This file describes the main codes needed to compute the second-order accurate approximation of the solution to a DSGE model featuring time-varying risk, as in "Second-Order Approximation of Dynamic Models with Time-Varying Risk", by G. Benigno, P. Benigno and S. Nisticò (JEDC, 2013)

The codes were written using MATLAB R2008b

MAIN CODES:

```
-----------------------------------------------------------------------------------------------------------
anal_deriv.m    Computes analytical first and second derivatives of the vector function f(yp,xp,y,x)
                with respect to yp, xp, y and x.
                Input:  the vector function f, its arguments, and the desired order of approximation
                Output: Analytical Jacobian and Magnus-Neudecker Hessian matrices of f.
                Notes:  The default value of for the order of approximation is 2.
                        This code requires MATLAB's Symbolic Math Toolbox

diagm.m         Defines the diagm operator, called by soappr_var.m
                Given (mn x n) matrix x, such that x = [x1; x2; ... xm], where each xi is (n x n)
                computes (n x m) matrix y, such that column i is the main diagonal of submatrix xi
                Input:  matrix x
                Output: matrix y

diagmn.m        Defines the diagmn operator, called by soappr_std.m
                Given (mn x n) matrix x, such that x = [x1; x2; ... xm], where each xi is (n x n)
                computes (mn x n) matrix y, such that y = [y1; y2; ... ym], where each yi is (n x n)
                and yi = diag(diag(xi)).
                Input:  matrix x
                Output: matrix y

foappr.m        Computes the first-order accurate approximation of the solution to a DSGE model
                Input:  the numerical value of the Jacobian matrix, appropriately partitioned
                Ouput:  the numerical value of the FOA coefficients, and an exitflag indicating
                        whether there is no solution, unique solution or indeterminacy
                Note:   the FOA coefficients are independent of the specific process assumed for
                        stochastic volatility (Certainty Equivalence)

ir.m            Computes T-period impulse-response functions for the vector of variables, solving a
                DSGE model with time-varying risk up to a second-order approximation.
                Input:  four scalars indicating 1. the class of shock (structural vs volatility),
                        2. the specific shock to simulate, 3. the specific model of stochastic
                        volatility (std vs variance), 4. the length of the impulse-responses (T);
                        the value of the structural parameters of the stochastic volatility process;
                        the numerical value of the Jacobian and MN Hessian matrices
                Output: the impulse-response functions, as a matrix of dimensions (n+nz x T)

MNhessian.m     Computes the Modified Magnus-Neudecker Hessian matrix.
                Given a tridimensional (n1 x n2 x n3) Hessian array x, computes appropriate
                permutations to reshape it into the two-dimensional (n1*n2 x n3) Modified MN Hessian
                matrix y
                Input:  matrix x
                Output: matrix y

num_eval.m      Evaluates the analytical first and second derivatives of f numerically.
                Input:  none
                Output: the numerical value of the Jacobian and Hessian matrices
                Calls:  MNhessian.m
                Note:   The second derivative is the Modified MN Hessian matrix of function f
                        The parameters, steady state values of the arguments of the function f, and
                        the order of approximation must be in the workspace.

soappr_std.m    Computes the second-order accurate approximation of the solution to a DSGE model
                with time-varying risk, where stochastic volatility is modeled by means of a linear
                process for the conditional STANDARD DEVIATION of structural shocks
                Input:  the numerical value of the Jacobian and Hessian matrices of function f, and
                        the numerical value of the coefficients of the first-order approximation
                Output: the numerical value of the coefficients of the second-order approximation
                Calls:  tracem.m
                        diagmn.m

soappr_var.m    Computes the second-order accurate approximation of the solution to a DSGE model
                with time-varying risk, where stochastic volatility is modeled by means of a linear
                process for the conditional VARIANCE of structural shocks
                Input:  the numerical value of the Jacobian and Hessian matrices of function f, and
                        the numerical value of the coefficients of the first-order approximation
                Output: the numerical value of the coefficients of the second-order approximation
                Calls:  diagm.m

tracem.m        Defines the tracem operator, called by soappr_std.m
                Given (mn x n) matrix x, such that x = [x1; x2; ... xm], where each xi is (n x n)
                computes (m x 1) vector y, such that element i is the trace of submatrix xi
                Input:  matrix x
                Output: vector y
```

NOTE: The codes compute the SOA as shown in the paper, using equations 33, 35 and 36. These equations use four MN Hessian matrices, each with respect to one of the four types of variables in the model (current controls, expected one-period-ahead controls, current states, one-period-ahead states) as defined in the equation right before 30. Accordingly, the Hessian matrix appearing in equation 28 is actually a Modified Magnus-Neudecker Hessian, defined as the vertical stack of the four MN matrices mentioned above, as in the equation right after 29. The definition implied by the equation right after 28, instead, refers to the vertical stack of the Hessian matrices of function f wrt all variables in the model. The latter equation should be disregarded, and the definition of the equation after the 29 should be used instead.
We thank Ivan Sutoris for bringing this point to our attention.

REPLICATION CODES:

RBC_model_run.m	This is the master file replicating the application to a simple neoclassical growth model with time-varying risk, discussed in the paper ``Second-Order Approximation of Dynamic Models with Time-Varying Risk,'' by Gianluca Benigno, Pierpaolo Benigno and Salvatore Nisticò, (2010) The code goes through the following steps: 1. set the desired order of approximation (1. or 2.; default is 2.) 2. set the specific model of stochastic volatility (0. constant volatility, 1. linear process for standard deviation, 2. linear process for variance; default is 2.) 3. set up the specific model to be analyzed: define the function f and its arguments - Calls the subroutine RBC_model.m 4. compute analytical derivatives: the Jacobian and MN Hessian matrices of f - Calls the subroutine anal_deriv.m 5. set up steady state and numerical values of parameters, for numerical evaluation - Calls the subroutine RBC_model_ss.m 6. evaluate the Jacobian and MN Hessian matrices numerically - Calls the subroutine num_eval.m 7. partition the numerical Jacobian for computation of FOA coefficients 8. compute matrix coefficients of first-order approximation - Calls the subroutine foappr.m 9. if approx=2, compute matrix coefficients of second-order approximation, for the specific stochastic volatility model selected in step 2. - Calls the subroutines soappr_std.m and soappr_var.m
RBC_model.m	This code sets up the neoclassical growth model: defines the vector function f, the relevant arguments with respect to which to take the Taylor expansion, the symbolic variables with respect to which to compute the analytical derivatives.
RBC_model_ss.m	This code produces the the deep structural parameters and computes the steady state of the neoclassical growth model
RBC_irfig.m	This code replicates the figure displaying the impulse-response function to a shock to the volatility of productivity, discussed in Benigno, Benigno and Nisticò (2010)