1 Overview

This version: April 1, 2009.

This file is a rough user's guide to the software, linlagex, containing an overview of the method and its implementation into MATLAB as well as two of the example models included with the software are discussed (a sticky-information model (Trabandt 2007) and a labor-hoarding model (Wang and Wen 2006)) in order to illustrate how models with lagged expectations (infinite series reaching back into the infinite past with the first example and single lagged expectations reaching back into the finite past) can be implemented with the software. The current options and their defaults are explained in the final section.

2 Overview of the Method

The reader wishing to only use the software is advised to skip this section. This section is designed to provide a brief overview of the methods presented in the associated paper and, thus, sacrifices much of the derivation for the sake of brevity: the truly interested reader is directed to the associated paper.

The software is designed to solve linear rational expectations models of the following form:

$$\begin{aligned} 0 &= \sum_{i=0}^{I} A_i E_{t-i}\left[Y_{t+1}\right] + \sum_{i=0}^{I} B_i E_{t-i}\left[Y_t\right] + \sum_{i=0}^{I} C_i E_{t-i}\left[Y_{t-1}\right] \\ &+ \sum_{i=0}^{I} F_i E_{t-i}\left[W_{t+1}\right] + \sum_{i=0}^{I} G_i E_{t-i}\left[W_t\right] \end{aligned} \tag{1}$$

$$W_t = \sum_{j=0}^{\infty} N^j \epsilon_{t-j},\ \epsilon_t \sim i.i.d.\mathcal{N}\left(0, \Omega\right) \tag{2}$$

$$\lim_{j \to \infty} \xi^{-j} E_t\left[Y_{t+j}\right] = 0,\ \forall \xi \in \mathbb{R}\ s.t.\ \xi > g^u,\ \text{where } g^u \geq 1 \tag{3}$$

where Y_t be an $n \times 1$ vector of endogenous variables of interest, W_t an $k \times 1$ vector of exogenous processes with moving average coefficients $\{N^j\}_{j=0}^{\infty}$ consistent with a VAR(1) representation, and where $I \in \mathbb{N}_0$. That the system not be underdetermined, the dimensions of the matrices in (1) are such that

there are as many equations (k) as endogenous variables of interest. Following, e.g., Uhlig (1999) and Sims (2001), variables dated t are in the information set at t; thusly, "state" variables are the non-empty column spaces of C_i.

As I show in the associated paper, the essentially "infinite dimensional" problem (by expanding the state vector for standard methods, i.e. Uhlig (1999), GENSYS, DYNARE, SOLAB, etc.) can be converted to a non-autonomous non-homogenous deterministic difference equation in matrices of size $n \times k$.

Following Muth (1961) and Taylor (1986), the solution in the form of MA coefficients is

$$Y_t = \sum_{j=0}^{\infty} \Theta_j \epsilon_{t-j} \tag{4}$$

where Θ_j is of size $n \times k$.

Inserting the $MA(\infty)$ representation into equation (1),

$$\begin{aligned} 0 &= \sum_{j=0}^{\infty} \left(\sum_{i=0}^{min(I,j)} A_i \right) \Theta_{j+1} \epsilon_{t-j} + \sum_{j=0}^{\infty} \left(\sum_{i=0}^{min(I,j)} B_i \right) \Theta_j \epsilon_{t-j} \\ &+ \sum_{j=0}^{\infty} \left(\sum_{i=0}^{min(I,j+1)} C_i \right) \Theta_j \epsilon_{t-j-1} + \sum_{j=0}^{\infty} \left(\sum_{i=0}^{min(I,j)} F_i \right) N_{j+1} \epsilon_{t-j} \\ &+ \sum_{j=0}^{\infty} \left(\sum_{i=0}^{min(I,j)} G_i \right) N_j \epsilon_{t-j} \end{aligned} \tag{5}$$

and defining

$$\tilde{M}_j = \sum_{i=0}^{min(I,j)} M_i, \text{ for } M = A, B, C, F, G \tag{6}$$

yields the non-stochastic linear recursion

$$0 = \tilde{A}_j \Theta_{j+1} + \tilde{B}_j \Theta_j + \tilde{C}_j \Theta_{j-1} + \tilde{F}_j N^{j+1} + \tilde{G}_j N^j \tag{7}$$

with initial conditions

$$\Theta_{-1} = 0 \tag{8}$$

and terminal conditions from (3)

$$\lim_{j \to \infty} \xi^{-j} \Theta_j = 0 \tag{9}$$

Essentially, a unique solution with respect to the uniform growth restriction will exist if the terminal conditions from equation (9) provide "enough" linear restrictions on the recursion to derive "initial conditions" for Θ_0.

So long as I **is finite**, we can convert the difference equation to block tri-diagonal system of $I+1$ linear equations:

$$\begin{bmatrix} \tilde{B}_0 & \tilde{A}_0 & 0 & & & \dots & & & 0 \\ \tilde{C}_1 & \tilde{B}_1 & \tilde{A}_1 & 0 & & \dots & & & 0 \\ 0 & \tilde{C}_2 & \tilde{B}_2 & \tilde{A}_2 & 0 & \dots & & & 0 \\ \vdots & & & & & & & & \vdots \\ 0 & & \dots & & 0 & \tilde{C}_{I-1} & \tilde{B}_{I-1} & \tilde{A}_{I-1} \\ 0 & & & \dots & & 0 & -\left(Z_{21}^I {Z_{11}^I}^{-1}\right) & I_n \end{bmatrix} \begin{bmatrix} \Theta_0 \\ \Theta_1 \\ \Theta_2 \\ \vdots \\ \Theta_{I-1} \\ \Theta_I \end{bmatrix}$$

$$= \begin{bmatrix} -\tilde{F}_0 N^1 - \tilde{G}_0 N^0 \\ -\tilde{F}_1 N^2 - \tilde{G}_1 N^1 \\ -\tilde{F}_2 N^3 - \tilde{G}_2 N^2 \\ \vdots \\ -\tilde{F}_{I-1} N^I - \tilde{G}_{I-1} N^{I-1} \\ \left(Z_{22}^I - Z_{21}^I {Z_{11}^I}^{-1} Z_{12}^I\right) M_I N^I \end{bmatrix} \tag{10}$$

with I_n being the n-dimensional identity matrix. The last equation is derived via a QZ-decomposition of the autonomous (remember, I is finite, thus the coefficients of the recursion are non-varying for $j \geq I$) non-homogenous difference equation:

$$\begin{bmatrix} 0 & -A_I \\ I_n & 0 \end{bmatrix} \begin{bmatrix} \Theta_I \\ \Theta_{I+1} \end{bmatrix} = \begin{bmatrix} C_I & B_I \\ 0 & I_n \end{bmatrix} \begin{bmatrix} \Theta_{I-1} \\ \Theta_I \end{bmatrix} + \begin{bmatrix} F_I N^{I+1} + G_I N^I \\ 0 \end{bmatrix} \tag{11}$$

If the system has k generalized eigenvalues less than or equal to the maximal growth rate g^u and k greater than the same, and if the given initial conditions are "translatable" (a more detailed discussion of these conditions can be found in the associated paper, based on the discussion in Klein (2000)), then $\forall j \geq I$

$$\Theta_j = \left(Z_{21} Z_{11}^{-1}\right) \Theta_{j-1} + \left(Z_{22} - Z_{21} Z_{11}^{-1} Z_{12}\right) M_I N^I \tag{12}$$

where

$$M_j = -T_{22}^{-1} \sum_{k=0}^{\infty} \left[T_{22}^{-1} S_{22}\right]^k Q_2 \begin{bmatrix} F_j N^{k+1} + G_j N^k \\ 0 \end{bmatrix} \tag{13}$$

The matrices Q, T, S, and Z are associated with the QZ-decomposition and the interested reader is directed to the discussion in the associated paper.

If I **is infinite**, additional assumptions are needed. Namely that the limit of the "sum" matrices defined earlier exist: thus(where k and l denote row and column)

$$\lim_{j\to\infty}\left(\tilde{M}_j\right)_{k,l}=\left(\tilde{M}_\infty\right)_{k,l},\ \text{for } M=A,B,C,F,G \tag{14}$$

In the associated paper, I show (using the definition of a limit in $\mathbb{R}^1$, that there exists some $I(\delta)_{max}$, where δ is a tolerance level, such that the system can be reduced to the form derived above.

$$0=\tilde{A}_j\Theta_{j+1}+\tilde{B}_j\Theta_j+\tilde{C}_j\Theta_{j-1}+\tilde{F}_jN_{j+1}+\tilde{G}_jN_j,\ 0\leq j<I(\delta)_{max} \tag{15}$$

and

$$0=\tilde{A}_\infty\Theta_{j+1}+\tilde{B}_\infty\Theta_j+\tilde{C}_\infty\Theta_{j-1}+\tilde{F}_\infty N_{j+1}+\tilde{G}_\infty N_j,\ j\geq I(\delta)_{max} \tag{16}$$

Details can be found in the associated paper. Note that this is still an approximation of the actual solution when lagged expectations reach back into the infinite past, but I have formalized the method of approximating (e.g. qualifying statements in the sticky-information literature of the form "adding further lagged expectations doesn't change the solution much").

Thus, a solution for the infinite MA coefficients has been found. These coefficients are the impulse response functions (see Hamilton (1994, pp. 318-19)).

The impulse responses to anticipated/pre-announced shocks can be found (see Taylor (1986))by solving the following recursion:

$$0=\tilde{A}_j\Theta_{j+1}+\tilde{B}_j\Theta_j+\tilde{C}_j\Theta_{j-1},\ \text{for}j<J \tag{17}$$

$$0=\tilde{A}_j\Theta_{j+1}+\tilde{B}_j\Theta_j+\tilde{C}_j\Theta_{j-1}+\tilde{F}_jN^{j+1-J}+\tilde{G}_jN^{j-J},\ \text{for}j\geq J \tag{18}$$

with initial conditions

$$\Theta_{-1}=0 \tag{19}$$

and terminal conditions from (3)

$$\lim_{j\to\infty}\xi^{-j}\Theta_j=0 \tag{20}$$

Solving this requires solving a system of linear equations with the recursive form derived via a QZ-decomposition for the autonomous part of the recursion: the framework for doing such was already established in the foregoing!

Only the entries on the right-hand-side of the tri-diagonal system presented above change (if $J > I$, the both sides will need to be extended, but the new entries will simply repeat values already determined).

Population moments can be found via an inverse Fourier transformation of the population spectrum (see Hamilton (1994) for the derivation, Uhlig (1999) and associated programs for implementation and inclusion of the HP-filter, and my associated paper for the "recursive" system used in the calculations here) if the model is covariance stationary.

Simulations and simulated moments can be obtained by drawing a sequence of shocks from the distribution of the shocks associated with the exogenous processes.

3 Using the Software

The software solves for models of the form

$$\begin{aligned} 0 &= \sum_{i=0}^{I} A_i E_{t-i}\left[Y_{t+1}\right] + \sum_{i=0}^{I} B_i E_{t-i}\left[Y_t\right] + \sum_{i=0}^{I} C_i E_{t-i}\left[Y_{t-1}\right] \\ &+ \sum_{i=0}^{I} F_i E_{t-i}\left[W_{t+1}\right] + \sum_{i=0}^{I} G_i E_{t-i}\left[W_t\right] \end{aligned} \tag{21}$$

$$W_t = \sum_{j=0}^{\infty} N^j \epsilon_{t-j},\ \epsilon_t \sim i.i.d.\mathcal{N}\left(0,\Omega\right) \tag{22}$$

$$\lim_{j\to\infty} \xi^{-j} E_t\left[Y_{t+j}\right] = 0,\ \forall \xi \in \mathbb{R}\ s.t.\ \xi > g^u,\ \text{where } g^u \geq 1 \tag{23}$$

where Y_t be an $n \times 1$ vector of endogenous variables of interest, W_t an $k \times 1$ vector of exogenous processes with moving average coefficients $\{N^j\}_{j=0}^{\infty}$ consistent with a VAR(1) representation, and where $I \in \mathbb{N}_0$. That the system not be underdetermined, the dimensions of the matrices in (1) are such that there are as many equations (k) as endogenous variables of interest. Following, e.g., Uhlig (1999) and Sims (2001), variables dated t are in the information set at t; thusly, "state" variables are the non-empty column spaces of C_i.

The user needs to define (in a MATLAB ".m file"):

- The structural parameters of the model,

- The names of the column spaces (i.e. variables) of Y_t and W_t as MATLAB strings in a MATLAB "cell" (i.e. using curly brackets) vector,
- A_0, B_0, C_0, F_0, and G_0 (the coefficient matrices with out lagged expectations) as MATLAB matrices,
- Functions A_j, B_j, C_j, F_j, and G_j (the coefficient matrices with lagged expectations) as MATLAB strings (My software will translate these into MATLAB autonomous functions), unless no lagged expectations are present,
- `it_name`, the name of "j" used by the user in the preceding matrix functions, as a MATLAB string, unless no lagged expectations are present,
- `it_max_value`, the lag of the "most lagged" expectation in the model (i.e. "I" in the statement of the problem above), as either an integer when "I" is finite or as a string if "I" is infinite — if no lagged expectations are present, this is equal to 0,
- Matrices N and Ω to define the exogenous process
- `PERIOD`, the number of periods per year in the model (e.g. "4" if the model is quarterly), as a MATLAB scalar,
- call "linlagex" at the end of the ".m file".

The user, who does not have MATLAB's Symbolic Toolkbox, and who wants to solve a model with infinite lagged expectations, needs to additionally define:

- `A_inf`, `B_inf`, `C_inf`, `F_inf`, and `G_inf` (the sum of the coefficient matrices; e.g. `A_inf`= $\sum_{i=0}^{\infty} A_i$, `B_inf`= $\sum_{i=0}^{\infty} B_i$, etc.) as MATLAB matrices

All users can, additionally, define the options listed in the last section of this guide to control the solution and output of the program.

To better understand the implementation, two of the included example programs are discussed.

4 Example: Trabandt (2007)

Trabandt (2007) derives a DSGE model with sticky-information price setters

under monopolistic competition. The structural equations of the model are:

$$\begin{aligned}
0 &= -m_t - \pi_t + m_{t-1} + \xi_t \\
0 &= -rr_t + \mu_z z_t + \mu_g g_t \\
0 &= -m_t + \frac{\sigma}{s_c \nu} x_t - \frac{1}{\nu\left(\bar{R}-1\right)} R_t + \frac{\sigma}{s_c \nu \psi} z_t - \gamma_g g_t \\
0 &= \frac{s_c}{\sigma} E_t\left[\pi_{t+1}\right] + E_t\left[x_{t+1}\right] + \frac{s_c}{\sigma} rr_t - x_t - \frac{s_c}{\sigma} R_t \\
0 &= -\pi_t + \frac{1-\lambda}{\lambda}\xi x_t + (1-\lambda)\sum_{i=0}^{\infty} \lambda^i E_{t-i-1}\left[\pi_t + \xi \Delta x_t\right] \\
0 &= -\Delta x_t + x_t - x_{t-1} \\
z_t &= \rho_z z_{t-1} + \epsilon_t^z,\ \epsilon_t^z \sim \mathcal{N}\left(0, \sigma_z^2\right) \\
g_t &= \rho_g g_{t-1} + \epsilon_t^g,\ \epsilon_t^g \sim \mathcal{N}\left(0, \sigma_g^2\right) \\
\xi_t &= \rho_\xi \xi_{t-1} + \epsilon_t^\xi,\ \epsilon_t^\xi \sim \mathcal{N}\left(0, \sigma_\xi^2\right)
\end{aligned}$$

The variables can be divided into endogenous variables:

$$Y_t = \begin{bmatrix} m_t & \pi_t & rr_t & x_t & R_t & \Delta x_t \end{bmatrix}' \tag{24}$$

and exogenous variables:

$$W_t = \begin{bmatrix} z_t & g_t & \xi_t \end{bmatrix}' \tag{25}$$

All I need to do now is tell MATLAB where to find the LinLagEx software and enter the model into the program.

To tell MATLAB where to find the software, enter the folder linlagex into you path (File-¿Set Path). Or if your model file is in a subdirectory of

```
LinLagEx_Version_xx
```

Starting with the parameter values, steady state relationships and simplifying parameter definitions:

```
%Parameters
lambda=0.75;

rho_z=0.95;
sigma_eps_z=0.71;
rho_g=0.95;
sigma_eps_g=0.6;
rho_zi=0.50;
```

```
sigma_eps_zi=0.80;

beta=0.99;
sigma=2;
phi=1.5;
v=2;
alpha=2/3;
theta=6;

%Steady States
g_bar=0.3;
zT_bar=1;
R_bar=1/beta;
y_bar=zT_bar;
c_bar=y_bar-g_bar;
sc=c_bar/y_bar;
w=phi/alpha+1/alpha-1;
xi=(w+sigma/sc)/(1+theta*w);
a_1=sc/sigma;
muz=sigma*(1+w)*(rho_z-1)/(sc*w+sigma);
a_3=sc*muz/sigma;
mug=(sigma*(rho_g-1)/sc)*((sigma*(1-sc)/(sc*w+sigma))+sc-1);
a_2=sc*mug/sigma;
psi=(w+sigma/sc)/(1+w);
gamma_g=(sigma*(1-sc)/(sc*v))*(1-(sigma/sc)/(w+sigma/sc));
```

Now, I need to name the column spaces of the model (i.e. give the variables names).

```
%Variable names
ENDOGENOUS_VARIABLE_NAMES={'Real Money Supply'
'Inflation'
'Natural Real Interest Rate'
'Output Gap'
'Nominal Interest Rate'
'Change in Output Gap'};
EXOGENOUS_VARIABLE_NAMES={'Technology'
'Government Spending'
'Money Supply'};
```

Having set parameter values and having named variables, I can enter the matrices without lagged expectations. Defining the matrices A_0, B_0, C_0, F_0, G_0,:

```
%       m    pi   rr   x    R    Dx
A_0=[   0    0    0    0    0    0
        0    0    0    0    0    0
        0    0    0    0    0    0
        0    sc/sigma  0    1    0    0
        0    0    0    0    0    0
        0    0    0    0    0    0];
%       m    pi   rr   x    R    Dx
B_0=[   -1   -1   0    0    0    0
        0    0    -1   0    0    0
        -1   0    0    sigma/(sc*v)    -1/(v*(R_bar-1))    0
        0    0    sc/sigma    -1   -sc/sigma    0
        0    -1   0    xi*(1-lambda)/lambda    0    0
        0    0    0    1    0    -1];
%       m    pi   rr   x    R    Dx
C_0=[   1    0    0    0    0    0
        0    0    0    0    0    0
        0    0    0    0    0    0
        0    0    0    0    0    0
        0    0    0    0    0    0
        0    0    0    -1   0    0];
%       z    g    zi
F_0=[   zeros(6,3)];
%       z    g    zi
G_0=[   0    0    1
        muz mug 0
        sigma/(sc*v*psi)    -gamma_g    0
        0    0    0
        0    0    0
        0    0    0];
```

Eq. (1) in the paper is thus complete with the exception of the lagged expectations from the Phillips curve.

Before the rest of the equations are entered, I need to define the name of the iteration counter (i.e. the "i" in the sum of the Phillips curve equation) and set its maximum value to 'infinity'.

```
%name of the counter used in subsequent matrices
it_name='J_1';
%maximum value of counter: either 'infinity' or a scalar (without
```

```
%quotation marks)
it_max_value='infinity';
```

Now, I can enter the rest of the matrices as MATLAB 'strings'. Noting that the counter must start at 1 (i.e. $E_{t-1}, E_{t-2}, ..., E_{t-\text{it\_max\_value}}$).

```
%counter matrices

A_j='[zeros(6,6)]';

%       m   pi  rr  x   R   Dx
B_j='[zeros(4,6); 0 (1-lambda)*lambda^(J_1-1)  0 0 0 xi*(1-lambda)*lambda^(J_1-1); zeros(1,6)]

C_j='[zeros(6,6)]';
F_j='[zeros(6,3)]';
G_j='[zeros(6,3)]';
```

Note that 'zeros(q,q)' yields a q*q matrix of zeros. One difficulty can arise: MATLAB does not allow a string to be entered over several lines. Thus the definition of B_j above must be entered in on one single line of code. This can be rather tedious at times, so here's the work around:

```
B_j=['[zeros(4,6);'...
 '0 (1-lambda)*lambda^(J_1-1)  0 0 0 xi*(1-lambda)*lambda^(J_1-1);'...
  'zeros(1,6)]'];
```

Or

```
B_j=['[zeros(4,6);'...
 '0 (1-lambda)*lambda^(J_1-1)  0 0'...
 '0 xi*(1-lambda)*lambda^(J_1-1);'...
  'zeros(1,6)]'];
```

As long as columns are separated by a semi-colon, it makes no difference where the string is split up - MATLAB reads the string as a single line.

Though not necessary if the optional MATLAB Symbolic Toolbox is installed, I then enter the limiting matrices $\tilde{A}_\infty, \tilde{B}_\infty, \tilde{C}_\infty, \tilde{F}_\infty, \tilde{G}_\infty$, to mimimze the computation time needed by my program.

```
%limit matrices
%If a counter goes to infinity, please enter the limiting summed matrices
A_inf=[A_0];
B_inf=[-1  -1  0   0   0   0
        0   0   -1  0   0   0
```

```
        -1  0   0   sigma/(sc*v)    -1/(v*(R_bar-1))    0
        0   0   sc/sigma    -1  -sc/sigma   0
        0   0  0   xi*(1-lambda)/lambda    0   xi
        0   0   0   1   0   -1];
C_inf=[C_0];
F_inf=[F_0];
G_inf=[G_0];
```

Note that as lagged expectations only entered in through the B matrices, all other matrices remain identical to their counterparts without lagged expectations.

Eq (1) from my paper is complete, I need only enter in eqs (2) and (3). The matrices N and Ω define the exogenous processes:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Define exogenous process as a VAR(1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%AR-coefficients
N=[rho_z,0,0;0,rho_g,0;0,0,rho_zi];
%Covariance matrix
Omega=[sigma_eps_z,0,0; 0,sigma_eps_g,0;0,0,sigma_eps_zi];
```

The default setting for eq (3) is to limit the model to unit-root growth and I have no reason to permit any other form of growth. Thus, nothing is entered.

Finally, I set the only real mandatory option:

```
%Options
PERIOD=4;
```

Thus defining the model as a quarterly model (needed not only for plots and the like, but for HP-filtering as well).

Finally, I tell MATLAB to call linlagex.m, the main program of my routine:

```
linlagex;
```

Note that Trabandt (2007) approximated his model with a finite number of lagged expectations as opposed to a finite number of lagged expectation errors. To obtain his solution, set it_max_value to 20 and remove the limiting matrices (both from the .m-file and, if there, the MATLAB desktop).

5 Example: Wang and Wen (2007) - Burnside Example

Wang and Wen (2006) derive an RBC-model with labor hoarding in their second example. The structural equations of the model are (in log-deviations from steady states):

$$
\begin{aligned}
0 &= -\alpha n_t + E_{t-N}\left[A_t + \lambda_t + \alpha\left(u_t + k_{t-1}\right) - \alpha e_t\right] \\
0 &= \lambda_t + c_t - \theta_t \\
0 &= A_t + (1-\alpha)\left(e_t + n_t - k_{t-1}\right) - (\phi - \alpha)\, u_t \\
0 &= A_t + \lambda_t + \alpha\left(u_t + k_{t-1}\right) - \alpha n_t - (\alpha + \pi)\, e_t \\
0 &= -\lambda_t - \eta k_t + E_t\left[\lambda_{t+1} + \frac{\eta}{1-\alpha} A_{t+1} - \eta u_{t+1} + \eta\left(e_{t+1} + n_{t+1}\right)\right] \\
0 &= A_t + \left(\alpha + s_i \frac{1-\bar{\delta}}{\bar{\delta}}\right) k_{t-1} - \frac{s_i}{\bar{\delta}} k_t - s_c c_t - s_g g_t + (\alpha - s_i * \phi)\, u_t + (1-\alpha)\left(e_t + n_t\right) \\
0 &= -y_t + A_t + \alpha k_{t-1} + \alpha u_t + (1-\alpha)\, e_t + (1-\alpha)\, n_t \\
\begin{bmatrix} A_t \\ \theta_t \\ g_t \end{bmatrix} &= \begin{bmatrix} \rho_a & 0 & 0 \\ 0 & \rho_\theta & 0 \\ 0 & 0 & \rho_g \end{bmatrix} \begin{bmatrix} A_{t-1} \\ \theta_{t-1} \\ g_{t-1} \end{bmatrix} + \begin{bmatrix} \epsilon_t^A \\ \epsilon_t^\theta \\ \epsilon_t^g \end{bmatrix}, \quad \begin{bmatrix} \epsilon_t^A \\ \epsilon_t^\theta \\ \epsilon_t^g \end{bmatrix} \sim \mathcal{N}(0, I_3)
\end{aligned}
$$

A few comments are in order here: (1) as no values were given, it is assumed that the steady states of the exogenous processes are equal to one (thus leading to the last equation in the foregoing), (2) the timing of the variable k (capital) needs to be changed in accord with the timing structure (i.e. capital used for production today is known yesterday, thus k_{t-1} is used in production, (3) labor (n_t) is known N periods in advance and can be taken out of the expectations in the first equation (ignoring the fact that labeling labor n_t doesn't strictly conform with the information structure of variables introduced in my paper).

Dividing the variables into endogenous:

$$Y_t = \begin{bmatrix} \lambda_t & u_t & k_t & e_t & n_t & c_t & y_t \end{bmatrix}' \tag{26}$$

and exogenous variables:

$$W_t = \begin{bmatrix} A_t & \theta_t & g_t \end{bmatrix}' \tag{27}$$

All I need to do now is tell MATLAB where to find the LinLagEx software and enter the model into the program.

To tell MATLAB where to find the software, enter the folder linlagex into you path (File-¿Set Path). Or if your model file is in a subdirectory of

`LinLagEx_Version_xx`

Entering the model into the program, starting with the parameter values, steady state relationships and simplifying parameter definitions:

```
%Paramters
T=1369;
xi=60;
f=324.77754855;
beta=0.99;
alpha=0.36;
rho_theta=0.9;
rho_g=0.9;
rho_a=0.9;
u_bar=1;

s_g=0.2;
delta_bar=0.025;
e_bar=fzero(@(x)log(T)-log(T-xi-x*f)-f*x/(T-xi-x*f),1);
pi=e_bar*f/(T-xi-e_bar*f);
eta=(1-beta*(1-delta_bar))*(1-alpha);
phi=(1-beta*(1-delta_bar))/(beta*delta_bar);
delta=delta_bar/(u_barˆphi);
k_y_bar=beta*alpha/(1-beta*(1-delta_bar));
s_i=delta_bar*(k_y_bar);
s_c=1-s_i-s_g;
```

Naming the column spaces of the model (i.e. give the variables names).

```
%Variable names
ENDOGENOUS_VARIABLE_NAMES={'Costate'
'Utilization'
'Capital'
'Effort'
'Employment'
'Consumption'
'Output'};
EXOGENOUS_VARIABLE_NAMES={'Productivity'
'Consumption Urge'
'Government Spending'};
```

Having set parameter values and having named variables, I can enter the matrices without lagged expectations. Defining the matrices A_0, B_0, C_0, F_0, G_0,:

```
%Matrices
%       L   u   k   e   n   c   y
A_0=[   zeros(4,7);
        1 -eta 0 eta eta 0   0
        zeros(2,7)];

%       L   u   k   e   n   c   y
B_0=[   0   0   0   0   -alpha 0 0
        1   0   0   0   0   1   0
        0   -(phi-alpha) 0 (1-alpha) (1-alpha) 0 0
        1   alpha 0 -(pi+alpha) -alpha 0 0
        -1 0    -eta   0   0   0   0
        0   (alpha-s_i*phi) -s_i/delta_bar (1-alpha) (1-alpha) -s_c 0
        0   alpha 0 (1-alpha) (1-alpha) 0 -1];
%       L   u   k   e   n   c   y
C_0=[   zeros(2,7)
        0   0   -(1-alpha)  0   0   0   0
        0   0   alpha       0   0   0   0
        zeros(1,7)
        0   0   (alpha+s_i*(1-delta_bar)/delta_bar) 0 0 0 0
        0   0   alpha   0   0   0   0];
%       A   th  g
F_0=[zeros(4,3)
        eta/(1-alpha) 0 0
    zeros(2,3)];
%       A   th  g
G_0=[   0   0   0
        0   -1  0
        1   0   0
        1   0   0
        0   0   0
        1   0   -s_g
        1   0   0];
```

Before the rest of the equations are entered, I need to define the name of the iteration counter (i.e. the "N" in the labor-hoarding equation) and set its maximum value to 50.

```
%counter=maximum_counter_value
%name of the counter used in subsequent matrices
it_name='J_1';
%maximum value of counter: either 'infinity' or a scalar (without
```

```
%quotation marks)
it_max_value=50;
```

Now, I can enter the rest of the matrices as MATLAB 'strings'.

```
%counter matrices
%       L   u   k   e   n   c   y
A_j='[zeros(7,7)]';
B_j='[zeros(7,7)]+(J_1==it_max_value)*[1 alpha 0 -alpha 0 0 0;zeros(6,7)]';
C_j='[zeros(7,7)]+(J_1==it_max_value)*[0 0 alpha 0 0 0 0;zeros(6,7)]';
F_j='[zeros(7,3)]';
G_j='[zeros(7,3)]+(J_1==it_max_value)*[1  0 0;zeros(6,3)]';
```

Note the difference in syntax here to the previous example. Unlike the sticky-information model, the labor-hoarding model has lagged expectations entering only for the single lag N. Thus, having defined the counter as J_1, if N=10, then there are no further lagged expectations after $t - 10$ (hence the idea "max_value"). But all lagged expectations enter the system with zero coefficients for any lag less than N: the term

```
(J_1==it_max_value)
```

B_j only when J_1=it_max_value (here 50). The matrix B_j is then given by:

$$B_j = \begin{cases} \text{zeros(7,7), if } j \neq 50 \\ \text{[1 alpha 0 -alpha 0 0 0;zeros(6,7)], if } j = 50 \end{cases} \tag{28}$$

As the lagged expectations do not reach back into the infinite past (i.e. only expectations lagged back N periods show up in the model's equations), no limiting matrices are or should be defined,

```
%limit matrices
%If a counter goes to infinity, please enter the limiting summed matrices
%Alternatively, the limits can be calculated symbolically.
```

Equation (1) from my paper is now fully defined, defining the exogenous processes completes eq(2),

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Define exogenous process as a VAR(1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%AR-coefficients
N=[rho_a 0 0
    0 rho_theta 0
    0 0 rho_g];
%Covariance matrix
Omega=[1,0,0;0,1,0;0,0,1];
```

The default setting for eq (3) is to limit the model to unit-root growth and I have no reason to permit any other form of growth. Thus, nothing is entered.

Finally, I set the only real mandatory option:

```
%Options
PERIOD=4;
HORIZON=60;
```

Thus defining the model as a quarterly model (needed not only for plots and the like, but for HP-filtering as well) and setting the horizon of plots to 60 periods (as the plots will otherwise be cut off before labor hoarding is over).

Finally, I tell MATLAB to call linlagex.m, the main program of my routine:

```
linlagex;
```

6 Options

The following lists the current options for linlagex.

```
it_max_value
tolerance
RUN_IMPULSE
RUN_ANTICIPATED_IMPULSE
RUN_SPECTRAL
RUN_SIMULATION
growth_restriction
PLOT_IMPULSE
PLOT_ANTICIPATED_IMPULSE
PLOT_CORRELATION
PLOT_SIMULATION
HORIZON
IMPULSE_SELECT
ANTICIPATED_PERIOD
ANTICIPATED_HORIZON
ANTICIPATED_SELECT
CORRELATION_SELECT
CORRELATION_HORIZON
grid_size
HP_LAMBDA
solution_method
DISPLAY_STD
```

it_max_value can be a scalar or a string. Every string is interpreted as 'infinity'. This option sets how far back lagged expectations go. If no value is specified by the user, the option is set to zero (no lagged expectations).

tolerance corresponds to δ in case 3 (infinite lagged expectations) of the paper and determines the convergence tolerance of including more lagged expectations. Default value is 1E-10.

RUN_IMPULSE, RUN_ANTICIPATED_IMPULSE, RUN_SPECTRAL, and RUN_SIMULATION allow impulse responses, anticipated impulse responses, frequency domain moments, and simulations to be run/calculated if they equal 1. Default is to run all four.

growth_restriction sets the upper bound on growth. Default is 1.

PLOT_IMPULSE, PLOT_ANTICIPATED_IMPULSE, PLOT_CORRELATION, and PLOT_SIMULATION allow impulse responses, anticipated impulse responses, cross- and autocorrelations, and simulations to be plotted. Default is to plot everything.

HORIZON sets the horizon out to which plots for impulse responses will be displayed. Default is 40.

IMPULSE_SELECT selects column spaces (variables) of the model to be plotted for impulse responses. Default is to plot all variables.

ANTICIPATED_PERIOD sets how many periods in advance shocks are anticipated/announced for calculating anticipated impulses. Default is 8.

ANTICIPATED_HORIZON and ANTICIPATED_SELECT are the counterparts of HORIZON and IMPULSE_SELECT for anticipated impulses.

CORRELATION_SELECT selects which variables' auto- and cross-correlations are to be calculated. Default is all.

CORRELATION_HORIZON sets the horizon of correlation calculations. Default is 6

grid_size sets the number of points to use for the discrete approximation of the spectra. Default is 64.

HP_LAMBDA sets the HP filter coefficient. Default is $1600\left(\frac{PERIOD}{4}\right)^4$ following Ravn and Uhlig (2002).

solution_method selects the solution method. If it is equal to 'QZ', the QZ method is used. If set to 'AIM', the AIM method is used. The default method is QZ.

DISPLAY_STD selects whether standard deviations (in spectral and simulation calculations) should be displayed in the command window.

References

HAMILTON, J. D. (1994): Time Series Analysis. Princeton University Press.

Klein, P. (2000): "Using the Generalized Schur Form to Solve a Multivariate Linear Rational Expectations Model," Journal of Economic Dynamics and Control, 24(10), 1405–1423.

Muth, J. F. (1961): "Rational Expectations and the Theory of Price Movements," Econometrica, 29(3), 315–335.

Ravn, M. O., and H. Uhlig (2002): "On Adjusting the Hodrick-Prescott Filter for the Frequency of Observations," The Review of Economics and Statistics, 84(2), 371–375.

Sims, C. A. (2001): "Solving Linear Rational Expectations Models," Computational Economics, 20(1-2), 1–20.

Taylor, J. B. (1986): "Econometric Approaches to Stabilization Policy in Stochastic Models of Macroeconomic Fluctuations," in Handbook of Econometrics, ed. by Z. Griliches, and M. D. Intriligator, vol. 3 of Handbook of Econometrics, chap. 34, pp. 1997–2055. Elsevier.

Trabandt, M. (2007): "Sticky Information vs. Sticky Prices: A Horse Race in a DSGE Framework," Sveriges Riksbank Working Paper Series 209, Sveriges Riksbank.

Uhlig, H. (1999): "A Toolkit for Analysing Nonlinear Dynamic Stochastic Models Easily," in Computational Methods for the Study of Dynamic Economies, ed. by R. Marimon, and A. Scott, chap. 3, pp. 30–61. Oxford University Press.

Wang, P., and Y. Wen (2006): "Solving Linear Difference Systems with Lagged Expectations by a Method of Undetermined Coefficients," Working Papers 2006-003, Federal Reserve Bank of St. Louis.