E^4: A MATLAB Toolbox for Time Series Modeling

Miguel Jerez†
Sonia Sotoca††
José Casals†††

Universidad Complutense de Madrid

Abstract: This paper describes E^4, a MATLAB Toolbox for time series analysis which uses state-space methods to achieve both, flexibility and reliability. Despite its orientation to state-space, E^4 also provides native support for many standard models, such as VARMAX, structural econometric models or single-output transfer functions. These models can be estimated by exact maximum-likelihood, either under standard conditions or in an extended framework that allows for measurement errors, missing data, vector GARCH errors and constraints on the parameters. The Toolbox includes ready-to-use functions for model specification, preliminary parameter estimation, analytical computation of the likelihood gradient and the information matrix, simulation, forecasting and signal extraction. Several examples illustrate these useful features.

Keywords: Time series; MATLAB; state-space; Kalman filter; signal extraction; forecasting

---

† **Corresponding author.** Departamento de Fundamentos del Análisis Económico II. Facultad de Ciencias Económicas. Campus de Somosaguas. 28223 Madrid (SPAIN). Email: mjerez@ccee.ucm.es, tel: (+34) 91 394 23 61, fax: (+34) 91 394 25 91.
†† Departamento de Fundamentos del Análisis Económico II. Facultad de Ciencias Económicas. Campus de Somosaguas. 28223 Madrid (SPAIN). Email: sotoca@ccee.ucm.es
††† Departamento de Fundamentos del Análisis Económico II. Facultad de Ciencias Económicas. Campus de Somosaguas. 28223 Madrid (SPAIN). Email: jcasalsc@cajamadrid.es

1. Introduction.

This paper describes a MATLAB Toolbox for time series modeling. Its name, E^4, refers to the Spanish: ***E**stimación de modelos **E**conométricos en **E**spacio de los **E**stados*, meaning "State-Space Estimation of Econometric Models." E^4 combines the intellectual input from many people who, in different moments of their careers, faced special needs that commercial software could not satisfy.

First of all, we need transparency. Statistical software often hides relevant computational outputs, such as the likelihood gradient or the information matrix conditioning number. This happens because these results convey one of two messages: either that everything worked as expected, or that there is a problem with the model, the data or the software. Giving these news may be not the best way to do business but, for some users at least, they are important news.

Second, we need reliability. Mainstream software tends to prioritize computational speed at the expense of numerical accuracy. This can be done in different ways such as, e.g., by choosing fast but sloppy computational algorithms or easy-to-satisfy default convergence criteria. Speed is crucial for a real-time computational system. For academic work, however, it is an attractive but not-so-valuable feature when compared to precision or computational reliability.

Third, we need consistency. Common time series packages implement different procedures to make the same job for different models and this may produce substantial inconsistencies. This idea may be better explained with a real-life example. Some years ago one of the authors of this paper used an expensive statistical software system to fit alternative ARIMA and intervention models to a given time series. He then tried to assess the significance of the intervention term through a likelihood-ratio test, which value resulted meaninglessly

negative. After revising the results, he found that both likelihood values were computed conditional on different information sets. This discrepancy was not discussed in the extensive documentation of this system, but revealed by the fact that the ARIMA procedure generated a residual series of the same size as the sample, while the analogous transfer function method returned a shorter vector of residuals. These inconsistencies may be due to poor software design or to the intrinsic difficulty of making a clean distinction between models and procedures. While many practitioners may never be aware of these subtle incongruences, they are difficult to avoid for heavy users.

Our fourth and last need has to do with the exploitation of the software generated by research. Academic work often produces neat pieces of code that does well a very specific task. This software could be useful to other people in an adequate framework but, if this framework does not exist, it will probably vanish forever in the limbo where neat pieces of code go after the paper is published. On the other hand, a coordinated effort to put together, organize and document these scientific by-products may be academically profitable for both, its authors and its users.

Our personal answer to these questions is E^4, a MATLAB toolbox where we collect the results from a research line in state-space methods, which originated around 1980.

E^4 is our main research tool and, accordingly, it prioritizes academic needs. As a consequence, its functions provide a transparent, documented and user-manageable output of results, providing many clues about problems such as lack of convergence or ill-conditioning. Also, the code has been carefully optimized for numerical accuracy and robustness, making an

intensive use of stability-inducing methods, such as factorization and scaling. Finally, computational tolerances are transparent, user-manageable and have strict default values.

E^4 does all the important calculations using the equivalent state-space (hereafter SS) representation of the models employed. This feature is important to enforce consistency, because the SS formulation allows for a clean separation between models and procedures. We achieve this by developing all the core algorithms for a SS model. Then, when an E^4 function receives an input model, the first thing it does is obtaining the equivalent SS representation, and only then applies the corresponding procedure. This approach ensures a high degree of coherence in the results, simplifies code development and, last but not least, makes the toolbox very easy to extend, because implementing new models with equivalent SS representations only requires coding new interfaces. Conversely, a new procedure only needs to support the SS model because, thanks to the bridge functions provided, it can be immediately applied to all the models supported.

Another advantage of SS methods is that they allow for certain relevant analyses that would otherwise be very difficult, such as modeling samples with missing data or observation errors. Furthermore, they allowed us to incorporate many efficient and well-tested numerical algorithms, coming mainly from aerospace and communications engineering. These procedures have been intensively used in different parts of the toolbox.

Finally, E^4 has been an effective repository to organize and distribute the code resulting from our research. For us, every new research project adds new functionalities to E^4 which, not only are documented in the user manual but also, with exceptional depth, in the corresponding papers. In year 2000 we launched the E^4 WEB (www.ucm.es/info/icae/e4), which offers the

source code for the Toolbox as well as a complete documentation. This WEB registers a modest but steady flow of downloads, in the order of magnitude of the low hundreds per year.

This paper is part of our coordinated effort to present and disseminate E^4 through the academic community. Its structure is as follows. Section 2 reviews the standard models supported, including SS, VARMAX, transfer functions, structural econometric models and models with GARCH errors. It also describes broadly the functions that define them and shows how a model can be created by joining several "building blocks" through two innovative operations, that we call *model nesting* and *model composition*. Mathematical details for the formulations supported and the model nesting operation are provided in Appendices A and B.

Section 3 provides a panoramic view of the procedures employed. They fall into three broad categories: standard time series methods, signal extraction procedures and algorithms related to model estimation. This revision is necessarily brief, but complete technical details are provided in several articles which are cited in this Section.

Section 4 illustrates the use of E^4 by means of complete input and output data for several applications to real data. Finally, Section 5 provides some concluding remarks and indicates how to obtain E^4.

2. Models supported by E^4.

2.1 The state-space model.

The most important formulation supported by E^4 is the SS model, because this is the standard representation for which most of its core computational algorithms are devised. The basic fixed-coefficients SS formulation is given by a *state equation*, characterizing the system dynamics:

$$\boldsymbol{x}_{t+1} = \boldsymbol{\Phi}\,\boldsymbol{x}_t + \boldsymbol{\Gamma}\boldsymbol{u}_t + \mathbf{E}\,\boldsymbol{w}_t \tag{2.1}$$

and a *state observation equation*, describing how the system output is realized as a linear combination of different components:

$$\boldsymbol{z}_t = \mathbf{H}\,\boldsymbol{x}_t + \mathbf{D}\,\boldsymbol{u}_t + \mathbf{C}\,\boldsymbol{v}_t \tag{2.2}$$

where $\boldsymbol{z}_t \in \mathbb{R}^m$ is a random vector of *endogenous variables* or *outputs*, $\boldsymbol{x}_t \in \mathbb{R}^n$ is vector of *state variables*, $\boldsymbol{u}_t \in \mathbb{R}^r$ is a vector of *exogenous variables* or *inputs*, and $\boldsymbol{w}_t$, $\boldsymbol{v}_t$ are conformable vectors of zero-mean white noise errors, such that

$$\mathrm{E}\left[\begin{pmatrix}\boldsymbol{w}_t\\ \boldsymbol{v}_t\end{pmatrix}\begin{pmatrix}\boldsymbol{w}_t & \boldsymbol{v}_t\end{pmatrix}\right] = \begin{bmatrix}\mathbf{Q} & \mathbf{S}\\ \mathbf{S}^{\mathrm{T}} & \mathbf{R}\end{bmatrix} \tag{2.3}$$

Last $\boldsymbol{\Phi}$, $\boldsymbol{\Gamma}$, **E**, **H**, **D**, **C**, **Q**, **R** and **S** are time-invariant coefficient matrices, being **Q** and **R** positive semi-definite.

A particular case of (2.1)-(2.2) is the steady-state innovations SS model, hereafter "innovations model", defined by:

$$\boldsymbol{x}_{t+1} = \boldsymbol{\Phi}\,\boldsymbol{x}_t + \boldsymbol{\Gamma}\boldsymbol{u}_t + \mathbf{E}\,\boldsymbol{a}_t \tag{2.4}$$

$$\boldsymbol{z}_t = \mathbf{H}\,\boldsymbol{x}_t + \mathbf{D}\,\boldsymbol{u}_t + \boldsymbol{a}_t \tag{2.5}$$

where $\boldsymbol{a}_t \in \mathbb{R}^m$ is such that $\boldsymbol{a}_t \sim iid\,(\mathbf{0}, \boldsymbol{Q})$. Comparing (2.4)-(2.5) with (2.1)-(2.2) it is immediate to see that, in this formulation, the errors in the state and observation equations are the same and **C** = **I**. Therefore the innovations model looks like a very restrictive particular case of (2.1)-(2.1). This impression is erroneous as, under weak assumptions, any output vector $\boldsymbol{z}_t$ realized by the SS model (2.1)-(2.2) is also the output of an innovations model (Casals, Sotoca and Jerez, 1999, Theorem 1).

Innovations models are important because, filtering and smoothing algorithms have special convergence properties when applied to SS models with this particular structure, allowing for the implementation of efficient and stable computational procedures (Casals, Jerez and Sotoca, 2000 and 2002). These possibilities are systematically exploited by E^4.

2.2 The THD format.

Internally, E^4 uses the SS formulation for most computations. However, SS models are difficult to store and manipulate through software. For this reason, E^4 manages models using a convenient format called "THD" (THeta-Din). Any THD format specification is composed of two matrices: `theta` and `din`, which contain, respectively, the parameter values and a description of the model dynamic structure. Additionally, the model can be documented by an optional character matrix, `lab`, which contains names for the parameters in `theta`. While some analyses may require editing `theta` and `lab`, most users will never need to manage the matrix `din`.

E[4] provides native support for many standard models, including SS, ARIMA, VARMAX structural econometric models and single-output transfer functions. These models have specialized interface functions which generate `theta`, `din` and `lab`. The most important of these functions is `ss2thd` (SS to THD), which translates any model in the form (2.1)-(2.2) to THD format. Its syntax is:

```
[theta, din, lab] = ss2thd(Phi, Gam, E, H, D, C, Q, S, R);
```

where the input arguments correspond to the parameter matrices in (2.1)-(2.1). The following example illustrates how to define a SS model using this function.

Example 2.1 (The HP filter / local linear trend model). The Hodrick and Prescott (1980) filter implicitly assumes that the series z_t is realized by the local linear trend model (Harvey, 1989):

$$\begin{bmatrix} T_{t+1} \\ \Delta_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} T_t \\ \Delta_t \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_t \tag{2.6}$$

$$z_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} T_t \\ \Delta_t \end{bmatrix} + v_t \tag{2.7}$$

where T_t is an unobserved trend component, Δ_t is the change in the trend component at time *t* and the only unknown parameters are the error variances:

$$\mathrm{E}\left[\begin{pmatrix} w_t \\ v_t \end{pmatrix} \begin{pmatrix} w_t & v_t \end{pmatrix} \right] = \begin{bmatrix} \sigma_w^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix} \tag{2.8}$$

Finally, for quarterly data, it is typically assumed that $\frac{\sigma_w^2}{\sigma_v^2} = \frac{1}{1600}$. This model can be defined in E[4] using the following code:

```
e4init  % Initializes E4 default options and tolerances

% *** Defines the model matrices
Phi= [1 1; NaN 1]; % The not-a-number value (NaN) indicates a null parameter
```

```
E= [1;NaN];
H=[1 0]; C=[1]; Q=[1/1600]; S=[0]; R=[1];

% *** Obtains the THD representation and displays the resulting model
[theta, din, lab] = ss2thd(Phi, [], E, H, [], C, Q, S, R);

% The only free parameters in this model are the variances. Accordingly,
%    nonzero values in the second column of theta indicate which parameters
%    should remain at its current value
theta(1,2)=1; theta(2,2)=1; theta(3,2)=1;
theta(4,2)=1; theta(5,2)=1; theta(6,2)=1;
theta(7,2)=1; theta(9,2)=1;

prtmod(theta, din, lab);
```

and the resulting output is:

```
********************** Options set by user ********************
Filter. . . . . . . . . . . . . : KALMAN
Scaled B and M matrices . . . . : NO
Initial state vector. . . . . . : AUTOMATIC SELECTION
Initial covariance of state v.  : IDEJONG
¿Variance or Cholesky factor? . : VARIANCE
Optimization algorithm. . . . . : BFGS
Maximum step length . . . . . . : 0.100000
Stop tolerance. . . . . . . . . : 0.000010
Max. number of iterations . . . :       75
Verbose iterations. . . . . . . : YES
***************************************************************

************************** Model **************************
Native SS model
1 endogenous v., 0 exogenous v.
Seasonality: 1
SS vector dimension: 2
Parameters (* denotes constrained parameter):
PHI(1,1)     *     1.0000
PHI(1,2)     *     1.0000
PHI(2,2)     *     1.0000
E(1,1)       *     1.0000
H(1,1)       *     1.0000
H(1,2)       *     0.0000
C(1,1)       *     1.0000
Q(1,1)             0.0006
S(1,1)       *     0.0000
R(1,1)             1.0000
***************************************************************
```

Note that asterisks denote which parameters should remain at its current values in any subsequent estimation.

2.3 Simple models.

Obtaining manually the SS representation of a standard model is tedious. Defining it in `theta-din-lab` format would be even worst. To alleviate these tasks, the E^4 functions `arma2thd`, `str2thd` and `tf2thd` generate the THD formulation for VARMAX, structural econometric models and transfer functions, respectively (Appendix A defines the precise mathematical formulation of the models supported). The general syntax of these functions is:

```
[theta, din, lab] = arma2thd(FR, FS, AR, AS, V, s, G, r)

[theta, din, lab] = str2thd(FR, FS, AR, AS, V, s, G, r)

[theta, din, lab] = tf2thd(FR, FS, AR, AS, V, s, W, D)
```

where the inputs to these functions are matrices of real numbers containing: (a) the initial values for the regular and seasonal autoregressive factors, `FR` and `FS`, (b) the regular and seasonal moving average factors, `AR` and `AS`, (c) the terms related to the exogenous variables in the transfer function model, `W` and `D`, and in the VARMAX and structural econometric model, `G`, (d) the noise covariance, `V`, as well as (e) scalar integers defining the seasonal period, `s`, and the number of inputs in the VARMAX and structural econometric models, `r`.

Example 2.2 (Defining an airline model). The following code defines a quarterly airline model in THD form, displays it and obtains the matrices of its SS representation:

```
e4init % Initializes E4 default options and tolerances

% Defines and displays the non-stationary airline model
%    (1-B)(1-B^4) y_t = (1 - .6 B)(1 - .5 B^4) a_t
[theta,din,lab] = arma2thd([-1],[-1],[-.6],[-.5],[.1],4);
theta(1,2)=1; theta(2,2)=1; % Constrains the AR parameters
prtmod(theta,din,lab);

% Now obtains the matrices of the equivalent SS representation
[Phi, Gam, E, H, D, C, Q, S, R] = thd2ss(theta, din)
```

and the resulting output is:

```
************************** Model **************************
VARMAX model (innovations model)
1 endogenous v., 0 exogenous v.
Seasonality: 4
SS vector dimension: 5
Parameters (* denotes constrained parameter):
FR1(1,1)      *    -1.0000
FS1(1,1)      *    -1.0000
AR1(1,1)           -0.6000
AS1(1,1)           -0.5000
V(1,1)              0.1000
***********************************************************

Phi =
     1     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     1     0     0     0     1
    -1     0     0     0     0

Gam =
     []

E =
    0.4000
         0
         0
    0.5000
   -0.7000

H =
     1     0     0     0     0

D =
     []

C =
     1

Q =
    0.1000

S =
    0.1000

R =
    0.1000
```

Finally, E^4 also includes functions that obtain the matrices characterizing a simple model from its THD formulation. The names of these functions are `thd2ss`, `thd2arma`, `thd2str` and `thd2tf`. For further details see the User Manual (Terceiro *et al.*, 2000).

2.4 Models with GARCH errors.

To define a model with GARCH errors in THD format it is necessary to obtain: (a) the THD formulation of the model for the mean, (b) the THD model corresponding to the model (ARCH, GARCH or IGARCH) for the variance, and (c) link both formulations using the `garc2thd` function, which has the following syntax:

```
[theta, din, lab] = garc2thd(t1, d1, t2, d2, lab1, lab2);
```

where `t1` and `d1` correspond to the model for the mean, `t2` and `d2` to the model for the variance, and `lab1` and `lab2` are optional labels for the parameters in `t1` and `t2`, respectively.

Example 2.3 (Defining a model with GARCH errors). The following code obtains and displays the THD representation of a regression model with GARCH errors:

```
e4init % Initializes E4 default options and tolerances

% Defines and displays a regression model with GARCH(1,1) errors:
%   y_t = .7 x_t1 + 1 x_t2 + a_t
%   a_t^2 = .1 + N_t
%   (1 - .8 B) N_t =  (1 - .7 B) v_t
[t1, d1, lab1] = arma2thd([], [], [], [], [.1], 1,[.7 1],2);
[t2, d2, lab2] = arma2thd([-.8], [], [-.7], [], [.1], 1);
[theta, din, lab] = garc2thd(t1, d1, t2, d2, lab1, lab2);
prtmod(theta, din, lab);
```

and the resulting output is:

```
************************** Model ****************************
GARCH model (innovations model)
1 endogenous v., 2 exogenous v.
Seasonality: 1
SS vector dimension: 0
Endogenous variables model:
  White noise model (innovations model)
  1 endogenous v., 2 exogenous v.
  Seasonality: 1
  SS vector dimension: 0
  Parameters (* denotes constrained parameter):
```

```
  G0(1,1)             0.7000
  G0(1,2)             1.0000
  V(1,1)              0.1000
  --------------
GARCH model of noise:
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 1
  Parameters (* denotes constrained parameter):
  FR1(1,1)           -0.8000
  AR1(1,1)           -0.7000
  --------------
************************************************************
```

2.5 Nested models.

A model that combines the formulation of several models is called a "nested" model. E^4 allows for two types of model nesting: *nesting in inputs* and *nesting in errors*.

Nesting in inputs consists in augmenting a SS model with the models for its exogenous variables. After doing so, the inputs become endogenous variables. This operation is useful, for example, when one wants to combine a transfer function and a VARMAX model for its inputs to compute forecasts for all these variables in a single operation. Another situation where nesting is useful is when there are missing values in the inputs. "Endogeneizing" is then required because E^4 (and SS methods in general) only allow for missing values in the endogenous variables.

On the other hand, nesting in errors consists in augmenting an innovations model with a model for its errors. Intuitively, it can be seen as defining a model by the product of different polynomials. This is handy, e.g., when one wants to separate the factors containing different types of roots, such as unit, complex or real roots. It is also useful when modeling a series with several seasonal factors, corresponding to different periods.

The mathematical definition of these operations is described in Appendix B. The following examples illustrate how E^4 implements the definition of these models.

Example 2.4 (Defining a nested-in-inputs model). Given the models:

$$z_t = \frac{.3+.6\mathrm{B}}{1-.5\mathrm{B}}\,u_{1,t} + \frac{1-.8\,\mathrm{B}}{1-.6\,\mathrm{B}+.4\,\mathrm{B}^2}\,a_{1,t}\;;\;\sigma_1^2 = 1 \tag{2.9}$$

$$\left(1-.7\,\mathrm{B}\right)u_{1,t} = a_{2,t}\;;\;\sigma_2^2 = .3 \tag{2.10}$$

… the code required to formulate a single nested-in-inputs model is:

```
e4init

% Obtains the THD representation for both models:
[t1,d1,l1]=tf2thd([-.6 .4],[],[-.8],[],[1.0],1,[ .3 .6],[-.5]);
[t2,d2,l2]=arma2thd(-.7,[],[],[],.3,1);

% Combines the models for the endogenous variable and the input
[theta, din, lab] = stackthd(t1, d1, t2, d2, l1, l2);
[theta, din, lab] = nest2thd(theta, din, 1, lab);

% … and displays the resulting model structure
prtmod(theta,din,lab);
```

… and the output from `prtmod` is:

```
************************** Model ***************************
Nested model in inputs (innovations model)
2 endogenous v., 0 exogenous v.
Seasonality: 1
SS vector dimension: 4
Submodels:
{
  Transfer function model (innovations model)
  1 endogenous v., 1 exogenous v.
  Seasonality: 1
  SS vector dimension: 3
  Parameters (* denotes constrained parameter):
  FR(1,1)           -0.6000
  FR(1,2)            0.4000
  AR(1,1)           -0.8000
  W1(1,1)            0.3000
  W1(2,1)            0.6000
  D1(1,1)           -0.5000
  V(1,1)             1.0000
  --------------
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
```

```
  SS vector dimension: 1
  Parameters (* denotes constrained parameter):
  FR1(1,1)          -0.7000
  V(1,1)             0.3000
  --------------
}
************************************************************
```

Note that the nested model has two endogenous variables z_t and $u_{1,t}$, and no inputs.

Example 2.5 (Defining a nested-in-errors model). Consider the model:

$$\left(1-.5\mathrm{B}+.3\mathrm{B}^2\right)\left(1-\mathrm{B}\right)\left(1-\mathrm{B}^{12}\right)z_t = a_t\ ;\ \sigma_a^2 = .2 \tag{2.11}$$

… which has nonstationary regular and seasonal AR factors. The following code defines and displays model (2.11) through nesting in errors.

```
e4init
% First obtains the THD representation of the factors
[t1, d1, l1] = arma2thd([-1], [-1], [], [], [0], 12);
[t2, d2, l2] = arma2thd([-.5 .3], [], [], [],[.2], 12);

% … and then the stacked and nested models
[ts, ds, ls] = stackthd(t1, d1, t2, d2, l1, l2);
[theta, din, lab] = nest2thd(ts, ds, 0, ls);
theta(1,2)=1; theta(2,2)=1; % Constrains the unit roots
prtmod(theta, din, lab);
% Note that the variance of model t1-d1 is ignored
```

… and the output from `prtmod` is:

```
************************** Model **************************
Nested model in errors (innovations model)
1 endogenous v., 0 exogenous v.
Seasonality: 12
SS vector dimension: 15
Submodels:
{
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 12
  SS vector dimension: 13
  Parameters (* denotes constrained parameter):
  FR1(1,1)     *    -1.0000
  FS1(1,1)     *    -1.0000
  --------------
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
```

```
  Seasonality: 12
  SS vector dimension: 2
  Parameters (* denotes constrained parameter):
  FR1(1,1)          -0.5000
  FR2(1,1)           0.3000
  V(1,1)             0.2000
  --------------
}
************************************************************
```

2.6 Component models.

A component model is defined as the sum of several stochastic processes. Two important examples are unobserved component models (STSM), see Harvey (1989) and models with observation errors, see Terceiro (1990).

The process to define a component model is very similar to that of nested models, but replacing the final call to `nest2thd` for a similar call to `comp2thd`.

Example 2.6. (Defining the HP filter model as a component model). The HP filter model already defined in Example 2.1 can also be seen as an integrated random-walk with a white noise observation error:

$$\begin{aligned} &\left(1-\mathrm{B}\right)^2 T_t = w_t \;;\; w_t \sim iid\left(0,1/1600\right) \\ &z_t = T_t + v_t \;;\; v_t \sim iid\left(0,1\right) \end{aligned} \tag{2.12}$$

The corresponding THD format can be obtained with the code:

```
e4init
% Defines the implicit model for the HP quarterly filter as an
% integrated random walk with a white noise observation error

% First, obtains the THD representation of the components
[t1, d1, l1] = arma2thd([-2 1], [], [], [], 1/1600, 1);
[t2, d2, l2] = arma2thd([], [], [], [], 1, 1);
% … and then the stacked and nested models
[ts, ds, ls] = stackthd(t1, d1, t2, d2, l1, l2);
[theta, din, lab] = comp2thd(ts, ds, ls);
theta(1,2)=1; theta(2,2)=1; % Constrains the unit roots
prtmod(theta, din, lab);
```

which yields the following output:

```
************************** Model **************************
Components model
1 endogenous v., 0 exogenous v.
Seasonality: 1
SS vector dimension: 2
Submodels:
{
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 2
  Parameters (* denotes constrained parameter):
  FR1(1,1)     *    -2.0000
  FR2(1,1)     *     1.0000
  V(1,1)             0.0006
  --------------
  White noise model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 0
  Parameters (* denotes constrained parameter):
  V(1,1)             1.0000
  --------------
}
***********************************************************
```

3. Procedures.

E^4 includes three groups of functions: (a) standard procedures for time series analysis and model validation, (b) signal extraction methods, and (c) model estimation algorithms.

3.1 Standard procedures for time series analysis.

The main standard procedures included are detailed in Table 1. These functions cover unexceptional (but important) needs such as plotting a time series, computing its autocorrelations or simulating a realization from a given model structure. They are included to make the Toolbox self-contained, but have no original functionality. Therefore, there are good alternatives available such as the analogous code in LeSage Toolbox (LeSage, 1999).

[Insert Table 1]

The following example illustrates the use of these standard functions using simulated data. We will not show the outputs resulting from this and the simulation-based examples because the exact results obtained would differ from the ones shown here.

Example 3.1 (Univariate analysis): Consider the ARMA(2,1) model:

$$\left(1-.5\mathrm{B}+.3\mathrm{B}^2\right)z_t = \left(1-.7\mathrm{B}\right)a_t \; ; \sigma_a^2 = .2 \tag{3.1}$$

The following code defines it, simulates a sample and performs a standard univariate identification process:

```
e4init
% Obtains the THD definition of the model
[theta, din, lab] = arma2thd([-.5 .3], [], [-.7], [], [.2], 1);
prtmod(theta,din,lab)
```

```
% Simulates the sample and omits the first observations
z=simmod(theta,din,250); z=z(51:250,1);

% Applies several standard time series analysis tools
% Standard-deviation mean plot
rmedser(z);
% Time series plot
plotsers(z);
% Augmented Dickey-Fuller test
augdft(z, 2);
% Histogram
histsers(z);
% Descriptive statistics
descser(z);
% Sample autocorrelation and partial autocorrelation functions
uidents(z,10);
```

Example 3.2 (Multiple time series analysis): The following code illustrates the use of the multiple time series analysis functions included with a realization of the VAR(1) model:

$$\begin{bmatrix} -.5 & .3 \\ 0 & -.6 \end{bmatrix}\begin{bmatrix} z_{1,t} \\ z_{2,t} \end{bmatrix} = \begin{bmatrix} a_{1,t} \\ a_{2,t} \end{bmatrix}; \Sigma_a = \begin{bmatrix} 1 & .5 \\ .5 & 1 \end{bmatrix} \tag{3.2}$$

```
e4init
% Obtains the THD definition of the model
Phi=[-.5 .3;NaN -.6];
Sigma=[1 .5;.5 1];
[theta,din,lab]=arma2thd([Phi],[],[],[],[Sigma],1);
prtmod(theta,din,lab)

% Simulates the sample and omits the first observations
y=simmod(theta,din,250); y=y(51:250,:);

% Descriptive statistics for multiple time series
descser(y);

% Multiple autocorrelation, partial autocorrelation and
%    cross-correlation functions
midents(y,10);
```

3.2 Signal extraction methods.

The SS literature (Anderson and Moore, 1979) distinguishes between three main signal extraction problems: filtering, forecasting and smoothing. All of them focus on estimating the sequence of states, considering in each case different information sets.

Consider a SS model, such as (2.1)-(2.2) or (2.4)-(2.5), and denote:

$$\boldsymbol{x}_{t|j} = E\left(\boldsymbol{x}_t \left| \boldsymbol{\Omega}_n \right.\right) ; \boldsymbol{P}_{t|j} = E\left[(\boldsymbol{x}_t - \boldsymbol{x}_{t|j})(\boldsymbol{x}_t - \boldsymbol{x}_{t|j})^T \left| \boldsymbol{\Omega}_n \right.\right] \tag{3.3}$$

where $\boldsymbol{\Omega}_n = \{z_1, \ldots, z_n\}$ is the information set containing all the observable measures up to time *n*. In these conditions, the sequences of filtered, forecasted and fixed-interval smoothed states are, respectively:

$$\boldsymbol{x}_{t+1|t} ; \boldsymbol{P}_{t+1|t} \quad (t = 1, 2, \ldots, n) \tag{3.4}$$

$$\boldsymbol{x}_{t|n} ; \boldsymbol{P}_{t|n} \quad (t = n+1, n+2, \ldots) \tag{3.5}$$

$$\boldsymbol{x}_{t|n} ; \boldsymbol{P}_{t|n} \quad (t = 1, 2, \ldots, n) \tag{3.6}$$

E[4] computes these sequences for different purposes. Filtered states like (3.4) are typically used to compute the Gaussian likelihood in prediction-error decomposition form (Terceiro, 1990, Chapter 4), so the filtering algorithms in E[4] are embedded in the likelihood evaluation functions, see Section 3.3. The forecasting sequences (3.5) are useful to predict future values of the endogenous variables, z_t, and they can be computed using the E[4] functions `foremod` and `foregarc`. Finally, the smoothed estimates in (3.6) have many uses, such as interpolating missing in-sample values (Kohn and Ansley 1986), calculating the residuals of a model allowing for missing values (Kohn and Ansley, 1989), "cleaning" noise-contaminated samples (Kohn and Ansley 1987), decomposing a time series into meaningful unobserved components (Harvey, 1989; Casals, Jerez and Sotoca 2002) and detecting outliers (De Jong and Penzer, 1998). Table 2 summarizes the main E[4] functions implementing signal-extraction algorithms, with the corresponding academic references where relevant.

[Insert Table 2]

Basic fixed-interval smoothing, hereafter FIS, is addressed by the E^4 function `fismod`, which implements the algorithm of Casals, Jerez and Sotoca (2000). Its syntax is:

```
[xhat, Px, e] = fismod(theta, din, z);
```

The input arguments of these functions are a THD format specification (`theta, din`) and the data matrix (`z`) where the missing values, if any, are coded as `NaN`, which is the not-a-number MATLAB special value. The output arguments of `fismod` are `xhat`, the sequence of expectations of the state vector conditional to the sample; `Px`, a matrix containing the corresponding covariances; and `e`, a matrix of smoothed errors. There is another FIS function, `fismiss`, which allows for missing values in `z`.

Smoothing is often used to decompose a time series into the sum of trend, cycle, seasonal and irregular components. The function `e4trend` implements the exact decomposition described in Casals, Jerez and Sotoca (2002). Its simplified syntax is:

```
[trend,season,cycle,irreg] = e4trend(theta,din,z)
```

where the input arguments `theta`, `din` and `z` are identical to those of `fismod`, and the output arguments are smoothed estimates of the corresponding structural components.

The function `foremod` predicts future values of the endogenous variables for a given model. Its syntax is:

```
[zf, Bf] = foremod(theta, din, z, k, u)
```

where the input arguments `theta`, `din` and `z` are identical to those of `fismod`, `k` is the number of forecasts to be computed and `u` is a matrix with the out-of-sample values of the exogenous variables, if any. The forecasts and the corresponding sequence of covariances are returned in the output arguments `zf`, `Bf`. The analogous function `foregarc` computes forecasts for conditional heteroscedastic models.

Model residuals are calculated through the function `residual`. Its syntax is:

```
[z1, vT, wT, vz1, vvT, vwT] = residual(theta, din, z)
```

where the input arguments `theta`, `din` and `z` are identical to those of `fismod`. On the other hand, the output arguments are `z1`, `vT` and `wT` which are, respectively, the model residuals, the FIS estimates of the observation errors and the FIS estimates of the state errors. The arguments `vz1`, `vvT` and `vwT` store the corresponding sequences of covariances.

Example 3.3 (HP filtering). The following code simulates the HP filter model defined in Example 2.1 and extracts the HP trend using different procedures:

```
e4init
% Working with standard deviations improves the model scaling
sete4opt('var','fac');

Phi= [1 1; NaN 1];
E= [1;NaN];
H=[1 0]; C=[1]; Q=[sqrt(1/1600)]; S=[0]; R=[1];
% Obtains the THD representation and displays the model
[theta, din, lab] = ss2thd(Phi, [], E, H, [], C, Q, S, R);
prtmod(theta, din, lab);
% Simulates the data and omits the first observations
y=simmod(theta,din,250); y=y(51:250,:);

% Extracts and plots the trend and error components
[trend,season,cycle,irreg] = e4trend(theta,din,y);
plotsers([trend,irreg]);

% e4trend does not provide variances for the trend component:
%    if needed, they can be obtained using fismod
[xhat, px, e] = fismod(theta, din, y); varhptrend=px(1:2:400,1);
figure; hold on
plot(varhptrend.^.5)
```

```
hold off

% Estimates of the HP trend resulting from e4trend and fismod are identical
HPTrendError=sum(xhat(:,1)-trend)
```

3.3 Likelihood and model estimation algorithms.

Table 3 summarizes the functionality of the main functions related with model estimation. These functions specialize in the computation of likelihood values, information matrices and parameter estimates.

[Insert Table 3]

The function `lffast` computes the gaussian log-likelihood of all the homoscedastic models supported by E^4. Its general syntax is:

```
[f, innov, ssvect] = lffast(theta, din, z)
```

Where the input arguments are identical to those defined for previous functions, and the output arguments are: (a) `f`, the value of the log-likelihood, (b) `innov`, a matrix of one-step-ahead forecast errors, and (c) `ssvect`, a matrix of estimates of the state variables organized so that its *t*-th row contains the filtered estimate of the state vector at time *t*, conditional on sample values up to *t*-1. Two variants of this function, `lfmiss` and `lfgarch`, allow for in-sample missing values and GARCH errors, respectively.

As in the case of the likelihood, there are three functions dealing with computation of the information matrix: `imod`, `imiss`, and `igarch`. They all have the same syntax. For example, any call to `imod` has the following structure:

```
[std, corrm, varm, Im] = imod(theta, din, z)
```

This function computes the information matrix of `lfmod` and `lffast` (`Im`) as well as the parameter analytical standard errors (`std`), the correlation matrix of the estimates (`corrm`) and the corresponding covariance matrix (`varm`). The functions `imiss` and `igarch` do the same for `lfmiss` and `lfgarch`, respectively. If the model may be misspecified or is non-Gaussian, the function `imodg` computes a robust information matrix, alternative to `imod`, see Ljung and Caines (1979) and White (1982).

Being able to compute the gaussian log-likelihood is not enough to estimate the parameters of a model: one needs an iterative procedure to compute the optimal value of the parameters. The E^4 function `e4min` implements two main optimization algorithms, BFGS and Newton-Raphson (Dennis and Schnabel, 1983). Its general syntax is:

```
[pnew,iter,fnew,gnew,hessin]=e4min('func',theta,'dfunc',P1,P2,P3,P4,P5)
```

The operation of `e4min` is as follows. Starting from the initial estimate of the parameters in `theta`, it iterates on the objective function which name is stored in the argument `func`. The iteration can be based on the analytical gradient (Terceiro et al. 2000) or on a numerical approximation, depending on whether `dfunc` contains the name of the analytical gradient function or is an empty string, `''`. Finally, the stop criterion takes into account the relative changes in the values of the parameters and/or the size of the gradient vector. The parameters `P1`, ..., `P5` are optional. If they are specified, its values are feed as input arguments to the objective function `func` without modifications.

Once the iterative process is stopped, the function returns the values: `pnew` which is the value of the parameters, `iter`, the number of iterations performed, `fnew`, the value of the objective function at `pnew`, `gnew` which is the analytical or numerical gradient, depending on the contents of `dfunc`, and finally `hessin`, which is a numerical approximation to the hessian.

The auxiliary function `sete4opt` manages different options which affect the optimizer, including the algorithm to use, the tolerance for stop criteria or the maximum number of iterations allowed.

Another important function for model estimation is `e4preest`. It computes fast and consistent estimates of the parameters in `theta` which, in most cases, are adequate starting values for likelihood optimization. The syntax of this function is:

```
thetanew = e4preest(theta, din, z)
```

where the input arguments are identical to those of `lffast` and the estimates are returned in `thetanew`. The algorithm implemented in `e4preest` is described by García-Hiernaux, Jerez and Casals (2008).

Example 3.4 (Simulation, estimation and signal extraction). The following code simulates, estimates and applies different signal-extraction procedures to the nonstationary airline model structure defined in Example 2.2:

```
e4init
% Defines and displays the non-stationary airline model
%    (1-B)(1-B^12) y_t = (1 - .6 B)(1 - .5 B^12) a_t
[theta,din,lab] = arma2thd([-1],[-1],[-.6],[-.5],.1,12);
theta(1,2)=1; theta(2,2)=1;  % Constrains the unit roots

% Simulates the sample and omits the first observations
z=simmod(theta,din,250); z=z(51:250,1);

% Computes preliminary estimates
theta=e4preest(theta,din,z);
% ... and then ML estimates
[thopt, it, lval, g, h] = e4min('lffast', theta,'', din, z);
% Computes the analytical standard errors
[std, corrm, varm, Im] = imod(thopt, din, z);
% Summary of estimation results
prtest(thopt, din, lab, z, it, lval, g, h, std, corrm)

% Residual diagnostics
```

```
ehat=residual(thopt, din, z);
descser(ehat);
plotsers(ehat);
uidents(ehat,39);

% Structural decomposition
[trend,season,cycle,irreg] = e4trend(thopt,din,z);
plotsers([trend,season,irreg]);

% Now computes and displays ten forecasts and their standard errors
[zfor, Bfor] = foremod(thopt, din, z, 10);
[zfor Bfor.^.5]

% Finally, marks two values as missing and estimates them
z1=z; z1(10)=NaN; z1(40)=NaN;
[zhat, pz] = fismiss(thopt, din, z1);
[z(1:50) z1(1:50) zhat(1:50)]
```

4. Examples with real data.

4.1. Univariate modeling of the U.S. single-family housing series.

As an illustrative example, consider the monthly U.S. single-family housing starts (ST_t) and sales (SA_t) series, in thousands of units, from January 1965 through May 1975. This dataset has been analyzed in many works, such as those of Hillmer and Tiao (1979), Ahn and Reinsel (1990) or Tiao (2001). Some stylized facts found in previous analyses are: (a) both series are adequately represented by an airline model, (b) their seasonal structure is probably deterministic, and (c) they are cointegrated. Our own analysis confirmed these facts and also points out that the housing starts series depends on the number of labor days (Monday to Friday) in each month. Accordingly, the following code defines and estimates a transfer function for housing starts versus labor days, performs a standard residual diagnosis and computes its trend and seasonal components.

```
e4init
sete4opt('var','fac');

% Housing.dat
%   Col #1: Housing Starts
%   Col #2: House Sales
%   Col #3: Number of Labor Days (Monday to Friday) in the month
%   Col #4: Number of Saturdays and Sundays in the month
load housing.dat

% *** Housing starts series ***
y=[housing(:,1) housing(:,3)];

% Defines the non-stationary version of the model
[theta, din, lab] = tf2thd([-1],[-1],[0],[0], [0],12,[0],[]);
% The AR parameters, corresponding to unit roots are constrained
theta(1,2) = 1; theta(2,2) = 1;

% Model estimation
theta=e4preest(theta,din,y);
[thopt,it,lval,g,h] = e4min('lffast', theta,'', din, y);
[std,corrm,varm,Im] = imod(thopt, din, y);
prtest(thopt,din,lab,y,it,lval,g,h,std,corrm);

% Residual diagnostics
ehat=residual(thopt,din,y);
plotsers(ehat,0,'house starts residuals');
descser(ehat,'house starts residuals');
uidents(ehat,39,'house starts residuals',12);
```

```
% Extraction of structural components
[trendstarts,seasstarts,cycle,irreg] = e4trend(thopt,din,y);
plotsers([trendstarts,seasstarts],0,['trend component    '; ...
'seasonal component ']);
```

The output from `prtest` is:

```
******************** Results from model estimation ********************
  Objective function: 386.2710
     # of iterations:   22
Information criteria: AIC =   6.2443, SBC =   6.3348

Parameter          Estimate    Std. Dev.       t-test     Gradient
FR(1,1)      *      -1.0000       0.0000       0.0000       0.0000
FS(1,1)      *      -1.0000       0.0000       0.0000       0.0000
AR(1,1)             -0.2623       0.0862      -3.0434       0.0004
AS(1,1)             -1.1718       0.0193     -60.6407      -0.0003
W1(1,1)              1.1383       0.4530       2.5129       0.0000
V(1,1)               5.4559       0.3451      15.8114      -0.0001
* denotes constrained parameter

************************ Correlation matrix *************************
AR(1,1)       1.00
AS(1,1)      -0.00  1.00
W1(1,1)       0.00  0.00  1.00
V(1,1)        0.00  0.00  0.00  1.00

           Condition number =   1.0053
Reciprocal condition number =   0.9947
*********************************************************************
```

… and can be written in transfer function form as:

$$ST_t = \underset{(.45)}{1.14}\, L_t + \hat{N}_t \; ; (1-B)(1-B^{12})\hat{N}_t = \underset{(.09)}{(1-.26B)}\underset{(.02)}{(1-1.17B^{12})}\hat{a}_t^{ST} \; ; \hat{\sigma}_{ST} = 5.46 \qquad (4.1)$$

where L_t denotes the number of labor days (Monday to Friday) in month *t* and the figures in parentheses are the standard errors of the estimates. Note that the estimated seasonal parameter is noninvertible. This happens sometimes because, as it is well known, if θ^* is and invertible optimum of a likelihood function, then the noninvertible root $1/\theta^*$ is also an optimum. This theoretical result can be easily checked with the code:

```
% If required, the following code yields an equivalent invertible model
theta1=thopt; theta1(4,1)=1/thopt(4,1);
[thopt2,it,lval,g,h] = e4min('lffast', theta1,'', din, y);
[std,corrm,varm,Im] = imod(thopt2, din, y);
```

```
prtest(thopt2,din,lab,y,it,lval,g,h,std,corrm);
```

… which output is:

```
******************** Results from model estimation ********************
  Objective function: 386.2710
     # of iterations:   15
Information criteria: AIC =   6.2443, SBC =   6.3348

Parameter          Estimate    Std. Dev.       t-test     Gradient
FR(1,1)      *      -1.0000       0.0000       0.0000       0.0000
FS(1,1)      *      -1.0000       0.0000       0.0000       0.0000
AR(1,1)             -0.2624       0.0865      -3.0314      -0.0037
AS(1,1)             -0.8534       0.0534     -15.9692      -0.0000
W1(1,1)              1.1378       0.4530       2.5117      -0.0025
V(1,1)               6.3930       0.4043      15.8114      -0.0000
* denotes constrained parameter

************************ Correlation matrix ************************
AR(1,1)       1.00
AS(1,1)      -0.00  1.00
W1(1,1)      -0.00 -0.00  1.00
V(1,1)        0.00  0.00 -0.00  1.00

           Condition number =   1.0024
Reciprocal condition number =   0.9976
************************************************************************
```

… note that the likelihood values achieved in this listing and the previous one are identical.

The following code continues this example, by defining and estimating an airline model for housing sales, performing its diagnosis and computing the corresponding trend and seasonal components.

```
% *** Housing sales series ***
y=[housing(:,2)];
% Defines the model.
[theta, din, lab] = arma2thd([-1],[-1],[0],[0], [0],12);
theta(1,2) = 1; theta(2,2) = 1;
% Model estimation
theta=e4preest(theta,din,y);
[thopt,it,lval,g,h] = e4min('lffast', theta,'', din, y);
[std,corrm,varm,Im] = imod(thopt, din, y);
prtest(thopt,din,lab,y,it,lval,g,h,std,corrm);

% Residual diagnostics
ehat=residual(thopt,din,y);
plotsers(ehat,0,'house sales residuals');
descser(ehat,'house sales residuals');
uidents(ehat,39,'house sales residuals',12);
```

```
% Structural components
[trendsales,seassales,cycle,irreg] = e4trend(thopt,din,y);
plotsers([trendsales,seassales,irreg],0,['trend component    '; ...
'seasonal component '; 'irregular component']);
```

The output from `prtest` is:

```
******************** Results from model estimation ********************
  Objective function: 323.6994
     # of iterations:   16
Information criteria: AIC =   5.2272, SBC =   5.2951

Parameter          Estimate    Std. Dev.       t-test     Gradient
FR1(1,1)     *      -1.0000       0.0000       0.0000       0.0000
FS1(1,1)     *      -1.0000       0.0000       0.0000       0.0000
AR1(1,1)            -0.1595       0.0887      -1.7985      -0.0004
AS1(1,1)            -1.0000       0.0412     -24.2893       0.0002
V(1,1)               3.4536       0.2184      15.8114       0.0000
* denotes constrained parameter

************************ Correlation matrix *************************
AR1(1,1)      1.00
AS1(1,1)     -0.00  1.00
V(1,1)        0.00  0.00  1.00

           Condition number =   1.0000
Reciprocal condition number =   1.0000
*********************************************************************
```

and this model, written in standard notation, is:

$$(1-B)(1-B^{12})SA_t = \underset{(.09)}{(1-.16B)}\underset{(.04)}{(1-1.00B^{12})}\hat{a}_t^{SA};\ \hat{\sigma}_{SA} = 3.45 \tag{4.2}$$

Finally, the following code eliminates the seasonal components from both series and compares the resulting seasonally-adjusted values.

```
adjstarts=housing(:,1)-seasstarts;
adjsales=[housing(:,2)]-seassales;
plotsers([adjstarts adjsales],1,['Starts';'Sales ']);
```

Figure 1 shows the output from `plotsers`. Note that both series are remarkably similar, so this informal analysis suggests that there could be a cointegration relationship. Section 4.2. explores this idea with more depth.

[Insert Figure 1]

The code and data required to replicate this example can be found at [**COMPLETE THIS URL**], files `Housing_Univariate.m` and `housing.dat`.

4.2. A nonstandard VARMA model for the housing series.

Allowing for cointegration, the dynamic structure of (4.1)-(4.2) could be written as:

$$\begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix} \begin{bmatrix} ST_t \\ SA_t \end{bmatrix} = \begin{bmatrix} \beta \\ 0 \end{bmatrix} L_t + \begin{bmatrix} N_t^{ST} \\ N_t^{SA} \end{bmatrix} \tag{4.3}$$

$$\begin{bmatrix} 1+\phi_{11}\mathrm{B} & 0 \\ 0 & 1-\mathrm{B} \end{bmatrix} \begin{bmatrix} 1-\mathrm{B}^{12} & 0 \\ 0 & 1-\mathrm{B}^{12} \end{bmatrix} \begin{bmatrix} N_t^{ST} \\ N_t^{SA} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1-\theta_{11}\mathrm{B} \end{bmatrix} \begin{bmatrix} 1+\Theta_{11}\mathrm{B}^{12} & 0 \\ 0 & 1+\Theta_{22}\mathrm{B}^{12} \end{bmatrix} \begin{bmatrix} a_t^{ST} \\ a_t^{SA} \end{bmatrix} \tag{4.4}$$

Note that this model allows the linear combination $ST_t + \alpha \cdot SA_t$ to have one unit root less than SA_t. On the other hand if the series were not cointegrated, the estimate for the parameter ϕ_{11} should converge to -1. The following code shows how to define (4.3)-(4.4) as a nested model, which is afterwards estimated and diagnosed.

```
e4init
sete4opt('var','fac');

% Housing.dat
%   Col #1: Housing Starts
%   Col #2: House Sales
%   Col #3: Number of Labor Days (Monday to Friday) in the month
%   Col #4: Number of Saturdays and Sundays in the month
load housing.dat
y=[housing(:,1:3)];

% First sub-model (4.3)
[theta1,din1,lab1]=str2thd([1 0; NaN 1],[],[],[],zeros(2),12,[0;NaN],1);

% Second sub-model (4.4)
F1=[0 NaN; NaN -1]; A1=[NaN NaN;NaN 0];
FS1=[-1 NaN; NaN -1]; AS1=[0 NaN; NaN 0];
SIG=[0 0;0 0];
[theta2,din2,lab2]=arma2thd(F1,FS1,A1,AS1,SIG,12,[NaN;NaN],1);
% Nesting requires both models to have the same number of inputs.
```

```
%    Because of this, the second sub-model have been defined with one input
%    with zero-constrained coefficients

% Model nesting
[theta0,din0,lab0]=stackthd(theta1,din1,theta2,din2,lab1,lab2);
[theta,din,lab]=nest2thd(theta0,din0,0,lab0);
theta(1,2)=1;theta(3,2)=1;
theta(6,2)=1;theta(7,2)=1;theta(8,2)=1;

% Estimation
theta=e4preest(theta,din,y);
[thopt,it,f,g,H]=e4min('lffast',theta,'',din,y);
[dts,corrm,varm,Im]=imod(thopt,din,y);
prtest(thopt,din,lab,y,it,f,g,H,dts,corrm);

% Residual diagnostics
ehat=residual(thopt,din,y);
plotsers(ehat,-1,['starts res.';'sales res. ']);
descser(ehat,['starts res.';'sales res. ']);
midents(ehat,10,['starts res.';'sales res. ']);
```

The parameter estimates, as displayed by `prtmod`, are

```
******************** Results from model estimation ********************
  Objective function: 689.1503
     # of iterations:   53
Information criteria: AIC =  11.1704, SBC =  11.3740

Parameter          Estimate    Std. Dev.       t-test     Gradient
FR0(1,1)     *       1.0000       0.0000       0.0000       0.0000
FR0(1,2)            -1.8198       0.0214     -84.9672       0.0005
FR0(2,2)     *       1.0000       0.0000       0.0000       0.0000
G0(1,1)              1.1049       0.4447       2.4848       0.0001
FR1(1,1)            -0.3730       0.0680      -5.4885      -0.0011
FR1(2,2)     *      -1.0000       0.0000       0.0000       0.0000
FS1(1,1)     *      -1.0000       0.0000       0.0000       0.0000
FS1(2,2)     *      -1.0000       0.0000       0.0000       0.0000
AR1(2,2)            -0.1351       0.0728      -1.8546       0.0002
AS1(1,1)            -1.0630       0.0266     -40.0298       0.0004
AS1(2,2)            -0.9741       0.0346     -28.1174      -0.0006
V(1,1)               6.4331       0.4091      15.7253      -0.0000
V(2,1)              -2.3707       0.2730      -8.6852      -0.0001
V(2,2)               2.5607       0.1630      15.7094       0.0004
* denotes constrained parameter

************************ Correlation matrix ************************
FR0(1,2)      1.00
G0(1,1)      -0.00  1.00
FR1(1,1)      0.06 -0.00  1.00
AR1(2,2)      0.03  0.00 -0.45  1.00
AS1(1,1)     -0.14  0.00 -0.01 -0.00  1.00
AS1(2,2)      0.10 -0.00  0.01  0.01  0.49  1.00
V(1,1)       -0.13  0.00 -0.00 -0.01  0.02 -0.03  1.00
V(2,1)        0.09 -0.00  0.01  0.00 -0.01  0.05 -0.55  1.00
V(2,2)        0.13 -0.00  0.01  0.00 -0.01  0.03 -0.01  0.01  1.00

           Condition number =   3.6759
Reciprocal condition number =   0.2399
************************************************************************
```

… so the estimated model can be written as:

$$\begin{bmatrix} 1 & \underset{(.02)}{-1.82} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} ST_t \\ SA_t \end{bmatrix} = \begin{bmatrix} \underset{(.45)}{1.11} \\ 0 \end{bmatrix} L_t + \begin{bmatrix} \hat{N}_t^{ST} \\ \hat{N}_t^{SA} \end{bmatrix} \tag{4.5}$$

$$\begin{bmatrix} \underset{(.07)}{1-.37\mathrm{B}} & 0 \\ 0 & 1-\mathrm{B} \end{bmatrix} \begin{bmatrix} 1-\mathrm{B}^{12} & 0 \\ 0 & 1-\mathrm{B}^{12} \end{bmatrix} \begin{bmatrix} \hat{N}_t^{ST} \\ \hat{N}_t^{SA} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \underset{(.07)}{1-.14\mathrm{B}} \end{bmatrix} \begin{bmatrix} \underset{(.03)}{1-1.06\mathrm{B}^{12}} & 0 \\ 0 & \underset{(.03)}{1-.97\mathrm{B}^{12}} \end{bmatrix} \begin{bmatrix} \hat{a}_t^{ST} \\ \hat{a}_t^{SA} \end{bmatrix} \tag{4.6}$$

Note that the the autoregressive parameter (`FR1(1,1)`) is safely far from unity and the cointegrating parameter (`FR0(1,2)`) is highly significant. Therefore, these results confirm that both series have a common nonstationary component. Finally no additional parameters seem necessary, as the output from `midents` shows that residual auto and cross-correlations are insignificant.

```
******************** Cross correlation functions ********************
            starts res  sales res.
starts res  ..........  ..........
sales res.  ..........  ..........
Each row is the cross correlation function when the column variable leads
the row variable.

Ljung-Box Q statistic for previous cross-correlations
            starts res  sales res.
starts res   12.93        6.47
sales res.   12.17        5.67

Summary in terms of +.-
   For MACF std. deviations are computed as 1/T^0.5 =   0.09
   For MPARF std. deviations are computed from VAR(k) model
************************************************************************
```

The code and data required to replicate this example can be found at [**COMPLETE THIS URL**], files `Housing_Multivariate.m` and `housing.dat`.

4.3. A model with vector-GARCH errors for two exchange rate series.

This example focuses in measuring the short-run comovements between the spot bid exchange rates of Deustche Mark (DM) and Japanese Yen (JY) against US Dollar, observed in

the London Market during 695 weeks, from January 1985 to April 1998. The data has been logged, differenced and scaled by a factor of 100, to obtain the corresponding log percent yields. Excess returns are then computed by substracting the sample mean.

A previous univariate analysis suggests that: (a) both series are stationary, (b) there may be a short autoregressive structure in the mean and (c) some conditional heteroscedasticity in the variance. We will model these features using a VAR(1) model for the mean and a diagonal GARCH(1,1) model for the variance. The following code defines and estimates this model:

```
e4init
load DMJY.dat
% DMJY.dat
%   Col #1: daily spot bid exchange rate of Deustche Mark against US Dollar
%   Col #2: daily spot bid exchange rate of Japanese Yen against US Dollar

% Transforms the data and computes the excess returns
y=transdif(DMJY,0,1)*100;
z(:,1)=y(:,1)-mean(y(:,1));
z(:,2)=y(:,2)-mean(y(:,2));

% Defines a diagonal VAR(1) for the mean
phi=[0 NaN;NaN 0];
sig=zeros(2);
[thetay,diny,laby]=arma2thd([phi],[],[],[],[sig],1);

% ... and a diagonal VGARCH(1,1) for the variance
phie1=[0 NaN NaN;NaN 0 NaN;NaN NaN 0];
thee1=[0 NaN NaN;NaN 0 NaN;NaN NaN 0];
[thetae, dine, labe]=arma2thd([phie1],[],[thee1],[],[zeros(3)],1);

% Combines both models, pre-estimates the parameters
[theta,din,lab]=garc2thd(thetay,diny,thetae,dine,laby,labe);
theta=e4preest(theta,din,z);

%... and then computes pseudo-ML estimates
[thopt,it,l,g,H]=e4min('lfgarch',theta,'', din,z);
[std,corrm,varm,Im]=igarch(thopt,din,z);
prtest(thopt,din,lab,z,it,l,g,H,std,corrm);
```

The resulting output is:

```
******************** Results from model estimation ********************
  Objective function: 2082.5467
     # of iterations:   49
Information criteria: AIC =   6.0333, SBC =   6.1053

Parameter          Estimate    Std. Dev.       t-test     Gradient
FR1(1,1)            -0.2444       0.0318      -7.6905       0.0016
```

```
FR1(2,2)            -0.3002       0.0340      -8.8158      -0.0006
V(1,1)               1.8096       0.2820       6.4166       0.0000
V(2,1)               1.1445       0.1553       7.3700       0.0003
V(2,2)               1.6656       0.1708       9.7519      -0.0001
FR1(1,1)            -0.9729       0.0120     -80.8656       0.0007
FR1(2,2)            -0.9420       0.0153     -61.4178       0.0056
FR1(3,3)            -0.8609       0.0350     -24.5948      -0.0030
AR1(1,1)            -0.9158       0.0206     -44.4318      -0.0006
AR1(2,2)            -0.8726       0.0239     -36.5462      -0.0060
AR1(3,3)            -0.7390       0.0468     -15.7981       0.0027

************************ Correlation matrix *************************
FR1(1,1)      1.00
FR1(2,2)      0.49  1.00
V(1,1)        0.02  0.01  1.00
V(2,1)        0.02  0.02  0.78  1.00
V(2,2)        0.01  0.02  0.44  0.77  1.00
FR1(1,1)     -0.03 -0.02 -0.27 -0.05 -0.01  1.00
FR1(2,2)     -0.04 -0.05 -0.22 -0.11 -0.10  0.57  1.00
FR1(3,3)     -0.02 -0.05 -0.18 -0.25 -0.29  0.18  0.57  1.00
AR1(1,1)     -0.02 -0.01  0.07  0.16  0.15  0.79  0.37 -0.00  1.00
AR1(2,2)     -0.03 -0.04  0.03  0.14  0.19  0.55  0.80  0.24  0.63  1.00
AR1(3,3)     -0.01 -0.04 -0.09 -0.08  0.03  0.24  0.62  0.85  0.16  0.51  1.00

           Condition number = 153.6101
Reciprocal condition number =   0.0040
*********************************************************************
```

… and the estimated model can be written as:

$$\begin{bmatrix} 1-.24\,\mathrm{B} & 0 \\ (.03) & \\ 0 & 1-.30\,\mathrm{B} \\ & (.03) \end{bmatrix} \begin{bmatrix} dm_t \\ yen_t \end{bmatrix} = \begin{bmatrix} \hat{\varepsilon}_t^{dm} \\ \hat{\varepsilon}_t^{yen} \end{bmatrix} \tag{4.7}$$

$$\begin{bmatrix} \left(\hat{\varepsilon}_t^{dm}\right)^2 \\ \hat{\varepsilon}_t^{dm} \cdot \hat{\varepsilon}_t^{yen} \\ \left(\hat{\varepsilon}_t^{yen}\right)^2 \end{bmatrix} = \begin{bmatrix} 1.81 \\ (.28) \\ 1.14 \\ (.16) \\ 1.66 \\ (.17) \end{bmatrix} + \begin{bmatrix} 1-.97\,\mathrm{B} & 0 & 0 \\ (.012) & & \\ 0 & 1-.94\,\mathrm{B} & 0 \\ & (.015) & \\ 0 & 0 & 1-.86\,\mathrm{B} \\ & & (.035) \end{bmatrix}^{-1} \begin{bmatrix} 1-.92\,\mathrm{B} & 0 & 0 \\ (.021) & & \\ 0 & 1-.87\,\mathrm{B} & 0 \\ & (.024) & \\ 0 & 0 & 1-.74\,\mathrm{B} \\ & & (.047) \end{bmatrix} \begin{bmatrix} \hat{v}_t^1 \\ \hat{v}_t^2 \\ \hat{v}_t^3 \end{bmatrix} \tag{4.8}$$

where dm_t and yen_t denote, respectively, the excess returns of the Deutsche Mark and the Japanese Yen. Note that (4.8) is a vector GARCH model in VARMAX form, see Appendix A, Eq. (A.16)-(A.21). Finally, the following code computes the standardized residuals of this model, as well as conditional volatilities and conditional correlations:

```
[e2s,vT,wT,ve2] = residual(thopt,din,z,1);
var1=ve2(1:2:size(ve2,1),1);
var2=ve2(2:2:size(ve2,1),2);
```

```
cov12=ve2(2:2:size(ve2,1),1);
corr=cov12./(sqrt(var1).*sqrt(var2));

figure; hold on; axis([1 700 0 1]);
plot(corr,'k-');
hold off
```

Figure 2 shows the resulting sequence of conditional correlations. Note that the time-varying correlation fluctuates around its unconditional value (.64) with highs and lows of .9-.3 respectively. A high value of the conditional correlation means that the comovement of both series is strong in this period. On the contrary, low conditional correlations indicate periods when both currencies fluctuate independently.

[Insert Figure 2]

The code and data required to replicate this example can be found at [**COMPLETE THIS URL**], files `DMJY.m` and `DMJY.dat`. About models with GARCH errors, it is relevant to be aware that pseudo-likelihood estimation procedures are intrinsically unstable by the reasons explained by Jerez, Casals and Sotoca (2007).

5. Concluding remarks.

This article describes broadly the capacities and use of a MATLAB Toolbox for Time Series Analysis called E^4. Its main differential characteristic is that all the internal computations are done using the equivalent SS representation of the models considered. However, it includes native support for many standard time series formulations, so most users will never need to interact with the inner layer of SS methods.

This software includes powerful functions for the statistical analysis of time series. They result from recent research and, as a consequence, are accurate, sophisticated and have nonstandard features such as: the ability to combine different models, to estimate nonstationary models without differencing the data and to model samples with missing values or measurement errors.

E^4 can be freely downloaded from www.ucm.es/info/icae/e4. Besides the source code of the toolbox, this WEB also provides: (a) a complete Reference Manual (Terceiro *et al.*, 2000) including a detailed description of all its public functions and many examples; (b) source code and data for several applications and analyses, including the examples in the Reference Manual and those in Section 4 of this paper; (c) technical papers including this one, either published or in draft, describing the operating details of many functions; and (d) a "Beta Test Corner", where we post useful documentation and code that is not yet fully integrated in E^4. This Beta code offers advanced functionality for estimating time-varying parameter regressions or modeling and disaggregating time series sampled at different frequencies. If any user of our software considers that an acknowledgment is due, it will be enough to cite this paper.

The legal terms and conditions for using, copying, modifying and distributing E^4 are those defined in the GNU General Public License. According with this free software philosophy, we deliberately avoided interesting options that could reduce portability of our code, such as

MATLAB object-oriented extensions or its graphical interface facilities. Users are not only allowed, but actively encouraged, to implement E^4 in other environments such as Octave (www.gnu.org/software/octave), Scilab (www.scilab.org) or GNU R (www.r-project.org).

References.

Ahn, S.K. and G.C. Reinsel (1990). "Estimation for Partially Nonstationary Multivariate Autoregressive Models," *Journal of the American Statistical Association*, 85, 813-823.

Anderson, B.D.O. and J.B. Moore (1979). *Optimal Filtering*, Prentice-Hall.

Box, G. E. P. and D. R. Cox (1964). "An Analysis of Transformations," *Journal of the Royal Statistical Society*, B, 26, 211-243.

Box, G.E.P., G. M. Jenkins and G.C. Reinsel (1994). *Time Series Analysis, Forecasting and Control*. Prentice-Hall.

Bollerslev, T., R.F. Engle and D.B. Nelson (1994). "ARCH Models," in R.F. Engle and D.L. McFadden (editors), *Handbook of Econometrics*, vol. IV, North-Holland.

Casals, J. and S. Sotoca (1997). "Exact Initial Conditions for Maximum Likelihood Estimation of State Space Models with Stochastic Inputs," *Economics Letters*, 57, 261-267.

Casals, J. and S. Sotoca, S. (2001). "The Exact Likelihood for a State Space Model with Stochastic Inputs," *Computers and Mathematics with Applications* , 42, 199-209.

Casals, J. M. Jerez and S. Sotoca (2000). "Exact Smoothing for Stationary and Nonstationary Time Series," *International Journal of Forecasting*, 16, 59-69.

Casals, J., Jerez, M. and S. Sotoca (2002). "An Exact Multivariate Model-Based Structural Decomposition," *Journal of the American Statistical Association*, 97, 458, 553-564.

Casals, J., Jerez, M. and S. Sotoca (2004). "Modeling and Forecasting Time Series Sampled at Different Frequencies," Mimeo. This paper can be freely downloaded at www.ucm.es/info/ecocuan/mjm

Casals, J. S. Sotoca and M. Jerez (1999). "A Fast and Stable Method to Compute the Likelihood of Time Invariant State-Space Models," *Economics Letters*, 65, 329-337.

De Jong, P. and J.R. Penzer (1998). "Diagnosing shocks in time series," *Journal of the American Statistical Association*, 93, 442, 796–806.

Dennis, J.E. and R.B. Schnabel (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall.

Dickey, D.A. and W.A. Fuller (1981). "Likelihood Ratio Statistics for Autoregressive Time Series with a Unit Root," *Econometrica*, 49, 1057-1072.

Engle, R.F. (1984)." Wald, Likelihood and Lagrange Multiplier Tests in Econometrics," in Z. Griliches and M.D. Intriligator (editors), *Handbook of Econometrics*, vol. II. North-Holland.

García-Hiernaux, A., M. Jerez and J. Casals (2008). "Fast Estimation Methods for Time Series Models in State-Space Form," *Journal of Statistical Computation and Simulation* (forthcoming).

Gómez, V., A. Maravall and D. Peña (1992). "Missing Observations in ARIMA Models: Skipping Approach versus Additive Outlier Approach," *Journal of Econometrics*, 88, 2, 341-363.

Harvey, A. C. (1989). *Forecasting, Structural Time Series Models and the Kalman filter*, Cambridge University Press.

Hillmer, S.C. and G.C. Tiao (1979). "Likelihood Function of Stationary Multiple Autoregressive Moving Average Models," *Journal of the American Statistical Association*, 74, 652-660.

Hodrick, R.J. and E.C. Prescott (1980). "Post-war U.S. Business Cycles," *Carnegie Mellon University Working Paper*.

Jerez, M., J. Casals and S. Sotoca (2007). "Likelihood Stabilization for Ill-conditioned Vector GARCH Models," *Computational Statistics* (forthcoming).

Kohn, R. and C. F. Ansley (1986). "Estimation, Prediction, and Interpolation for ARIMA models with missing data," *Journal of the American Statistical Association*, 81, 395, 751–761.

Kohn, R. and C. F. Ansley (1987). "Signal Extraction for Finite Nonstationary Time Series," *Biometrika*, 74, 2, 411– 421.

Kohn, R. and C. F. Ansley (1989). "A Fast Algorithm for Signal Extraction, Influence and Cross-Validation in State Space Models," *Biometrika* 76, 1, 65–79.

LeSage, J. P. (1999). "Applied Econometrics using MATLAB," (This is part of LeSage's Toolbox documentation and can be freely downloaded at www.spatial-econometrics.com/html/mbook.pdf)

Ljung, L. and P. E. Caines (1979). "Asymptotic Normality of Prediction Error Estimators for Approximate System Models," *Stochastic*, 3, 29-46.

Mauricio, A. (2008). "Computing and Using Residuals in Time Series Models," *Computational Statistics and Data Analysis*, 52, 3, 1746-1763.

Terceiro, J. (1990). *Estimation of Dynamic Econometric Models with Errors in Variables*. Springer-Verlag.

Terceiro, J., J. Casals, M. Jerez, G.R. Serrano and S. Sotoca (2000): "Time Series Analysis using MATLAB, Including a complete MATLAB Toolbox," (This is the User Manual for E^4, and can be freely downloaded at www.ucm.es/info/icae/e4).

Tiao, G. C. (2001). "Vector ARMA Models," In D. Peña, G.C. Tiao and R.S. Tsay (editors), *A Course in Time Series Analysis*, John Wiley & Sons.

Wei, W.W. (2005). *Time Series Analysis: Univariate and Multivariate Methods*. Addison-Wesley.

White, H. (1982). "Maximum Likelihood Estimation of Misspecified Models," *Econometrica*, 50, 1, 1-25.

Acknowledgements:

This work was funded by Ministerio de Educación y Ciencia, DGCICYT, Project SEJ2005-07388. Intellectual input from Jaime Terceiro, Gregorio Serrano, Patricio Gómez, Alfredo García-Hiernaux, Antonio García-Ferrer, Marcos Bujosa, and Alberto Mauricio is gratefully acknowledged. Special thanks to the small (but growing) community of E^4 users. Their many e-mails with encouragement messages, questions, bug reports and suggestions are very important to us.

Appendix A. Mathematical representation of the simple models supported.

A.1. Homoscedastic models.

E[4] supports to three simple models: VARMAX, structural econometric models and single-output transfer functions. A VARMAX model is given by:

$$\phi_p(\mathrm{B})\,\boldsymbol{\Phi}_P(\mathrm{B}^{\mathrm{S}})\,\boldsymbol{z}_t = \mathbf{G}_n(\mathrm{B})\,\boldsymbol{u}_t + \boldsymbol{\theta}_q(\mathrm{B})\,\boldsymbol{\Theta}_Q(\mathrm{B}^{\mathrm{S}})\,\boldsymbol{a}_t \tag{A.1}$$

where S denotes the period of the seasonal cycle, B is the backshift operator, such that for any sequence w_t: $B^k w_t = w_{t-k}$, $k = 0, \pm 1, \pm 2, \ldots$, the vectors $\boldsymbol{z}_t$, $\boldsymbol{u}_t$ and $\boldsymbol{a}_t$ were defined in Section 2.1 and the polynomial matrices in (A.1) are given by:

$$\phi_p(\mathrm{B}) = \mathbf{I} + \phi_1\mathrm{B} + \phi_2\mathrm{B}^2 + \ldots + \phi_p\mathrm{B}^p \tag{A.2}$$

$$\boldsymbol{\Phi}_P(\mathrm{B}^{\mathrm{S}}) = \mathbf{I} + \boldsymbol{\Phi}_1\mathrm{B}^{\mathrm{S}} + \boldsymbol{\Phi}_2\mathrm{B}^{2\cdot\mathrm{S}} + \ldots + \boldsymbol{\Phi}_P\mathrm{B}^{P\cdot\mathrm{S}} \tag{A.3}$$

$$\mathbf{G}_n(\mathrm{B}) = \mathbf{G}_1\mathrm{B} + \mathbf{G}_2\mathrm{B}^2 + \ldots + \mathbf{G}_n\,\mathrm{B}^n \tag{A.4}$$

$$\boldsymbol{\theta}_q(\mathrm{B}) = \mathbf{I} + \boldsymbol{\theta}_1\mathrm{B} + \boldsymbol{\theta}_2\mathrm{B}^2 + \ldots + \boldsymbol{\theta}_q\mathrm{B}^q \tag{A.5}$$

$$\boldsymbol{\Theta}_Q(\mathrm{B}^{\mathrm{S}}) = \mathbf{I} + \boldsymbol{\Theta}_1\mathrm{B}^{\mathrm{S}} + \boldsymbol{\Theta}_2\mathrm{B}^{2\cdot\mathrm{S}} + \ldots + \boldsymbol{\Theta}_Q\mathrm{B}^{Q\cdot\mathrm{S}} \tag{A.6}$$

The structural econometric model is similar to (A.1), but allowing for contemporary relationships between the endogenous variables and their errors. Specifically, it is given by:

$$\bar{\phi}_p(\mathrm{B})\,\bar{\boldsymbol{\Phi}}_P(\mathrm{B}^{\mathrm{S}})\,\boldsymbol{z}_t = \bar{\mathbf{G}}_n(\mathrm{B})\,\boldsymbol{u}_t + \bar{\boldsymbol{\theta}}_q(\mathrm{B})\,\bar{\boldsymbol{\Theta}}_Q(\mathrm{B}^{\mathrm{S}})\,\boldsymbol{a}_t \tag{A.7}$$

with:

$$\bar{\phi}_p(\mathrm{B}) = \bar{\phi}_0 + \bar{\phi}_1\mathrm{B} + \bar{\phi}_2\mathrm{B}^2 + \ldots + \bar{\phi}_p\mathrm{B}^p \tag{A.8}$$

$$\bar{\boldsymbol{\Phi}}_P(\mathrm{B}^{\mathrm{S}}) = \bar{\boldsymbol{\Phi}}_0 + \bar{\boldsymbol{\Phi}}_1\mathrm{B}^{\mathrm{S}} + \bar{\boldsymbol{\Phi}}_2\mathrm{B}^{2\cdot\mathrm{S}} + \ldots + \bar{\boldsymbol{\Phi}}_P\mathrm{B}^{P\cdot\mathrm{S}} \tag{A.9}$$

$$\bar{\mathbf{G}}_n(\mathrm{B}) = \bar{\mathbf{G}}_1\mathrm{B} + \bar{\mathbf{G}}_2\mathrm{B}^2 + \ldots + \bar{\mathbf{G}}_n\,\mathrm{B}^n \tag{A.10}$$

$$\bar{\theta}_q(\mathrm{B}) = \bar{\theta}_0 + \bar{\theta}_1\mathrm{B} + \bar{\theta}_2\mathrm{B}^2 + \ldots + \bar{\theta}_q\mathrm{B}^q \tag{A.11}$$

$$\bar{\boldsymbol{\Theta}}_Q(\mathrm{B}^{\mathrm{S}}) = \bar{\boldsymbol{\Theta}}_0 + \bar{\boldsymbol{\Theta}}_1\mathrm{B}^{\mathrm{S}} + \bar{\boldsymbol{\Theta}}_2\mathrm{B}^{2\cdot\mathrm{S}} + \ldots + \bar{\boldsymbol{\Theta}}_Q\mathrm{B}^{Q\cdot\mathrm{S}} \tag{A.12}$$

Finally, the single-output transfer function supported is:

$$z_t = \frac{\omega_1(\mathrm{B})}{\delta_1(\mathrm{B})}\,u_{1,t} + \ldots + \frac{\omega_r(\mathrm{B})}{\delta_r(\mathrm{B})}\,u_{r,t} + \frac{\theta_{\mathrm{q}}(\mathrm{B})\Theta_{\mathrm{Q}}(\mathrm{B}^{\mathrm{S}})}{\phi_{\mathrm{p}}(\mathrm{B})\Phi_{\mathrm{P}}(\mathrm{B}^{\mathrm{S}})}\,a_t \tag{A.13}$$

where:

$$\omega_i(\mathrm{B}) = \omega_0 + \omega_1\mathrm{B} + \ldots + \omega_{m_i}\mathrm{B}^{m_i} \quad (i = 1, 2, \ldots, r) \tag{A.14}$$

$$\delta_i(\mathrm{B}) = 1 + \delta_1\mathrm{B} + \ldots + \delta_{k_i}\mathrm{B}^{k_i} \quad (i = 1, 2, \ldots, r) \tag{A.15}$$

A.2. Models with GARCH errors.

E[4] allows one to combine any VARMAX model, transfer function or SS in the steady-state innovations form (2.4)-(2.5), with a vector GARCH process for the error $\boldsymbol{a}_t$. For a survey on conditional heteroscedastic processes, see Bollerslev, Engle and Nelson (1994).

The general conditional heteroscedastic process is defined by the distributions $\boldsymbol{a}_t \sim iid\,(\mathbf{0}, \boldsymbol{Q})$, $\boldsymbol{a}_t \big| \boldsymbol{\Omega}_{t-1} \sim iid\,(\mathbf{0}, \boldsymbol{\Sigma}_t)$ where $\boldsymbol{\Omega}_{t-1}$ is the set of information available up to t-1.

In a GARCH model, the conditional covariances, $\boldsymbol{\Sigma}_t$, are such that:

$$\left[\mathbf{I} - \mathbf{B}_{\mathrm{m}}(\mathrm{B})\right]\mathrm{vech}\left(\boldsymbol{\Sigma}_t\right) = \mathbf{w} + \mathbf{A}_{\mathrm{k}}(\mathrm{B})\mathrm{vech}\left(\boldsymbol{a}_t\boldsymbol{a}_t^T\right) \tag{A.16}$$

where vech(.) denotes the vector-half operator, which stacks the columns of the lower triangle of a $\mathrm{N}\times\mathrm{N}$ symmetric matrix into a $\left[\mathrm{N}\left(\mathrm{N}+1\right)/2\right]\times 1$ vector, and the polynomial matrices are:

$$\mathbf{B}_{\mathrm{m}}(\mathrm{B}) = \mathbf{B}_1\,\mathrm{B} + \mathbf{B}_2\,\mathrm{B}^2 + \ldots + \mathbf{B}_{\mathrm{m}}\,\mathrm{B}^{\mathrm{m}} \tag{A.17}$$

$$\mathbf{A}_{\mathrm{k}}(\mathrm{B}) = \mathbf{A}_1\,\mathrm{B} + \mathbf{A}_2\,\mathrm{B}^2 + \ldots + \mathbf{A}_{\mathrm{k}}\,\mathrm{B}^{\mathrm{k}} \tag{A.18}$$

E^4 manages model (A.16) in its alternative VARMAX representation. To derive it, consider the white noise process $\boldsymbol{v}_t = \mathrm{vech}\left(\boldsymbol{a}_t\boldsymbol{a}_t^T\right) - \mathrm{vech}\left(\boldsymbol{\Sigma}_t\right)$, which has a null conditional mean and a very complex heteroscedastic structure. Then $\mathrm{vech}\left(\boldsymbol{\Sigma}_t\right) = \mathrm{vech}\left(\boldsymbol{a}_t\boldsymbol{a}_t^T\right) - \boldsymbol{v}_t$ and, substituting this expression in (A.16) and rearranging some terms, yields:

$$\left[\mathbf{I} - \mathbf{A}_{\mathrm{k}}(\mathrm{B}) - \mathbf{B}_{\mathrm{m}}(\mathrm{B})\right]\mathrm{vech}\left(\boldsymbol{a}_t\boldsymbol{a}_t^T\right) = \mathbf{w} + \left[\mathbf{I} - \mathbf{B}_{\mathrm{m}}(\mathrm{B})\right]\boldsymbol{v}_t \tag{A.19}$$

Eq. (A.19) defines a VARMAX model for $\mathrm{vech}\left(\boldsymbol{a}_t\boldsymbol{a}_t^T\right)$, which can be written as:

$$\mathrm{vech}\left(\boldsymbol{a}_t\boldsymbol{a}_t^T\right) = \mathrm{vech}\left(\boldsymbol{Q}\right) + \boldsymbol{N}_t \tag{A.20}$$

$$\left[\mathbf{I} + \bar{\mathbf{A}}_{\max\{\mathrm{k},\mathrm{m}\}}(\mathrm{B})\right]\boldsymbol{N}_t = \left[\mathbf{I} + \bar{\mathbf{B}}_{\mathrm{m}}(\mathrm{B})\right]\boldsymbol{v}_t \tag{A.21}$$

where $\bar{\mathbf{A}}_{\max\{\mathrm{k}\,m\}}(\mathrm{B}) = -\mathbf{A}_{\mathrm{k}}(\mathrm{B}) - \mathbf{B}_{\mathrm{m}}(\mathrm{B})$ and $\bar{\mathbf{B}}_{\mathrm{m}}(\mathrm{B}) = -\mathbf{B}_{\mathrm{m}}(\mathrm{B})$. This is the formulation supported by E^4. Note that it allows for IGARCH (Integrated-GARCH) components, which would require specifying some unit roots in the AR factor of (A.21).

Appendix B. Mathematical definition of nested models.

B.1. Nesting in inputs.

Consider a SS model:

$$\boldsymbol{x}_{t+1}^{a} = \boldsymbol{\Phi}^{\mathrm{a}}\, \boldsymbol{x}_{t}^{a} + \boldsymbol{\Gamma} \boldsymbol{u}_{t} + \mathbf{E}^{\mathrm{a}}\, \boldsymbol{w}_{t}^{a} \tag{B.1}$$

$$\boldsymbol{z}_{t} = \mathbf{H}^{\mathrm{a}}\, \boldsymbol{x}_{t}^{a} + \mathbf{D}\, \boldsymbol{u}_{t} + \mathbf{C}^{\mathrm{a}}\, \boldsymbol{v}_{t}^{a} \tag{B.2}$$

and the SS model for the inputs in (B.1)-(B.2):

$$\boldsymbol{x}_{t+1}^{b} = \boldsymbol{\Phi}^{\mathrm{b}}\, \boldsymbol{x}_{t}^{b} + \mathbf{E}^{\mathrm{b}}\, \boldsymbol{w}_{t}^{b} \tag{B.3}$$

$$\boldsymbol{u}_{t} = \mathbf{H}^{\mathrm{b}}\, \boldsymbol{x}_{t}^{b} + \mathbf{C}^{\mathrm{b}}\, \boldsymbol{v}_{t}^{b} \tag{B.4}$$

where the errors in (B.1)-(B.2) and (B.3)-(B.4) are mutually independent. Substituting (B.4) in (B.1)-(B.2) yields:

$$\boldsymbol{x}_{t+1}^{a} = \boldsymbol{\Phi}^{\mathrm{a}}\, \boldsymbol{x}_{t}^{a} + \boldsymbol{\Gamma}\left(\mathbf{H}^{\mathrm{b}}\, \boldsymbol{x}_{t}^{b} + \mathbf{C}^{\mathrm{b}}\, \boldsymbol{v}_{t}^{b}\right) + \mathbf{E}^{\mathrm{a}}\, \boldsymbol{w}_{t}^{a} \tag{B.5}$$

$$\boldsymbol{z}_{t} = \mathbf{H}^{\mathrm{a}}\, \boldsymbol{x}_{t}^{a} + \mathbf{D}\left(\mathbf{H}^{\mathrm{b}}\, \boldsymbol{x}_{t}^{b} + \mathbf{C}^{\mathrm{b}}\, \boldsymbol{v}_{t}^{b}\right) + \mathbf{C}^{\mathrm{a}}\, \boldsymbol{v}_{t}^{a} \tag{B.6}$$

Finally, the state equation of the nested model results from combining Eqs. (B.5) and (B.3) into a single state equation:

$$\begin{bmatrix} \boldsymbol{x}_{t+1}^{a} \\ \boldsymbol{x}_{t+1}^{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}^{\mathrm{a}} & \boldsymbol{\Gamma}\mathbf{H}^{\mathrm{b}} \\ \mathbf{0} & \boldsymbol{\Phi}^{\mathrm{b}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{t}^{a} \\ \boldsymbol{x}_{t}^{b} \end{bmatrix} + \begin{bmatrix} \mathbf{E}^{\mathrm{a}} & \boldsymbol{\Gamma}\mathbf{C}^{\mathrm{b}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{E}^{\mathrm{b}} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_{t}^{a} \\ \boldsymbol{v}_{t}^{b} \\ \boldsymbol{w}_{t}^{b} \end{bmatrix} \tag{B.7}$$

and the observer results from combining Eqs. (B.6) and (B.4):

$$\begin{bmatrix} z_t \\ u_t \end{bmatrix} = \begin{bmatrix} \mathbf{H}^{\mathrm{a}} & \mathbf{D}\mathbf{H}^{\mathrm{b}} \\ \mathbf{0} & \mathbf{H}^{\mathrm{b}} \end{bmatrix} \begin{bmatrix} x_t^a \\ x_t^b \end{bmatrix} + \begin{bmatrix} \mathbf{C}^{\mathrm{a}} & \mathbf{D}\mathbf{C}^{\mathrm{b}} \\ \mathbf{0} & \mathbf{C}^{\mathrm{b}} \end{bmatrix} \begin{bmatrix} v_t^a \\ v_t^b \end{bmatrix} \tag{B.8}$$

B.1. Nesting in errors.

Consider a SS model in innovations form:

$$x_{t+1}^a = \mathbf{\Phi}^{\mathrm{a}} x_t^a + \mathbf{\Gamma} u_t + \mathbf{E}^{\mathrm{a}} a_t^a \tag{B.9}$$

$$z_t = \mathbf{H}^{\mathrm{a}} x_t^a + \mathbf{D} u_t + a_t^a \tag{B.10}$$

where the errors a_t^a are colored noise, resulting from the SS model:

$$x_{t+1}^b = \mathbf{\Phi}^{\mathrm{b}} x_t^b + \mathbf{E}^{\mathrm{b}} a_t^b \tag{B.11}$$

$$a_t^a = \mathbf{H}^{\mathrm{b}} x_t^b + a_t^b \tag{B.12}$$

Substituting (B.12) in (B.9) and (B.10) yields:

$$x_{t+1}^a = \mathbf{\Phi}^{\mathrm{a}} x_t^a + \mathbf{\Gamma} u_t + \mathbf{E}^{\mathrm{a}} \left(\mathbf{H}^{\mathrm{b}} x_t^b + a_t^b \right) \tag{B.13}$$

$$z_t = \mathbf{H}^{\mathrm{a}} x_t^a + \mathbf{D} u_t + \mathbf{H}^{\mathrm{b}} x_t^b + a_t^b \tag{B.14}$$

The state equation of the nested model results from combining Eqs. (B.13) and (B.11) into a single state equation:

$$\begin{bmatrix} \boldsymbol{x}_{t+1}^{a} \\ \boldsymbol{x}_{t+1}^{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}^{\mathrm{a}} & \mathbf{E}^{\mathrm{a}}\mathbf{H}^{\mathrm{b}} \\ \mathbf{0} & \boldsymbol{\Phi}^{\mathrm{b}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{t}^{a} \\ \boldsymbol{x}_{t}^{b} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Gamma} \\ \mathbf{0} \end{bmatrix} \boldsymbol{u}_{t} + \begin{bmatrix} \mathbf{E}^{\mathrm{a}} \\ \mathbf{E}^{\mathrm{b}} \end{bmatrix} \boldsymbol{a}_{t}^{b} \tag{B.15}$$

and the nested observer results from rearranging terms in (B.14):

$$\boldsymbol{z}_{t} = \begin{bmatrix} \mathbf{H}^{\mathrm{a}} & \mathbf{H}^{\mathrm{b}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{t}^{a} \\ \boldsymbol{x}_{t}^{b} \end{bmatrix} + \mathbf{D}\,\boldsymbol{u}_{t} + \boldsymbol{a}_{t}^{b} \tag{B.16}$$

Table 1. Standard procedures for time series analysis.

Function	Description	References
augdft	Computes the augmented Dickey-Fuller test	Dickey and Fuller (1981)
descser	Computes some common descriptive statistics for a vector of time series	--
histsers	Computes and displays histograms for a set of time series	
lagser	Generates lags and leads from a data matrix	
midents	Multiple sample autocorrelation and partial autoregression functions for a vector of series	Wei (2005)
plotsers	Displays standardized plots for a vector of time series	--
rmedser	Displays the mean/std. deviation plot of a vector of time series	Box and Cox (1964)
transdif	Computes differences and Box-Cox transforms for a vector of time series	Box, Jenkins and Reinsel (1994)
uidents	Computes the sample autocorrelation and partial autocorrelation functions for a vector of time series	
simmod	Simulates a homoscedastic model	--
simgarch	Simulates a model with GARCH errors	

Table 2. Signal extraction procedures.

Function	Description	References
`aggrmod`	Computes smoothed estimates for a high-frequency time series from: (a) a sample with low and high-frequency data, and (b) the high-frequency model for the endogenous variable(s)	Casals, Jerez and Sotoca (2004)
`e4trend`	Decomposes a vector of time series into the trend, seasonal, cycle and irregular components implied by an econometric model in THD format	Casals, Jerez and Sotoca (2002)
`fismod`	Computes fixed interval smoothing estimates of the state and observable variables of a model in THD form. The function `fismiss` allows for in-sample missing values	Casals, Jerez and Sotoca (2000) Mauricio (2008)
`fismiss`		
`foremod`	Computes forecasts for a homoscedastic model in THD format	
`foregarc`	Computes forecasts for a model in THD format, allowing for GARCH errors	
`residual`	Computes the residuals for a model in THD format	

Table 3. Likelihood and model estimation algorithms.

Function	Description	References
Likelihood computation and derivatives		
`lfmod`	Computes the log-likelihood function for a model in THD form. `lfmod` and `lffast` support models with homoscedastic errors, while `lfgarch` admits GARCH errors. `lfmiss` allows for in-sample missing values	Terceiro (1990) Casals and Sotoca (1997) Casals, Sotoca and Jerez (1999)
`lffast`		
`lfmiss`		
`lfgarch`		
`gmod`	Computes the analytical gradient for `lfmod` and `lffast` (`gmod`), `lfmiss` (`gmiss`) and `lfgarch` (`ggarch`)	
`gmiss`		
`ggarch`		Casals and Sotoca (2001)
`imod`	Computes the Gaussian analytical information matrix for `lfmod` and `lffast` (`imod`), `lfmiss` (`imiss`) and lfgarch (`ggarch`). The function `imodg` computes the robustified information matrix for `lfmod` and `lffast`	Terceiro (1990) Ljung and Caines (1979) White (1982)
`imiss`		
`imodg`		
`igarch`		
Model estimation		
`e4min`	Computes the unconstrained numerical minimum of a nonlinear function using the BFGS or the Newton-Raphson algorithms	Dennis and Schnabel (1983)
`e4preest`	Computes a fast estimate of the parameters for a model in THD format	García-Hiernaux, Jerez and Casals (2008)
`prtest`	Displays estimation results	--
Toolbox defaults and options		
`e4init`	Initializes the global variables and defaults	--
`sete4opt`	Modifies the toolbox options	

Figure 1. Seasonally-adjusted Starts and Sales series resulting from models (4.1)-(4.2).

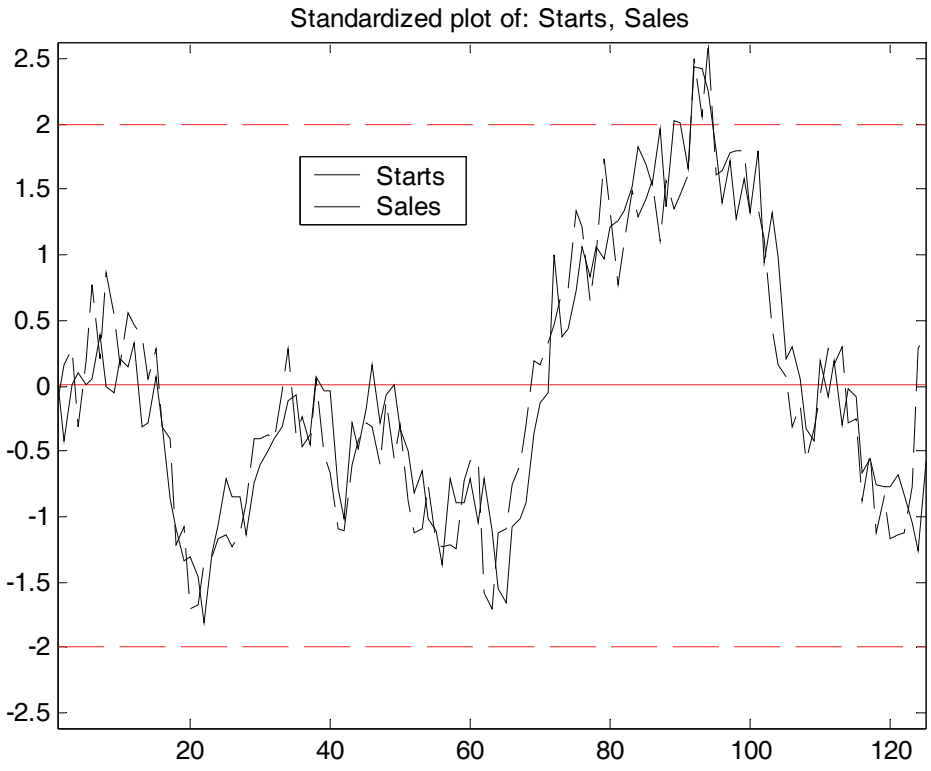

Figure 2. Conditional correlations between the excess returns of DM/USD and JY/USD.