operatorname{vec}(\Phi_2) \\ \mu \\ \operatorname{vech}(\mathbb{P}) \end{bmatrix}$$

with vech the operator in Lütkepohl sense.

136. **kalman5b.prg**
Does not income/consumption $(y_{2,t} - y_{3,t})$ cause investment $(y_{1,t})$? We can test this hypothesis by using the Wald statistic. Note that this hypothesis corresponds to the fact that the matrices Φ_1 and Φ_2 are of the form

$$\begin{bmatrix} \cdot & 0 & 0 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$$

This is equivalent to test $\theta_4 = \theta_7 = \theta_{13} = \theta_{16} = 0$.

137. **kalman5c.prg**
Using the results of the t-statistics in the *kalman5a.prg* example, we impose that the following coefficients are zero:

$$\{\theta_2, \theta_3, \theta_4, \theta_5, \theta_7, \theta_{10}, \theta_{11}, \theta_{12}, \theta_{13}, \theta_{14}, \theta_{16}, \theta_{17}, \theta_{18}, \theta_{19}, \theta_{23}\}$$

The restricted model is estimated by maximum likelihood in the time domain.

138. **kalman5d.prg**
We check the accuracy of the above restrictions. To this end, we use the Likelihood Ratio (LR) and the Lagrange Multiplier (LM) statistics. The LM test is computed using the different matrices of the `_ml_derivatives` external variable.

139. **kalman5e.prg**
Another way to compute the LM tests with the `TDml_derivatives` procedure.

140. **kalman5f.prg**
Computes the LM test with an OPG artificial regression.

141. **kalman6a.prg**
We study a time-variant model

$$y_t = \beta_{0,t} x_{0,t} + \beta_{1,t} x_{1,t} + u_t \tag{4.28}$$

with $u_t \sim \mathcal{N}\left(0, \sigma_u^2\right)$ and

$$\left\{ \begin{array}{lcl} \beta_{0,t} & = & \beta_{0,t-1} + v_0 \\ \beta_{1,t} & = & \beta_{1,t-1} + v_1 \end{array} \right. \tag{4.29}$$

with $\begin{bmatrix} v_0 \\ v_1 \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \Sigma_v)$. We suppose that

$$\Sigma_v = \begin{bmatrix} \sigma_0^2 & 0 \\ 0 & \sigma_1^2 \end{bmatrix}$$

The state space form of the model (4.28-4.29) is

$$\left\{ \begin{array}{rcl} y_t & = & \begin{bmatrix} x_{0,t} & x_{1,t} \end{bmatrix} \begin{bmatrix} \beta_{0,t} \\ \beta_{1,t} \end{bmatrix} + u_t \\ \begin{bmatrix} \beta_{0,t} \\ \beta_{1,t} \end{bmatrix} & = & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_{0,t-1} \\ \beta_{1,t-1} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \end{bmatrix} \end{array} \right. \tag{4.30}$$

This example shows how to construct a time-variant state space model. Next, we use the `KFiltering` and the `KSmoothing` procedures to estimate the unobservable components $\beta_{0,t}$ and $\beta_{1,t}$.

142. **kalman6b.prg**
Maximum Likelihood of the model (4.30). Note the declaration of `sigma` as an external variable and the definition of `sigma` in the `ml` procedure.

143. **kfgain1-2.prg**
Illustration of the `KF_gain` procedure.

144. **kernel1.prg**
Density estimation (normal random number).

145. **kernel2.prg**
Density estimation (χ_2 random number) with the truncated (at left) normal kernel.

146. **kernel3.prg**
Density estimation (uniform random number) with the trunacted normal (at left and right) kernel.

147. **kernel4.prg**
We investigate the empirical probability density of the FRF/DEM return for different scales : 1, 2, 5, 10 and 30 days. We use the thresholding method to compare the "noisy" density with the "denoised" density.

148. **kpss.prg — kpss.src**
KPSS statistic.

149. **ks1.prg**
Computes the Kolmogorov-Smirnov test for a white noise process presented in Brockwell and Davis [1991].

150. **ks2.prg**
Computes the Kolmogorov-Smirnov test for a unit root process.

151. **ks3.prg**
Computes the Kolmogorov-Smirnov test for the random walk plus noise model applied to the *purse* data.

152. **ll1.prg**
Estimates the Local Level model for the *purse* data in the frequency domain with the method of scoring and the BFGS algorithm.

153. **ll2.prg**
Estimates the unobserved component of the *purse* Local Level model.

154. **ll3.prg**
Estimates the *purse* Local Level model in the time domain. This program shows the importance of the choice of the initial conditions.

155. **llt1.prg**
Estimates the Local Linear model for the *gnp* data in the frequency domain with the method of scoring and the BFGS algorithm.

156. **llt2.prg**
Estimates the unobserved component of the *gnp* Local Linear model.

157. **llt3.prg**
Estimates the *gnp* Local Linear model in the time domain with the BHHH algorithm.

158. **matrix1.prg**
Computes the matrices $\mathbf{L}_4$, $\mathbf{D}_4$ and $\mathbf{K}_{4,3}$.

159. **matrix2.prg**
Shows the difference between the `vech` and `vech_` operators.

160. **matrix3.prg**
Verifies the following propositions (Lütkepohl [1991]) for $m = 1, \ldots, 10$

$$\mathbf{L}_m \mathbf{D}_m = I_{m(m+1)/2}$$

$$\mathbf{K}_{m,m} \mathbf{D}_m = \mathbf{D}_m$$

$$\mathbf{K}_{m,1} = \mathbf{K}_{1,m} = I_m$$

$$\operatorname{trace}\left(\mathbf{K}_{m,m}\right) = m$$

$$\operatorname{trace}\left(\mathbf{D}_m^\top \mathbf{D}_m\right) = m^2$$

$$\mathbf{L}_m \mathbf{L}_m^\top = I_{m(m+1)/2}$$

$$\operatorname{trace}\left(\mathbf{D}_m^\top \mathbf{D}_m\right)^{-1} = \frac{m(m+3)}{4}$$

161. **matrix4-5.prg**
An illustration of the `xpnd2` procedure with real and complex matrices.

162. **matrix6a-6d.prg**
An illustration of the `Explicit_to_Implicit` and `Implicit_to_Explicit` procedures.

163. **missing1.prg**
Illustration of the `Missing` procedure.

164. **nw.prg — nw.src**
Newey and West estimator of the variance.

165. **optmum2a-2c.prg**
Some examples with the `optmum2` procedure.

166. **pdgm1.prg**
Computes the periodogram of the *Lynx* data and the Fisher's g statistic.

167. **pdgm2.prg**
Smoothed periodogram of a stable AR(2) process.

168. **pdgm3.prg**
Periodogram of the *sunspots* data.

169. **pdgm4.prg**
Covariogram of a time series using the `PDGM` and `inverse_fourier` procedures.

170. **pdgm5.prg**
We consider the state space model

$$\left\{ \begin{array}{l} \left[\begin{array}{c} y_{1,t} \\ y_{2,t} \end{array} \right] = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] \left[\begin{array}{c} \alpha_{1,t} \\ \alpha_{2,t} \end{array} \right] + \varepsilon_t \\ \left[\begin{array}{c} \alpha_{1,t} \\ \alpha_{2,t} \end{array} \right] = \left[\begin{array}{cc} 0.5 & 0.3 \\ 0 & -0.5 \end{array} \right] \left[\begin{array}{c} \alpha_{1,t-1} \\ \alpha_{2,t-1} \end{array} \right] + \left[\begin{array}{c} 1 \\ 1 \end{array} \right] \eta_t \end{array} \right. \tag{4.31}$$

with $\eta_t \sim \mathcal{N}(0, 0.25)$ and

$$\varepsilon_t \sim \mathcal{N}(\left[\begin{array}{c} 0 \\ 0 \end{array} \right], \left[\begin{array}{cc} 0.2 & 0 \\ 0 & 0.1 \end{array} \right])$$

We compare the spectral density of the state space model with the smoothed periodogram of a realization of the model.

171. **pdgm6.prg**
We compare the theoretical covariances and the cross-covariances of the model (4.31) with the empirical covariances and the cross-covariances of a realization.

172. **pdgm7.prg**
Spectral density of a univariate ARMA(2,1) process.

173. **qmf1.prg**
In this example, we verify the following properties of quadrature mirror filters:

$$\left\{ \begin{array}{rcl} \sum_{k=0}^{p} h_k & = & \sqrt{2} \\ \sum_{k=0}^{p} g_k & = & 0 \\ \sum_{k=0}^{p} h_k^2 & = & 1 \\ \sum_{k=0}^{p} h_k g_k & = & 0 \end{array} \right.$$

174. **qmf2.prg**
Same program as *qmf1.prg* with Pollen filters.

175. **riccati1.prg**
Solves the Algebraic Riccati Equation for the state space model given in exercise 4.8 by Harvey [1990].

176. **rls1.prg**
We apply the `RLS` procedure to a time-invariant model.

177. **rls2.prg**
We apply the `RLS` procedure to a model whose coefficients follow a random walk process.

178. **robinson.prg — robinson.src**
Estimates the Hurst exponent with Robinson's [1995] method in the frequency domain.

179. **scalogrm.prg**
This example is taken from Arino and Vidakovic [1995]. The scalogram can be used to decompose a time series into different time series. We consider a time series x_t which is the sum of two time series y_t and z_t, that is we have

$$x_t = y_t + z_t$$

The first component y_t is a trend and the second component z_t is a cycle. Wavelet analysis is useful for describing the time series x_t because the trend is better localized for high scales (the cycle is better localized for the first scales).

180. **spectrum.prg**
There are several techniques to estimate the power spectrum with wavelet or wavelet packet. Firstly, we compute the periodogram. Secondly, we calculate the coefficients of the wavelet transform. Thirdly, we transform the coefficients. Finally, we compute the inverse wavelet transform. We can use thresholding techniques to perform the transformation. In this example, we transform the wavelet coefficients by extracting some subbands.

181. **ssm1-5.prg**
Printing state space models.

182. **ssm6a.prg**
Same program as *varx1e.prg*, but responses to forecast errors are computed with the *SSM_impulse* procedure.

183. **ssm6b.prg**
Same program as *varx1e.prg*, but responses to orthogonal impulses are computed with the *SSM_orthogonal* procedure.

184. **ssm6c.prg**
Same program as *varx1d.prg*, but we compute the forecast error variance decomposition with the *SSM_fevd* procedure.

185. **ssm7a.prg**
Same program as *arma1k.prg*, but responses to forecast errors are computed with the *SSM_impulse* procedure.

186. **ssm7b.prg**
Same program as *arma1k.prg*, but responses to orthogonal impulses are computed with the *SSM_orthogonal* procedure.

187. **ssm7c.prg**
Same program as *arma1j.prg*, but we compute the forecast error variance decomposition with the *SSM_fevd* procedure.

188. **ssm8a.prg**
We consider the state space model

$$\left\{ \begin{array}{l} \begin{bmatrix} y_{1,t} \\ y_{2,t} \\ y_{3,t} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 4 & 2 \\ 2 & -3 \end{bmatrix} \begin{bmatrix} \alpha_{1,t} \\ \alpha_{2,t} \end{bmatrix} + \varepsilon_t \\ \begin{bmatrix} \alpha_{1,t} \\ \alpha_{2,t} \end{bmatrix} = \begin{bmatrix} .5 & .45 \\ -.5 & .8 \end{bmatrix} \begin{bmatrix} \alpha_{1,t} \\ \alpha_{2,t} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \eta_{1,t} \\ \eta_{2,t} \end{bmatrix} \end{array} \right. \tag{4.32}$$

with

$$\varepsilon_t \sim \mathcal{N}\left(\mathbf{0}_3, \begin{bmatrix} 5 & 1 & 0 \\ 1 & 4 & 0 \\ 0 & 0 & 8 \end{bmatrix}\right)$$

and

$$\begin{bmatrix} \eta_{1,t} \\ \eta_{2,t} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}_2, \begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix}\right)$$

We compute the responses to the forecast error $\mathbf{e} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$.

189. **ssm8b.prg**
We compute the responses to the forecast error $\mathbf{e} = \begin{bmatrix} 1 & -1 \end{bmatrix}^\top$ for the state space model (4.32).

190. **ssm9a.prg**
We compute the responses to the orthogonal impulse $\mathbf{e} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ for the state space model (4.32).

191. **ssm9b.prg**
We compute the responses to the orthogonal impulse $\mathbf{e} = \begin{bmatrix} 1 & -1 \end{bmatrix}^\top$ for the state space model (4.32).

192. **ssm10.prg**
We compute the forecast error variance decomposition for the state space model (4.32).

193. **surrog1.prg**
Surrogate data in the univariate case.

194. **surrog2.prg**
Surrogate data in the multivariate case.

195. **surrog3.prg — rk4.src**
Surrogate data can be used to detect non-linearities. In this example, we use the Lorenz model defined by

$$\begin{cases} \frac{dx}{dt} &= \sigma(y-x) \\ \frac{dy}{dt} &= -xz+Rx-y \\ \frac{dz}{dt} &= xy-\beta z \end{cases}$$

196. **tdml1a.prg**
Teräsvirta [1994] suggests a LSTAR model to fit the *lynx* data

$$\begin{aligned} x_t &= \beta_1 x_{t-1} + [\beta_2 x_{t-2} + \beta_3 x_{t-3} + \beta_4 x_{t-4} + \beta_5 x_{t-9} + \beta_6 x_{t-11}] \\ &\quad \times [1+\exp(\rho \times 1.8 (x_{t-3}-\theta))]^{-1} + u_t \end{aligned} \tag{4.33}$$

This program estimates the model (4.33). Note the use of the external variable `_tsm_parnm` for the names of the estimated coefficients.

197. **tdml2a.prg**
To model the *lynx* data, Ozaki [1982] suggests to use the EXPAR model

$$\begin{aligned} x_t &= \left[\beta_1 + (\beta_2 + \beta_3 x_{t-1}) \exp\left(-\delta x_{t-1}^2\right)\right] x_{t-1} \\ &\quad \left[\beta_4 + (\beta_5 + \beta_6 x_{t-1}) \exp\left(-\delta x_{t-2}^2\right)\right] x_{t-2} + u_t \end{aligned}$$

with $u_t \sim \mathcal{N}\left(0,\sigma^2\right)$. With the `TD_ml` procedure, we estimate the coefficients $\theta = \begin{bmatrix} \beta^\top & \delta & \sigma \end{bmatrix}^\top$.

198. **tdml2b.prg**
This is a modification of the *tdml2a.prg* example by setting δ to 3.89.

199. **tdml3a.prg**
Maximum Likelihood estimation of a linear model.

200. **tdml3b.prg**
Maximum Likelihood estimation of a linear model under linear restrictions.

201. **tdml4a.prg**
Maximum Likelihood of the linear model with AR(1) errors:

$$\begin{cases} y_t &= x_t\beta + u_t \\ u_t &= \rho u_{t-1} + \varepsilon_t \end{cases}$$

with $\varepsilon_t \sim N\left(0,\sigma^2\right)$. The parameter vector θ is $\begin{bmatrix} \beta^\top & \rho & \sigma \end{bmatrix}^\top$. The ML function is based on Beach and MacKinnon [1978]. We test also $\rho = 0$ with LM and LR statistics.

202. **tdml4b.prg**
Maximum Likelihood of a PROBIT model. The program contains the normality test for PROBIT models of Bera, Jarque and Lee [1984]. Note that the ML procedure uses the analytical Jacobian.

203. **twofft.prg**
An illustration of the `fourier2` procedure.

204. **varx1a.prg**
Define the following series with the *Lutkepohl* data

$$\begin{aligned} y_{1,t} &= INV(t) - INV(t-1) \\ y_{2,t} &= INC(t) - INC(t-1) \\ y_{3,t} &= CONS(t) - CONS(t-1) \end{aligned}$$

The program estimates the VAR(2) process

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \\ y_{3,t} \end{bmatrix} = \Phi_1 \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \\ y_{3,t-1} \end{bmatrix} + \Phi_2 \begin{bmatrix} y_{1,t-2} \\ y_{2,t-2} \\ y_{3,t-2} \end{bmatrix} + \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} + \varepsilon_t \tag{4.34}$$

and performs a stability analysis. The θ vector of coefficients corresponds to

$$\begin{bmatrix} \operatorname{vec}(\Phi_1) \\ \operatorname{vec}(\Phi_2) \\ \mu \end{bmatrix}$$

205. **varx1b.prg**
Computes the Wald test for no Granger-causality from INC/CONS to INV.

206. **varx1c.prg**
Computes the Wald test for no Instantaneous-causality between INC/CONS and INV.

207. **varx1d.prg**
Forecast Error Variance Decomposition of the above VAR(2) model.

208. **varx1e.prg**
Impulses Responses of the above VAR(2) model.

209. **varx1f.prg**
Impulses Responses of the above VAR(2) model (graphical representation).

210. **varx1g.prg**
VAR order selection with the BIC, AIC alpha, SIC, FPE, AIC and HQ criteria.

211. **varx1h.prg**
Estimates the model (4.34) with the restrictions

$$\Phi_1 = \begin{bmatrix} \cdot & 0 & 0 \\ 0 & 0 & \cdot \\ 0 & \cdot & \cdot \end{bmatrix}$$

$$\Phi_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \cdot & 0 \end{bmatrix}$$

and

$$\mu = \begin{bmatrix} 0 \\ \cdot \\ \cdot \end{bmatrix}$$

212. **varx2a.prg**
Estimation of the Dynamic Simultaneous Equations

$$\begin{bmatrix} \mathrm{INC}_t \\ \mathrm{CONS}_t \end{bmatrix} = \Phi_1 \begin{bmatrix} \mathrm{INC}_{t-1} \\ \mathrm{CONS}_{t-1} \end{bmatrix} + \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \mathrm{INV}_{t-1} + \begin{bmatrix} \varepsilon_{1,t} \\ \varepsilon_{2,t} \end{bmatrix} \tag{4.35}$$

213. **varx2b.prg**
Estimation of the Constrained Dynamic Simultaneous Equations

$$\begin{bmatrix} \mathrm{INC}_t \\ \mathrm{CONS}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \phi_{21} & \phi_{22} \end{bmatrix} \begin{bmatrix} \mathrm{INC}_{t-1} \\ \mathrm{CONS}_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ \mu_2 \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ 0 \end{bmatrix} \mathrm{INV}_{t-1} + \begin{bmatrix} \varepsilon_{1,t} \\ \varepsilon_{2,t} \end{bmatrix} \tag{4.36}$$

214. **varx2c.prg**
Estimation of the model (4.36) by Maximum Likelihood.

215. **varx3a.prg**
Use of the `varx_ls` procedure to compute OLS estimates. The results are compared with those calculated with the `ols` procedure.

216. **varx3b.prg**
We use the `varx_cls` procedure to compute the SUR estimator. The example is taken from Judge, Hill, Griffiths, Lütkepohl and Lee [1988] pages 453 and 454.

217. **varx3c.prg**
We use the `varx_cls` procedure to compute the restricted SUR estimator. The example is taken from Judge, Hill, Griffiths, Lütkepohl and Lee [1988] pages 460 to 462.

218. **varx3d.prg**
We use the `varx_cls` procedure to estimate a system of simultaneous equations. The example is taken from Judge, Hill, Griffiths, Lütkepohl and Lee [1988] pages 656 to 663.

219. **window2a-2b.prg**
Somes examples to show the use of the `window2` procedure.

220. **wn1.prg**
Estimates the white noise model in the frequency domain

$$y_t = \varepsilon_t$$

with $\varepsilon_t \sim \mathcal{N}\left(0, \sigma^2\right)$. Then we plot the empirical distribution of $2\frac{I(\lambda_j)}{g(\lambda_j)}$ and the theoretical χ^2_2 distribution.

221. **wn2.prg**
This is the same program as *wn1.prg* applied to a unit root process.

222. **wn3.prg**
We check if the FRF/DEM is a unit root process.

223. **wpkt1.prg**
We select a wavelet packet basis for a time series with the `BestBasis` and the `BestLevel` procedures based on the entropy cost function.

224. **wpkt2.prg**
We simulate a fractional process. Then, we draw the wavelet packet table of this process. We illustrate the fact that the wavelet transform is a special case of the wavelet packet transform with a special basis.

225. **wt1a.prg-wt1b.prg**
We evaluate the inverse wavelet transform for different unit vectors $\mathbf{e}$ to see how wavelets look like (see PRESS, TEUKOLSKY, VETTERLING and FLANNERY [1992], page 591).

226. **wt2.prg**
Reconstruction of an ARMA process by the quantile thresholding method with different values of p.

227. **wt3.prg**
An important result is that the "mother" coefficient of the wavelet transform of a time series x_t of length $N = 2^M$ is equal to

$$\mathbf{c}_0 = \sum_{t=1}^{N} x_t \Big/ 2^{\frac{M}{2}}$$

We indicate the correspondence between the examples and the procedures:

- `ARE` — *riccati1.prg.*
- `arfima` — *arfima1-4.prg, filter2a.prg, gph5.prg.*
- `arma_autocov` — *arma2a-2c.prg, autocov1-5.prg.*
- `arma_CML` — *arma1c-1h.prg.*
- `arma_fevd` — *arma1j.prg, varx1d.prg.*
- `arma_Filter` — *filter1a-2a.prg, gmm5a-5b.prg.*
- `arma_impulse` — *arma1k.prg, impuls1a-1b.prg, varx1e-1f.prg.*
- `arma_ML` — *arma1a.prg, arma1h.prg, cusum2-3.prg, filter1a.prg.*
- `arma_orthogonal` — *arma1k.prg, impuls2a.prg, varx1e-1f.prg.*
- `arma_roots` — *arma1i.prg, varx1a.prg.*
- `arma_to_SSM` — *arma1b.prg, arma2a-2e.prg, autocov4-5.prg, cml2-5.prg, cusum2-3.prg, fdml6.prg, hankel5-6.prg, impuls1b.prg, impuls2b.prg, pdgm7.prg, ssm6a-7c.prg.*
- `arma_to_VAR1` —
- `Basis` — *chirp1b.prg, basis3.prg, denois2b.prg, wpkt2.prg.*

- `BasisPlot` — *basis2.prg.*
- `BestBasis` — *basis4.prg, wpkt1.prg.*
- `BestLevel` — *basis3.prg, wpkt1.prg.*
- `Bootstrap` —
- `bootstrap_SSM` — *boot1-3.prg, impuls1b.prg, impuls2b.prg.*
- `BSM` — *bsm1-3.prg.*
- `canonical_arfima` — *canon4-7.prg.*
- `canonical_arma` — *canon1-4.prg.*
- `Coiflet` — *chirp1a-1b.prg, qmf1.prg, wt1b.prg, wt3.prg.*
- `commutation_` — *matrix1.prg, matrix3.prg.*
- `CPDGM` — *cpdgm1.prg, cspect1-3.prg.*
- `CSpectrum` — *cspect1-3.prg.*
- `Daubechies` — *denois1a-1d.prg, denois2a-2b.prg, fractal1-4.prg, jump.prg, kernel4.prg, qmf1.prg, scalogrm.prg, spectrum.prg, wpkt1.prg, wt1a-b.prg, wt2-3.prg.*
- `duplication_` — *matrix1.prg, matrix3.prg.*
- `elimination_` — *matrix1.prg, matrix3.prg.*
- `Entropy` — *basis4.prg, wpkt1.prg.*
- `Explicit_to_Implicit` — *matrix6a-6d.prg.*
- `extract` — *band2.prg, spectrum.prg.*
- `FDml_derivatives` — *fdml1b.prg, fdml2b.prg.*
- `FD_cml` — *fdml1b.prg, fdml2b.prg, kalman4d.prg.*
- `FD_ml` — *cycle2-3.prg, fdml1a.prg, fdml2a-2b.prg, fdml3-4.prg, kalman4c.prg, wn1-3.prg.*
- `FLS` — *fls1-4.prg, gfls1-2.prg.*
- `fourier` — *twofft.prg.*
- `fourier2` — *twofft.prg.*
- `fractional_Filter` — *filter2a.prg.*
- `garch_Filter` — *filter1b-1c.prg.*
- `GFLS` — *gfls3-6.prg.*
- `GFLS2` — *gfls1-2.prg.*

- GMM — *gmm1a-6c.prg.*
- Haar — *basis3-4.prg, qmf1.prg, wpkt2.prg.*
- Hankel — *hankel1-4.prg, hankel6.prg.*
- Implicit_to_Explicit — *matrix6a-6d.prg.*
- insert — *band4.prg, denois2a.prg.*
- inverse_fourier — *pdgm4.prg, pdgm6.prg.*
- isBasis — *basis1.prg, basis2.prg.*
- iwpkt — *chirp1b.prg, denois2b.prg.*
- iwt — *chirp1a.prg, denois1a-1d.prg, denois2a.prg, kernel4.prg, scalogrm.prg, spectrum.prg, wt1a-b.prg, wt2.prg.*
- iwt_matrix —
- Kernel — *fractal3-4.prg, gph4.prg, kernel1-4.prg.*
- KFiltering — *arma1b.prg, arma2d-2e.prg, bsm3.prg, cml2-5.prg, cusum1-3.prg, gfls5-6.prg, gmm6a-6c.prg, impuls1b.prg, impuls2b.prg, kalman1d-1k.prg, kalman2a-2b.prg, kalman3b-3c.prg, kalman4a-4b.prg, kalman4d-4g.prg, kalman5a-5f.prg, kalman6a-6b.prg, ll2-3.prg, llt2-3.prg, riccati1.prg.*
- KForecasting — *kalman1k.prg, kalman4e.prg.*
- KF_gain — *kfgain1-2.prg.*
- KF_matrix — *cusum1-3.prg, gfls5-6.prg, gmm6a-6c.prg, kalman1d-1f.prg, kalman1j.prg, kalman2a-2b.prg, kalman4a.prg, kalman4g.prg, kalman6a-6b.prg, ll2.prg, llt2.prg, riccati1.prg.*
- KF_ml — *arma1b.prg, arma2d-2e.prg, bsm3.prg , cml2-5.prg, kalman1g-1i.prg, kalman2a-2b.prg, kalman3b-3c.prg, kalman4b.prg, kalman4d.prg, kalman5a-5f.prg, kalman6b.prg, ll3.prg, llt3.prg.*
- KSmoothing — *gmm6a.prg, kalman1j.prg, kalman4f.prg, kalman4g.prg, kalman6a-6b.prg.*
- Linear_Filter — *icss2.prg.*
- LogEnergy — *basis3.prg, basis4.prg.*
- LpNorm — *basis4.prg.*
- Missing — *fls1.prg, missing1.prg, surrog2.prg, tdml3a-3b.prg, tdml4a-4b.prg.*
- optmum2 — *optmum2a-2c.prg.*
- padding — *kernel4.prg.*

- `PDGM` — *cdgm1.prg, cspect1-3.prg, cycle3.prg, fdml1a.prg, fractal3-4.prg, gph1-6.prg, kalman4c.prg, ks1-3.prg, pdgm1-4.prg, robinson.src, spectrum.prg, surrog1.prg, wn1-3.prg.*
- `PDGM2` — *fdml5-6.prg, pdgm5-6.prg.*
- `Pollen` — *qmf2.prg, wt3.prg.*
- `RLS` — *fls3.prg, rls1-2.prg.*
- `RND_arfima` — *arfima4-5.prg, fractal2.prg, fractal4.prg, gph5.prg, robinson.prg, wpkt2.prg.*
- `RND_arma` — *filter1a-2a.prg, gmm5a-5b.prg, pdgm4.prg, surrog2.prg, wt2.prg.*
- `RND_SSM` — *bsm1-3.prg, fdml5.prg, kalman1c.prg, pdgm5-6.prg.*
- `Savitzky-Golay` — *golay1-2.prg.*
- `Scalogram` — *scalogrm.prg.*
- `select` — *band3.prg, fractal.src, denois2a.prg, jump.prg.*
- `SemiSoft` — *denois1b.prg.*
- `sgf_arfima` — *pdgm7.prg.*
- `sgf_SSM` — *fdml4-6.prg, pdgm5-7.prg.*
- `Smoothing` — *cpdgm1.prg, cycle3.prg, gph2-4.prg, gph6.prg, pdgm2.prg, pdgm5.prg, spectrum.prg.*
- `sm_cycle` —
- `sm_LL` — *cusum1.prg, gfls5.prg, gmm6b.prg, ks3.prg, ll1-2.prg.*
- `sm_LLT` — *llt1-2.prg, gfls6.prg, gmm6a.prg.*
- `split` — *band1.prg, scalogrm.prg.*
- `SSM` — *hankel7.prg, ssm1-5.prg.*
- `SSM_autocov` — *autocov4-6.prg.*
- `SSM_build` — *arma1b.prg, arma2a-2e.prg, autocov4-6.prg, bsm1-3.prg, cml2-5.prg, cusum1-3.prg, fdml4-6.prg, hankel5-7.prg, gfls5-6.prg, gmm6a-6c.prg, impuls1b.prg, impuls2b.prg, kalman1a-1k.prg, kalman2a-2b.prg, kalman3b-3c.prg, kalman4a-4b.prg, kalman4d-4g.prg, kalman5a-5f.prg, kalman6a-6b.prg, ll2-3.prg, llt2-3.prg, pdgm5-7.prg, ssm6a-10.prg.*
- `SSM_fevd` — *ssm6c.prg, ssm7c.prg, ssm10.prg.*
- `SSM_Hankel` — *hankel5-7.prg.*
- `SSM_ic` — *arma1b.prg, arma2a-2e.prg, cml2-5.prg, cusum2-3.prg, kalman1d.prg, kalman1f.prg, kalman4d-4f.prg.*

- `SSM_impulse` — *ssm6a.prg, ssm7a.prg, ssm8a-8b.prg.*
- `SSM_orthogonal` — *impuls2b.prg, ssm6b.prg, ssm7b.prg, ssm9a-9b.prg.*
- `SSM_to_arma` —
- `surrogate` — *surrog1-3.prg.*
- `TDml_derivatives` — *cml5.prg, filter2c.prg, kalman5e-5f.prg.*
- `TD_cml` — *gmm1b.prg, gmm2b.prg, kalman4d.prg, kalman5c-5d.prg, tdml3b.prg, tdml4a.prg.*
- `TD_ml` — *arma1b.prg, arma2d-arma2e.prg, bsm3.prg, cusum2-3.prg, fdml5-6.prg, filter1a-2a.prg, fractal1-4.prg, gmm1a.prg, gmm2a.prg, gmm3a-3b.prg, gmm5a-5b.prg, gmm6a-6c.prg, kalman1h-1i.prg, kalman2a-2b.prg, kalman3b-3c.prg, kalman4b.prg, kalman5a-5b.prg, kalman6b.prg, ll3.prg, llt3.prg, tdml1a.prg, tdml2a-2b.prg, tdml3a.prg, tdml4b.prg.*
- `Thresholding` — *denois1d.prg, denois2b.prg, kernel4.prg, wt2.prg.*
- `TSMset` —
- `varx_CLS` — *varx1h.prg, varx2b.prg, varx3b-3d.prg.*
- `varx_CML` — *varx1h.prg, varx2c.prg.*
- `varx_LS` — *impuls1a-2b.prg, ssm6a-6c.prg, varx1a-1b.prg, varx1e-1f.prg, varx2a.prg, varx3a.prg.*
- `varx_ML` — *impuls2a-2b.prg, kalman5a.prg, varx1c.prg, varx1g.prg.*
- `vech_` — *cml2.prg, matrix2.prg.*
- `VisuShrink` — *denois1c.prg, denois2a.prg.*
- `WaveShrink` — *denois1a.prg.*
- `window2` — *window2a-2b.prg.*
- `wpkPlot` — *chirp1b.prg.*
- `wpkt` — *basis3-4.prg, chirp1b.prg, denois2b.prg, wpkt2.prg.*
- `wPlot` — *chirp1a.prg, denois1a.prg, wt2.prg.*
- `wt` — *chirp1a.prg, denois1a-1d.prg, denois2a.prg, fractal1-4.prg, jump.prg, kernel4.prg, scalogrm.prg, spectrum.prg, wpkt2.prg, wt2-3.prg.*
- `wt_matrix` —
- `xpnd_` — *matrix2.prg.*
- `xpnd2` — *autocov4-6.prg, matrix4-5.prg, ssm6a-7c.prg.*