Exploiting block decomposition

Ferhat MIHOUBI *

EPEE Department of Economics University of Evry val d'Essonne

and

CEPREMAP, Dynare team

28 June 2013

Dynare Summer School

* University of Evry val d'Essonne, Department of Economics, 4 bd. F. Mitterrand, 91025 Evry, cedex ; Mail : fmihoubi@univ-evry.fr
CEPREMAP, 142 rue du Chevaleret, 75013 Paris

- The deterministic Simulation, the stochastic simulation and the estimation of large and medium scale DSGE models require a large amount of CPU time.
- One strategy to reduce the computational burden is to apply a “Divide and conquer” approach. Instead to solve or estimate the overall model, it can be split in recursive blocks of smaller size, involving a decrease of the computational burden => Block decomposition strategy.
- The purpose is to show how we can speed-up
  - The deterministic simulation
  - Stochastic simulation and estimation

  exploiting the block decomposition of large and medium scale DSGE models.

- Deterministic simulation assumes perfect foresight: all future shocks a perfectly known.
- Deterministic simulations are useful to measure the consequences of:
  - a purely temporary shock: a shock occurring only at the current period.
  - a permanent shock: for example a regime change in the monetary or the fiscal policy.
  - an announced and credible future policy change.
- The simulation can start and finish at the steady state or at any other values.

- It could be opposed to stochastic simulations where:
  - Only the distribution and the parameters characterizing the distribution are known. As a consequence, the values of future shocks are set to their expected values (no permanent shock).
  - The simulations are carried out using an approximation of the model (local or global approximation).
  - The stochastic simulations carried out with perturbation methods (linear, quadratic, or even higher order approximations) lay necessarily in the vicinity of the steady state solution.

- Several methods could be used to carry out a deterministic simulation:
  - The Fair & Taylor (1984) algorithm. This method is based on a generalization of Gauss Seidel algorithm (in two loops: one with fixed expectations, the other updating the expectations). The drawback of this algorithm is that its convergence is not warranty.
  - Shooting (or multi-shooting) methods could also be considered. But they suffer from a numerical instability problem in its basic form and the multi-shooting requires a large amount of calculation.
  - Relaxation methods as proposed by Laffargue (1990), Bouccekine (1995) and Juillard (1996). That's the method used in Dynare to compute deterministic simulations.

- We want to solve a non linear model with perfect foresight of the following form:

$$f_i(y_{t-k},\ldots,y_{t-1},y_t,y_{t+1},\ldots y_{t+l},x_t)=0$$

for i=1,…,n, t=1,…,T and for given $y_{-k+1},\ldots,y_0$ and $y_{T+1},\ldots,y_{T+l}$.

where y_t denote endogenous variables and x_t the exogenous variables.

- This model could rewritten using intermediates variables in a more compact form:

$$f_i(y_{t-1},y_t,y_{t+1},x_t)=0$$

- The principle of the relaxation method is to solve the stacked time system: the system composed of *n*x*T* equations:

$$
\begin{cases}
\left.\begin{array}{l} f_1(y_0, y_1, y_2, x_1) = 0 \\ \vdots \\ f_n(y_0, y_1, y_2, x_1) = 0 \end{array}\right\} \text{time } 1 \\
\left.\begin{array}{l} f_1(y_1, y_2, y_3, x_2) = 0 \\ \vdots \\ f_n(y_1, y_2, y_3, x_2) = 0 \end{array}\right\} \text{time } 2 \\
\vdots \\
\left.\begin{array}{l} f_1(y_{T-1}, y_T, y_{T+1}, x_T) = 0 \\ \vdots \\ f_n(y_{T-1}, y_T, y_{T+1}, x_T) = 0 \end{array}\right\} \text{time } T
\end{cases}
$$

Because *f* is potentially non-linear we can use the Newton Algorithm to solve the overall system

- Newton algorithm: the linear approximations of the model are successively solved until convergence.

Newton algorithm to solve the nonlinear system: $f(Y)=0$

Step 1 : form an initial guess $Y=Y^0$ and set k=0

Step 2 : solve for Y^{k+1} the linear approximation around Y^k :

$$\lambda . f(Y^k) + J(Y^k)\left[Y^{k+1} - Y^k\right] = 0$$

with $J\left(Y^k\right)=\frac{\partial f\left(Y^k\right)}{\partial Y}$ and λ the path length.

Step 3 : if convergence (ie. $f(Y^{k+1})' f(Y^{k+1}) < tol$) then stop

else k=k+1 and return to step 2.

- In the step 2 the linear approximation

$$\lambda . f(Y^k) + J(Y^k)\left[Y^{k+1} - Y^k\right] = 0$$

could be solved inverting the Jacobian matrix:

$$Y^{k+1} = Y^k - \lambda . \left[J\left(Y^k\right)\right]^{-1} f\left(Y^k\right)$$

=> Very bad idea: the Jacobian matrix $J\left(Y^k\right)$ is very sparse in our case.

- For our model: $f_i(y_{t-1}, y_t, y_{t+1}, x_t) = 0$, if we defined: $A_t = \frac{\partial f}{\partial y_{t-1}}$, $B_t = \frac{\partial f}{\partial y_t}$ and $C_t = \frac{\partial f}{\partial y_{t+1}}$ the Jacobian matrix becomes:

$$J = \begin{bmatrix} B_1 & C_1 & 0 & \cdots & \cdots & 0 \\ A_2 & B_2 & C_2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & A_T & B_{T-1} & C_{T-1} \\ 0 & \cdots & \cdots & 0 & A_T & B_T \end{bmatrix}$$

- The linear system to solve in step 2 could be rewritten as:

$$\begin{bmatrix} B_1 & C_1 & 0 & \cdots & \cdots & 0 \\ A_2 & B_2 & C_2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & A_T & B_{T-1} & C_{T-1} \\ 0 & \cdots & \cdots & 0 & A_T & B_T \end{bmatrix} \begin{bmatrix} y_1^{k+1} - y_1^k \\ y_2^{k+1} - y_2^k \\ y_3^{k+1} - y_3^k \\ \vdots \\ y_{T-1}^{k+1} - y_{T-1}^k \\ y_T^{k+1} - y_T^k \end{bmatrix} = - \begin{bmatrix} f\left(y_0, y_1^k, y_2^k, x_1\right) + A_1 y_0 \\ f\left(y_1^k, y_2^k, y_3^k, x_2\right) \\ f\left(y_2^k, y_3^k, y_4^k, x_3\right) \\ \vdots \\ f\left(y_{T-2}^k, y_{T-1}^k, y_T^k, x_{T-1}\right) \\ f\left(y_{T-1}^k, y_T^k, y_{T+1}, x_T\right) + C_T y_{T+1} \end{bmatrix}$$

Or equivalently:

$$J.Y = b$$

- The J is triangularised using Laffargue (1990) algorithm which takes into account the sparsity of J.

$$\begin{bmatrix} I & S_1 & 0 & \cdots & \cdots & 0 \\ 0 & B_2 - A_2 S_1 & C_2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & A_T & B_{T-1} & C_{T-1} \\ 0 & \cdots & \cdots & 0 & A_T & B_T \end{bmatrix} \begin{bmatrix} y_1^{k+1} - y_1^k \\ y_2^{k+1} - y_2^k \\ y_3^{k+1} - y_3^k \\ \vdots \\ y_{T-1}^{k+1} - y_{T-1}^k \\ y_T^{k+1} - y_T^k \end{bmatrix} = - \begin{bmatrix} d_1 \\ f\left(y_1^k, y_2^k, y_3^k, x_2\right) + A_2 d_1 \\ f\left(y_2^k, y_3^k, y_4^k, x_3\right) \\ \vdots \\ f\left(y_{T-2}^k, y_{T-1}^k, y_T^k, x_{T-1}\right) \\ f\left(y_{T-1}^k, y_T^k, y_{T+1}, x_T\right) + C_T y_{T+1} \end{bmatrix}$$

With: $S_1 = B_1^{-1} C_1$ and $d_1 = B_1^{-1}\left[f\left(y_0^k, y_1^k, y_2^k, x_1\right) + A_1 y_0 \right]$

- The J is triangularised using Laffargue (1990) algorithm which takes into account the sparsity of J.

$$\begin{bmatrix} I & S_1 & 0 & \cdots & \cdots & 0 \\ 0 & I & S_2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & A_T & B_{T-1} & C_{T-1} \\ 0 & \cdots & \cdots & 0 & A_T & B_T \end{bmatrix} \begin{bmatrix} y_1^{k+1} - y_1^k \\ y_2^{k+1} - y_2^k \\ y_3^{k+1} - y_3^k \\ \vdots \\ y_{T-1}^{k+1} - y_{T-1}^k \\ y_T^{k+1} - y_T^k \end{bmatrix} = - \begin{bmatrix} d_1 \\ d_2 \\ f\left(y_2^k, y_3^k, y_4^k, x_3\right) + A_3 d_2 \\ \vdots \\ f\left(y_{T-2}^k, y_{T-1}^k, y_T^k, x_{T-1}\right) \\ f\left(y_{T-1}^k, y_T^k, y_{T+1}, x_T\right) + C_T y_{T+1} \end{bmatrix}$$

With: $S_1 = B_1^{-1} C_1$ and $d_1 = B_1^{-1}\left[f\left(y_0^k, y_1^k, y_2^k, x_1\right) + A_1 y_0 \right]$

$$S_2 = \left[B_2 - A_2 S_1\right]^{-1} C_2 \text{ and } d_2 = \left[B_2 - A_2 S_1\right]^{-1}\left[f\left(y_1^k, y_2^k, y_3^k, x_2\right) + A_2 d_1 \right]$$

- The J is triangularised using Laffargue (1990) algorithm which takes into account the sparsity of J.

$$\begin{bmatrix} I & S_1 & 0 & \cdots & \cdots & 0 \\ 0 & I & S_2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 0 & I & S_{T-1} \\ 0 & \cdots & \cdots & 0 & 0 & I \end{bmatrix} \begin{bmatrix} y_1^{k+1} - y_1^k \\ y_2^{k+1} - y_2^k \\ y_3^{k+1} - y_3^k \\ \vdots \\ y_{T-1}^{k+1} - y_{T-1}^k \\ y_T^{k+1} - y_T^k \end{bmatrix} = - \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{T-1} \\ d_T \end{bmatrix}$$

With: $S_1 = B_1^{-1} C_1$ and $d_1 = B_1^{-1} \left[f\left(y_0^k, y_1^k, y_2^k, x_1\right) + A_1 y_0 \right]$

$S_2 = \left[B_2 - A_2 S_1\right]^{-1} C_2$ and $d_2 = \left[B_2 - A_2 S_1\right]^{-1} \left[f\left(y_1^k, y_2^k, y_3^k, x_2\right) + A_2 d_1 \right] \ldots$

$\ldots S_T = \left[B_T - A_T S_{T-1}\right]^{-1} C_T$ and $d_T = \left[B_T - A_T S_{T-1}\right]^{-1} \left[f\left(y_{T-1}^k, y_T^k, y_{T+1}^k, x_T\right) + C_T y_{T+1} \right].$

- The solution of linearized system is computed recursively using a backward substitution:

$$\begin{cases} y_T^{k+1} = y_T^k - d_T \\ y_{T-1}^{k+1} = y_{T-1}^k - d_{T-1} - S_{T-1}\left(y_T^{k+1} - y_T^k\right) \\ \vdots \\ \vdots \\ y_2^{k+1} = y_2^k - d_2 - S_2\left(y_3^{k+1} - y_3^k\right) \\ y_1^{k+1} = y_1^k - d_1 - S_1\left(y_2^{k+1} - y_2^k\right) \end{cases}$$

- Juillard (1996) extends the relaxation method to the general model without using intermediate variables:

$$f_i(y_{t-k},\ldots,y_{t-1},y_t,y_{t+1},\ldots y_{t+l},x_t) = 0$$

Practical implementation in dynare: a temporary shock at time 1

```
var k, c;
varexo x;
parameters alph gam delt bet aa;
alph=0.5; gam=0.5; delt=0.02; bet=0.05; aa=0.5;

model;
c + k - aa*x*k(-1)^alph - (1-delt)*k(-1);
c^(-gam) - (1+bet)^(-1)*(aa*alph*x(+1)*k^(alph-1) + 1 - delt)*c(+1)^(-gam);
end;

initval;
x = 1;
k = ((delt+bet)/(1.0*aa*alph))^(1/(alph-1));
c = aa*k^alph-delt*k;
end;

steady;

shocks;
var x;
periods 1;
values 1.2;
end;
simul(periods=200);
```

Practical implementation in dynare: a permanent shock that occurs at time 1, use endval.

```
initval;
x = 1;
k = ((delt+bet)/(1.0*aa*alph))^(1/(alph-1));
c = aa*k^alph-delt*k;
end;

steady;

endval;
x = 1.2;
k = ((delt+bet)/(1.0*aa*alph))^(1/(alph-1));
c = aa*k^alph-delt*k;
end;

steady;

simul(periods=200);
```

Practical implementation in dynare: a permanent shock that occurs at time 10 combine endval and shocks:

```
initval;
x = 1;
k = ((delt+bet)/(1.0*aa*alph))^(1/(alph-1));
c = aa*k^alph-delt*k;
end;
steady;

endval;
x = 1.2;
k = ((delt+bet)/(1.0*aa*alph))^(1/(alph-1));
c = aa*k^alph-delt*k;
end;
steady;

shocks;
var x;
periods 1:9;
value 1;
end;

simul(periods=200);
```

Some recommendations:

- Because we face “a two-point boundary value problem” (given by y_0 and y_{T+1}), the simulation periods has to be long enough to be sure that transition is completely achieved at time *T* (take a look at what happen at the end of the simulation).
- In order to get the steady-state (using steady command) you have to set the endogenous variables in initval section close to their steady-state’s values. The same remark also applies for endval.

Block decomposition to speed up deterministic simulations

- To reduce the size of the stacked-time problem a block decomposition of the model could be performed (a block decomposition of the Jacobian of the steady-state model A*=A+B+C):
- Some parts of medium/large scale models are purely recursive and could be simply evaluated instead of being iteratively solved.
  - In some case the remaining equations of the model could be split in several simultaneous blocks (i.e. a two-country model with a large and a small country and no feedback effect).
- Only the blocks containing leads and lags endogenous variables are solved with the stacked-time representation $O\left[(T.n)^3\right]$.The other ones are solved $T.O\left[n^3\right]$ or evaluated at each period.

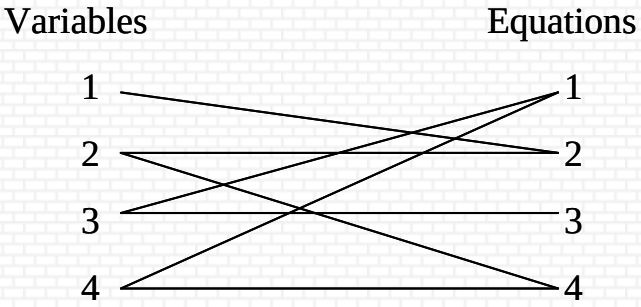
Normalization

- To get a block decomposition of the model, each endogenous variable has to be assigned unambiguously to an equation. In addition, to avoid singularity problems, this affectation has to pay attention to the weight of each variable in an equation (the value of the Jacobian matrix components).

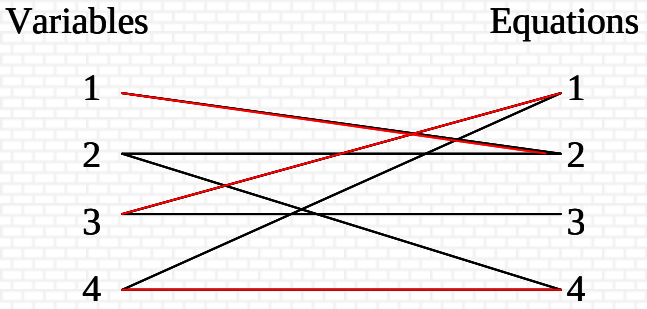
- The normalization of equations can be carried out using a matching method on a bipartite graph (equation-endogenous variable). In a first step, the variables are arbitrarily assigned to equations. In a second step, normalization is improved, trying to assign the unassigned equations to the remaining variables. The second step is repeated until all endogenous variables are assigned or no augmenting path is found. In the first case, the normalization of the model is achieved.

- For the following model, we get the bipartite graph:

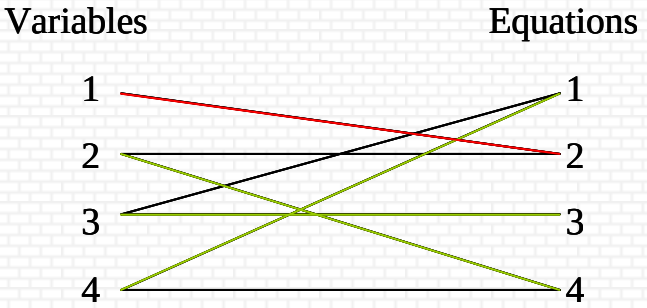
$$f_1(y_3, y_4) = 0$$
$$f_2(y_1, y_2) = 0$$
$$f_3(y_3) = 0$$
$$f_4(y_2, y_4) = 0$$



- We get a first cheap matching:



- There is an augmenting path with the following assignment:



- To get a non singular system, the previous normalization process is applied iteratively using the normalized Jacobian matrix (each row is divided by its maximum absolute value component) with a decreasing cutoff (all the elements in the Jacobian matrix below the cutoff are set to zero) until a normalisation is found.

Finding the recursive equations of the model

- Once the model is normalized, the purely recursive equations are easily found. The equations and the endogenous variables are reordered to get an incidence matrix of the long run model with an initial and a final triangular block.

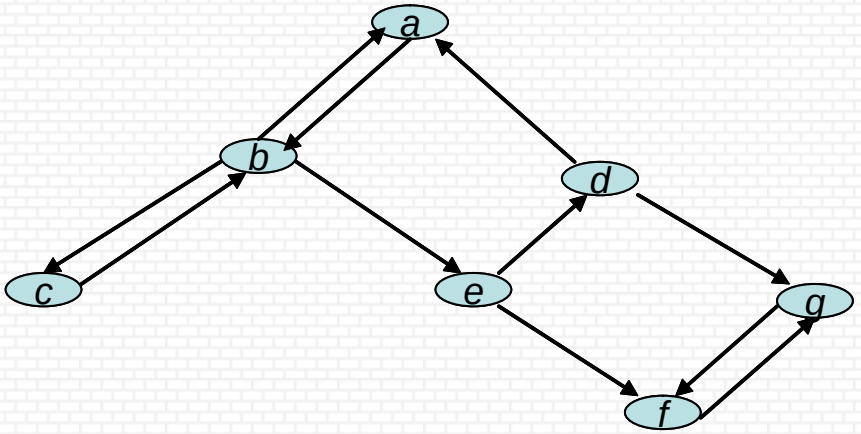
- The first block, called prologue, is composed of equations involving only exogenous variables or endogenous variables determined by previous equations.
- The last block, the epilogue, is composed of variables which are pure output in the model: they do not appear in the previous equations of the model.

The incidence matrix of the model has then the following form:

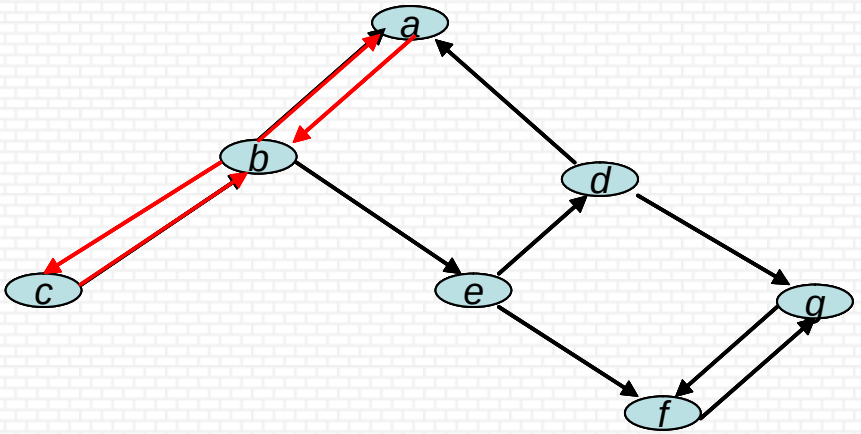
$$\begin{cases} k = f_1(i) \\ l = f_2(y,k) \\ c = f_3(r) \\ \pi = f_4(y) \\ i = f_5(y,r) \\ g = f_6(\bar{g}) \\ y = f_7(c,i,g) \\ r = f_8(\pi,y) \end{cases}$$

	g	*y*	*i*	*c*	*r*	π	*k*	*l*
6	1	0	0	0	0	0	0	0
7	1	1	1	1	0	0	0	0
5	0	1	1	0	1	0	0	0
3	0	0	0	1	1	0	0	0
8	0	1	0	0	1	1	0	0
4	0	1	0	0	0	1	0	0
1	0	0	1	0	0	0	1	0
2	0	1	0	0	0	0	1	1

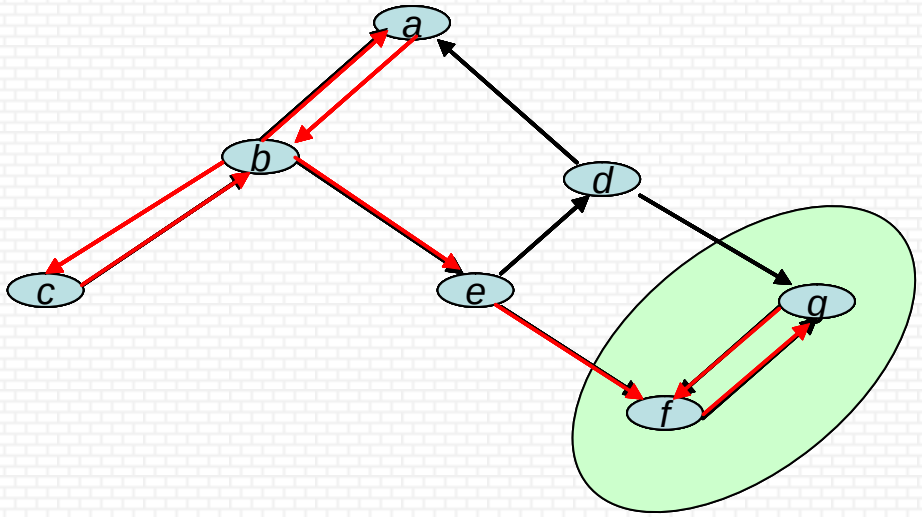
- The simultaneous block composed of equations that do not belong to the prologue or the epilogue can be split in sub recursive blocks. These recursive blocks are also the strong components (briefly speaking distinct circuits) of the long run model graph. They could be found using the depth search Tarjan Algorithm.



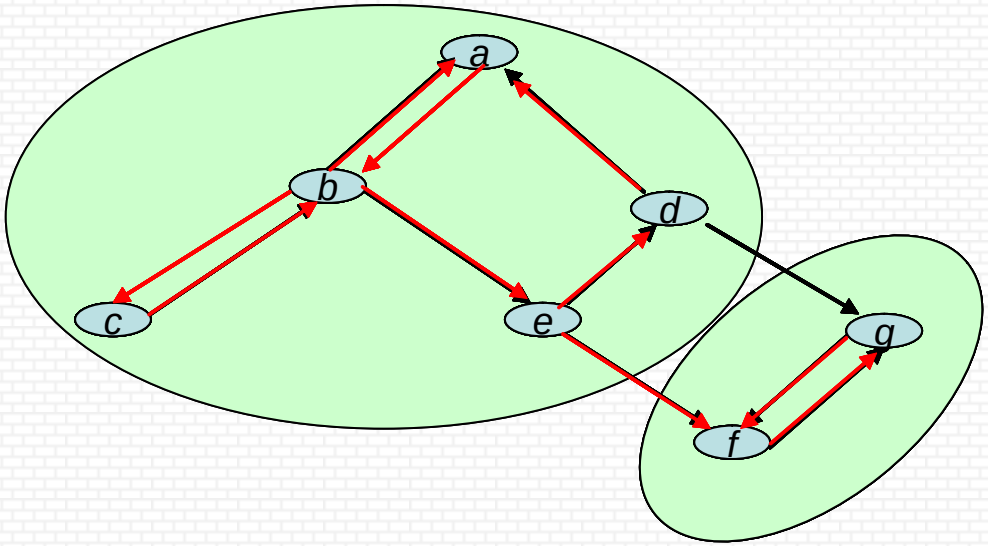
- The model representation in a block recursive form
  - The simultaneous block composed of equations that do not belong to the prologue or the epilogue could be split in sub recursive blocks. These recursive blocks are also the strong components (briefly speaking distinct circuits) of the long run model graph. They could be found using the depth search Tarjan Algorithm.



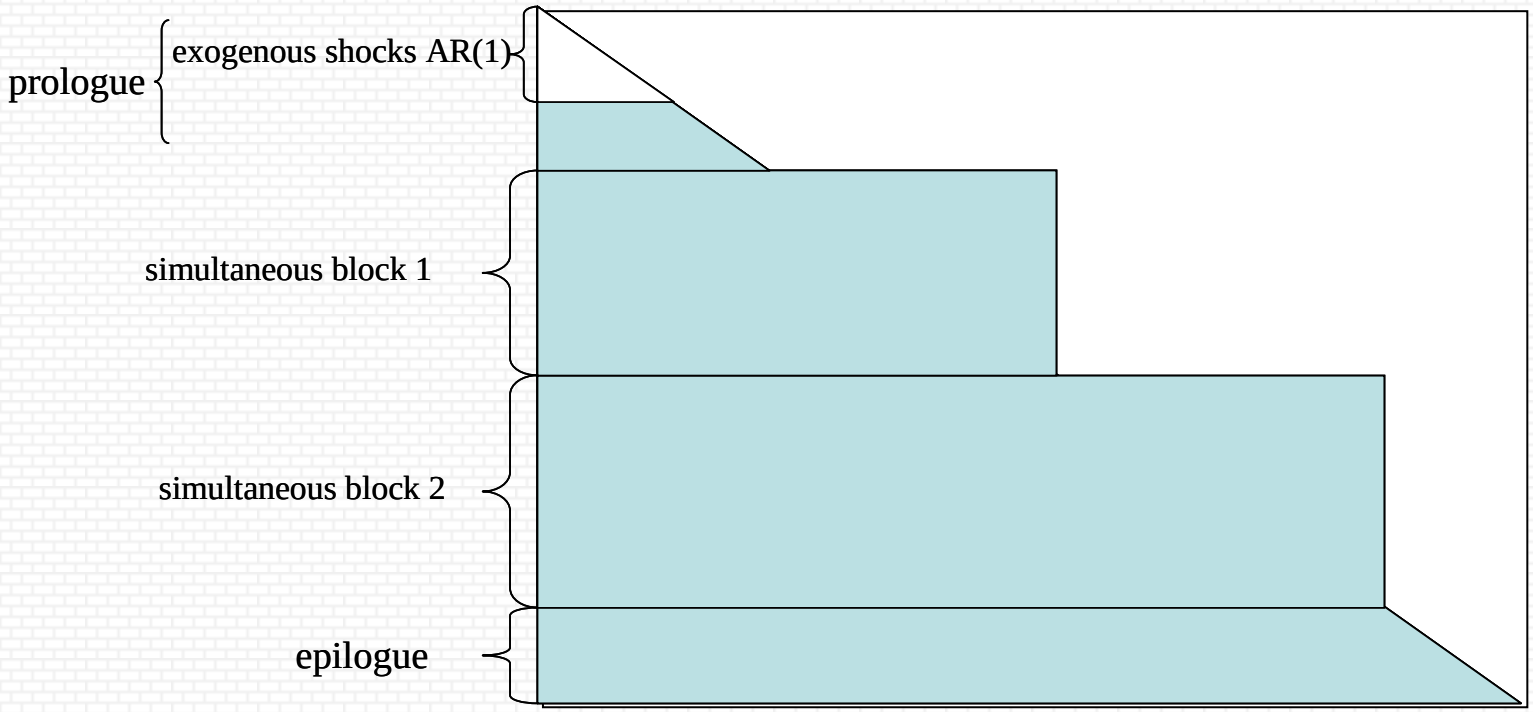
- The model representation in a block recursive form
  - The simultaneous block composed of equations that do not belong to the prologue or the epilogue could be split in sub recursive blocks. These recursive blocks are also the strong components (briefly speaking distinct circuits) of the long run model graph. They could be found using the depth search Tarjan Algorithm.



- The model representation in a block recursive form
  - The simultaneous block composed of equations that do not belong to the prologue or the epilogue could be split in sub recursive blocks. These recursive blocks are also the strong components (briefly speaking distinct circuits) of the long run model graph. They could be found using the depth search Tarjan Algorithm.



- The General form of the reordered Jacobian matrix

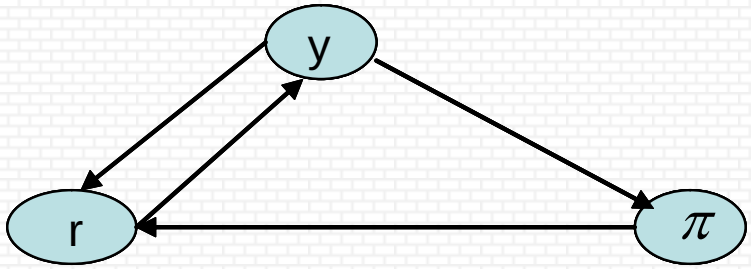


- The linear problem to solve at each step of the Newton for a simultaneous block could be reduced taking advantage of the feedback variables.

- If we consider the following model

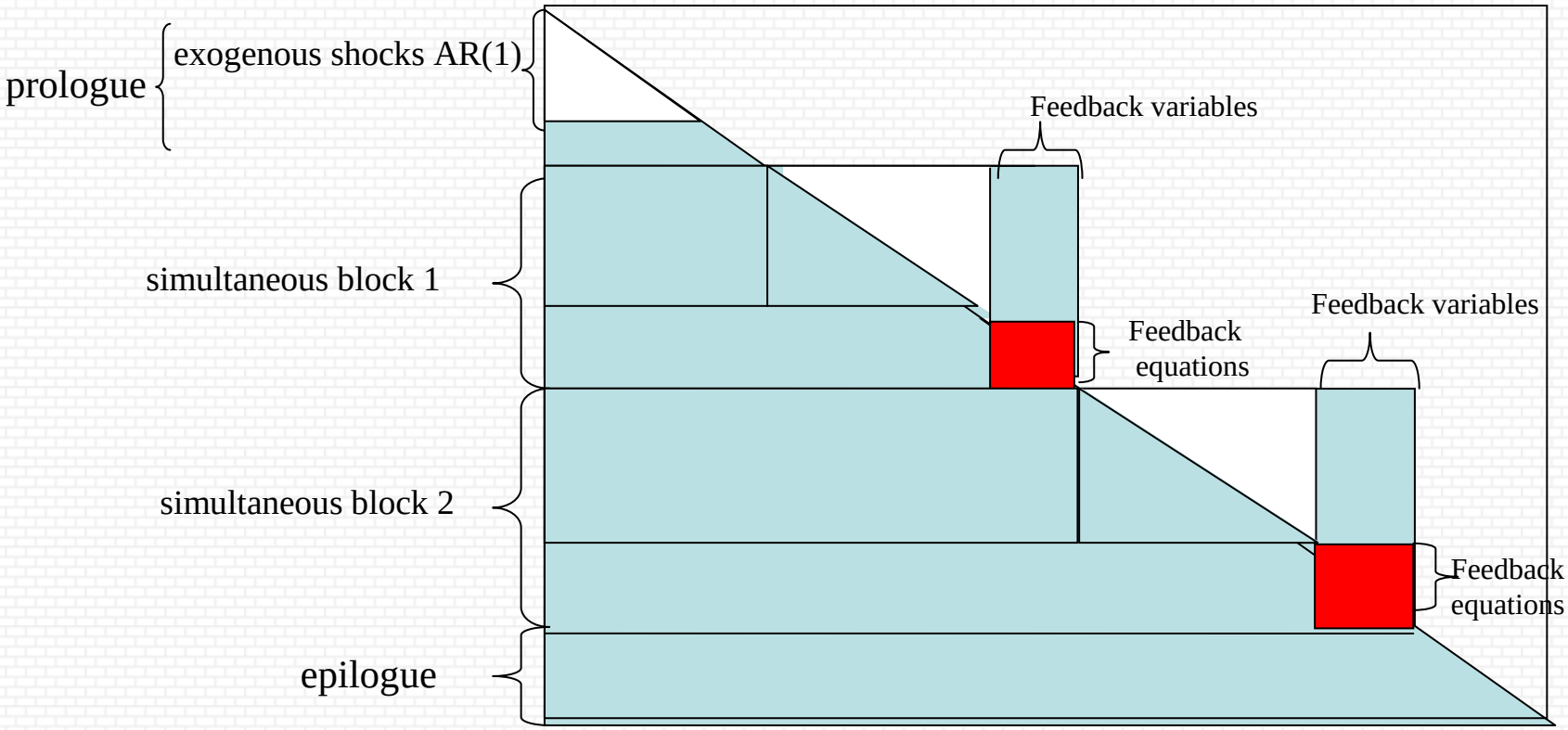
$$\begin{cases} y_t = \alpha_1 r_t + \varepsilon_t^g \\ \pi_t = \varphi y_t + \varepsilon_t^\pi \\ r_t = \mu \pi_t + \tau y_t \end{cases}$$

- If we rewrite the simultaneous block in terms of a directed graph we get:



Knowing *r* (or *y)* we can compute recursively *y* and π (*r* and π). *r* or *y* is the feedback variable. The block could be solved only for this feedback variable; the other variables are computed as a sub product of the block evaluation.

The General form of the reordered Jacobian matrix



Evaluation: Simulation time on an Intel double Core 6600 2.4Ghz with 3Go RAM

		OLG model 80 periods	Multimod Mark III 80 periods
without block and bytecode	time /iteration	20.63	88
	iterations	46	no convergence
	Total time	949	no convergence
with block	time /iteration	21	42
	iterations	7	4
	Total time	149	166
with block and bytecode	time /iteration	2	30
	iterations	6	4
	Total time	13	120

Exploiting the block decomposition to speed-up the stochastic simulation and estimation

- The Bayesian estimation requires a considerable amount of evaluation of the likelihood especially during the computation of the posterior distribution mode and the posterior distribution using MCMC.
- At each evaluation of the likelihood the four following steps are performed (three first steps for stochastic simulation):
  i. Solve the deterministic steady-state;
  ii. Check the Blanchard and Kahn condition;
  iii. Compute the first order approximation of the DSGE;
  iv. Calculate the likelihood using a Kalman filter.
- Examine whether each step can be speed up using block decomposition in large scale models (multi-country models, models estimated with an optimal monetary or fiscal policy …)

- We consider the following rational expectation model:

$$E_t\left[f\left(y_{t+1}, y_t, y_{t-1}, u_t\right)\right] = 0$$

- To describe the block decomposition of the model, we will consider its linearized form:

$$A\hat{y}_{t-1} + B\hat{y}_t + CE_t\left[\hat{y}_{t+1}\right] + HE_t\left[u_t\right] = 0 \quad (1)$$

- As we want to construct a recursive block structure where the variables of one block has no feedback effect on the variables of the previous blocks whatever the periods considered, we have to consider the block decomposition of the Jacobian matrix of the deterministic steady-state model:

$$D^* = A + B + C$$

- To check the Blanchard and Kahn's conditions, the contemporaneous endogenous variables have to be split in three sets and the Jacobian matrix B of the linear approximation (1) in three components:

$$A^1\hat{y}_{t-1}^1 + \left(B^0|B^1|B^2\right)\left[\hat{y}_t^0\hat{y}_t^1\hat{y}_t^2\right] + C^2E_t\left[\hat{y}_{t+1}^2\right] = 0 \quad (2)$$

- The system could be reduced by eliminating the purely static variables $\hat{y}_t^0$. To do it, a QR decomposition of the B^0 matrix is performed:

$$B^0 = QR$$

with Q an orthogonal matrix and R an upper triangular matrix.
The reduced system becomes:

$$\tilde{A}^1\hat{y}_{t-1}^1 + \left(\tilde{B}^1|\tilde{B}^2\right)\left[\hat{y}_t^1\hat{y}_t^2\right] + \tilde{C}^2E_t\left[\hat{y}_{t+1}^2\right] = 0 \quad (3) \qquad \text{with } \tilde{A}^1 = QA^1 \ldots$$

- A part of this reduction step can be performed, once for all, using the feedback variables computed under the constraint that all the dynamic variables have to belong to the set of feedback variables

- The dynamic system could be rewritten as:

$$\begin{bmatrix} \tilde{C}^2 & \tilde{B}^1 \\ 0 & I^1 \end{bmatrix} \begin{bmatrix} E_t\left[\hat{y}_{t+1}^2\right] \\ \hat{y}_t^1 \end{bmatrix} = \begin{bmatrix} -\tilde{B}^2 & -\tilde{A}^1 \\ I^2 & 0 \end{bmatrix} \begin{bmatrix} \hat{y}_t^2 \\ \hat{y}_{t-1}^1 \end{bmatrix}$$

$$\Leftrightarrow F \begin{bmatrix} E_t\left[\hat{y}_{t+1}^2\right] \\ \hat{y}_t^1 \end{bmatrix} = G \begin{bmatrix} \hat{y}_t^2 \\ \hat{y}_{t-1}^1 \end{bmatrix} \qquad (4)$$

- The generalised eigenvalues of this dynamic system are computed using a Schur decomposition of the pencil (*F*, *G*).

- This procedure is applied only for the simultaneous blocks. For the other dynamics blocks purely recursive, the eigenvalues are straightforward computed using the diagonal terms of the dynamic Jacobian matrix.

Table 1: Computation time of the steady-state and the Blanchard and Kahn's conditions with and without block decomposition

		Without block decomposition	With block decomposition
Eagle	Size of the biggest block[1]	965	560
	Blanchard and Kahn conditions (seconds)	23	7.5
	Steady state (seconds)	0.10	0.08
GIMF	Size of the biggest block	2032	903
	Blanchard and Kahn conditions (seconds)	258	45
	Steady state (seconds)	0.11	0.10
Overlapping generations model	Size of the biggest block	1086	441
	Blanchard and Kahn conditions (seconds)	91	22
	Steady state (seconds)	0.27	0.13

First order perturbation method applied to a block-decomposed model

- In case of a block decomposed model, we have for a block *b*:

$$E_t\left[f_b(y_{t+1}, y_t, y_{t-1}, u_t, x_{t+1}, x_t, x_{t-1})\right] = 0 \quad (5)$$

with x_t the endogenous variables from the previous blocks and y_t the endogenous variables of the current block.

- The RE solution of the previous blocks is supposed to be computed and for the endogenous variables determined in the previous and appearing in the block *b*:

$$x_t = l_b\left(x_{t-1}, u_t\right)$$

- We want to compute the RE solution of the current block:

$$y_t = g_b(y_{t-1}, u_t, x_{t-1})$$

- To do so, we first compute the following function, where the expected values of y and x are replaced by their RE solutions:

$$F_b\left(y_{t-1}, u_t, u_{t+1}, x_t, x_{t-1}\right)$$

$$= f_b\left(g_b\left(g_b\left(y_{t-1}, u_t, x_t\right), u_{t+1}, l_b\left(x_t, u_{t+1}\right)\right), g_b\left(y_{t-1}, u_t, x_t\right), y_{t-1}, u_t, l_b\left(l_b\left(x_{t-1}, u_t\right), u_{t+1}\right), l_b\left(x_{t-1}, u_t\right), x_{t-1}\right)$$

- The first order expansion of (5) is

$$
\begin{aligned}
E_t\left[F_b\left(\mathrm{y}_{t-1},\mathrm{u}_t,\mathrm{u}_{t+1},\mathrm{x}_t,\mathrm{x}_{t-1}\right)\right] \approx E_t\Big[& f_b\left(\bar{\mathrm{y}},\bar{\mathrm{y}},\bar{\mathrm{y}},0,\bar{\mathrm{x}},\bar{\mathrm{x}},\bar{\mathrm{x}}\right) \\
& + A\left(g_y\left(g_y\hat{y}_{t-1}+g_x\hat{x}_{t-1}+g_u u_t\right)+g_u u_{t+1}+g_x\left(l_x\hat{x}_{t-1}+l_u u_t\right)\right) \\
& + B\left(g_y\hat{y}_{t-1}+g_x\hat{x}_{t-1}+g_u u_t\right)+C\hat{y}_{t-1}+f_u u_t \;+ f_{x_{t+1}}\left(l_x\left(l_x\hat{x}_{t-1}+l_u u_t\right)+l_u u_{t+1}\right) \\
& + f_{x_t}\left(l_x\hat{x}_{t-1}+l_u u_t\right)+f_{x_{t-1}}l_{sx}\hat{x}_{t-1}\Big] \\
= & f\left(\bar{\mathrm{y}},\bar{\mathrm{y}},\bar{\mathrm{y}},0,\bar{\mathrm{x}},\bar{\mathrm{x}},\bar{\mathrm{x}}\right) \\
& + \left(Ag_yg_y+Bg_y+C\right)\hat{y}_{t-1} \\
& + \left[Ag_yg_u+Bg_u+f_u+f_{x_{t+1}}\left(l_xl_u\right)+f_{x_t}l_u+Ag_xl_u\right]u_t \\
& + \left[A\left(g_yg_x+g_xl_x\right)+Bg_x+\left(f_{x_t}+f_{x_{t+1}}l_x\right)l_x+f_{x_{t-1}}l_{sx}\right]\hat{x}_{t-1}
\end{aligned}
$$

with $A=\dfrac{\partial f_b\left(\bar{\mathrm{y}},\bar{\mathrm{y}},\bar{\mathrm{y}},0,\bar{\mathrm{x}},\bar{\mathrm{x}},\bar{\mathrm{x}}\right)}{\partial y_{t+1}}$, $B=\dfrac{\partial f_b\left(\bar{\mathrm{y}},\bar{\mathrm{y}},\bar{\mathrm{y}},0,\bar{\mathrm{x}},\bar{\mathrm{x}},\bar{\mathrm{x}}\right)}{\partial y_t}$, $C=\dfrac{\partial f_b\left(\bar{\mathrm{y}},\bar{\mathrm{y}},\bar{\mathrm{y}},0,\bar{\mathrm{x}},\bar{\mathrm{x}},\bar{\mathrm{x}}\right)}{\partial y_{t-1}}$, $f_u=\dfrac{\partial f_b\left(\bar{\mathrm{y}},\bar{\mathrm{y}},\bar{\mathrm{y}},0,\bar{\mathrm{x}},\bar{\mathrm{x}},\bar{\mathrm{x}}\right)}{\partial u_t}$,

$f_{x_t}=\dfrac{\partial f_b\left(\bar{\mathrm{y}},\bar{\mathrm{y}},\bar{\mathrm{y}},0,\bar{\mathrm{x}},\bar{\mathrm{x}},\bar{\mathrm{x}}\right)}{\partial x_{t+1}}$, $f_{x_t}=\dfrac{\partial f_b\left(\bar{\mathrm{y}},\bar{\mathrm{y}},\bar{\mathrm{y}},0,\bar{\mathrm{x}},\bar{\mathrm{x}},\bar{\mathrm{x}}\right)}{\partial x_t}$, $f_{x_{t-1}}=\dfrac{\partial f_b\left(\bar{\mathrm{y}},\bar{\mathrm{y}},\bar{\mathrm{y}},0,\bar{\mathrm{x}},\bar{\mathrm{x}},\bar{\mathrm{x}}\right)}{\partial x_{t-1}}$, $l_x=\dfrac{\partial l_b\left(\bar{x},0\right)}{\partial x_{t-1}}$,

$l_u=\dfrac{\partial l_b\left(\bar{x},0\right)}{\partial u_t}$, $g_u=\dfrac{\partial g_b\left(\bar{y},0,\bar{x},\bar{x}\right)}{\partial u_t}$ and $g_x=\dfrac{\partial g_b\left(\bar{y},0,\bar{x},\bar{x}\right)}{\partial x_{t-1}}$

- The RE solution imposes the three following conditions:

$$\begin{cases} Ag_y g_y + Bg_y + C = 0 & (6) \\ A\left(g_y g_x + g_x l_x\right) + Bg_x + \left(f_{x_t} + f_{x_{t+1}} l_x\right) l_x + f_{x_{t-1}} = 0 & (7) \\ Ag_y g_u + Bg_u + f_u + f_{x_{t+1}}\left(l_x l_u\right) + \left(f_{x_t} + Ag_x\right) l_u = 0 & (8) \end{cases}$$

  - g_y is recovered from the equation (6). Using a Schur decomposition of the pencil (F,G) and excluding the explosive trajectories we get the RE solution in g_y (Collard and Juillard (2001)).
  - Knowing g_y, we get g_x from equation (7) which is a Sylvester equation:

$$\left[Ag_y + B\right] g_x + Ag_x l_x = -\left[\left(f_{x_t} + f_{x_{t+1}} l_x\right) l_x + f_{x_{t-1}}\right]$$

    g_x is then the solution of:

$$\left\{I \otimes \left[Ag_y + B\right] + l_x' \otimes A\right\} vec\left(g_x\right) = -vec\left(\left(f_{x_t} + f_{x_{t+1}} l_x\right) l_x + f_{x_{t-1}}\right)$$

    or more efficiently, using a cyclic reduction algorithm.
  - knowing g_y and g_x, we recover directly g_u from equation (8):

$$g_u = \left[Ag_y + B\right]^{-1}\left[f_u + f_{x_{t+1}}\left(l_x l_u\right) + \left(f_{x_t} + Ag_x\right) l_u\right]$$

- The gain related to the block decomposition is weaker than the complexity reduction for the Blanchard and Kahn conditions, because of the additional specific cost induced by the computation of g_x for a block decomposed model
- The implementation of the first order approximation on a block decomposed model remains useful as soon as the evaluation of the likelihood using a Kalman filter takes advantage of the block triangular form of the RE solution.
- Evaluation of the block decomposition to estimate the mode of the Smets & Wouters model[2] in minutes

		Block	
		Without	With
bytecode	Without	9.26	5.51
	With	8.1	4.2

[2] "Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach", AER 2007

The Block Kalman filter

- The general idea of Block Kalman Filter (Strid and Walentin 2009) is to take advantage of the block structure of the model to reduce the number of operations involved in the likelihood evaluation with a Kalman filter.

- Consider the state-space representation of a DSGE:

$$\mathrm{Y}_t = d(\theta) + ZX_t + v_t \qquad \text{measurement equation}$$

$$X_t = c + T(\theta)X_{t-1} + R(\theta)\varepsilon_t \qquad \text{state equation}$$

In our case $T(\theta) = g_y = l_x$ has a block triangular form and $R(\theta) = g_u$.

- The traditional Kalman Filter (without block decomposition) is composed of the following two steps:
  - The updating equations:

$$X_{t|t} = E\left[X_t \middle| Y_1,\ldots,Y_t\right] = X_{t|t-1} + K_t F_t^{-1}\left(Y_t - ZX_{t|t-1} - d\right)$$

$$P_{t|t} = P_{t|t-1} - K_t F_t^{-1} K_t'$$

with $K_t = P_{t|t-1}Z'$ and $F_t = V\left[Y_t \middle| Y_1,\ldots,Y_{t-1}\right] = ZK_t + H$

  - The prediction equations

$$X_{t+1|t} = TX_{t|t} + c$$

$$P_{t+1|t} = TP_{t|t}T' + RQR'$$

The likelihood is simply:

$$\log L(\theta) = -\frac{NT}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{T}\log\left(\left|F_t\right|\right) - \frac{1}{2}\sum_{t=1}^{T}\hat{v}_t' F_t^{-1}\hat{v}_t$$ with $\hat{v}_t = Y_t - ZX_{t|t-1}$.

The most expensive step in CPU time is the computation of the covariance matrix in the prediction step $P_{t+1|t}$ (about the half time).

- For a block recursive *T* matrix, only the lower block triangular terms of the matrix *T* have to be considered in the product (the upper triangular terms are equal to 0). Since the block triangular form is constant throughout the estimation process, the block structure is hard coded in the matrix multiplication.
- The covariance matrix $P_{t|t}$ implied in the expression is symmetric. This feature is also be taken into account to reduce the computational burden.

- Practical implementation of block decomposition in dynare:

```
var k, c;
varexo x;
parameters alph gam delt bet aa;
alph=0.5; gam=0.5; delt=0.02; bet=0.05; aa=0.5;

model(block, bytecode, cutoff=0);
c + k - aa*x*k(-1)^alph - (1-delt)*k(-1);
c^(-gam) - (1+bet)^(-1)*(aa*alph*x(+1)*k^(alph-1) + 1 - delt)*c(+1)^(-gam);
end;
```

Concluding remarks

- For large scale models, block decomposition strongly reduces the time required to compute the steady-state, the deterministic simulation and the Blanchard and Kahn's conditions.
- The reduction of the computational time seems to be modest when we consider the RE solution and the evaluation of the likelihood using a block Kalman Filter.
- The gains using block decomposition seem to be much more promising when global methods are used to compute the RE solution: one solution to curse of dimension problem.