Solving Deterministic Models

Shanghai Dynare Workshop

Sébastien Villemot

CEPREMAP

October 27, 2013

Introduction

- Deterministic = perfect foresight
- Perfect anticipation of all shocks in the future, and therefore of all future choice variables
- Can be solved exactly (up to rounding errors)
- Full nonlinearities taken into account
- Often useful when starting study of a model

Outline

Outline

The (deterministic) neoclassical growth model

$$\max_{\{c_t\}_{t=1}^{\infty}} \sum_{t=1}^{\infty} \beta^{t-1} \frac{c_t^{1-\sigma}}{1-\sigma}$$

s.t.

$$c_t + k_t = A_t k_{t-1}^{\alpha} + (1-\delta)k_{t-1}$$

First order conditions:

$$c_t^{-\sigma} = \beta c_{t+1}^{-\sigma} \left(\alpha A_{t+1} k_t^{\alpha-1} + 1 - \delta\right)$$
$$c_t + k_t = A_t k_{t-1}^{\alpha} + (1-\delta)k_{t-1}$$

Steady state:

$$\bar{k} = \left(\frac{1-\beta(1-\delta)}{\beta\alpha\bar{A}}\right)^{\frac{1}{\alpha-1}}$$
$$\bar{c} = \bar{A}\bar{k}^{\alpha} - \delta\bar{k}$$

Note the absence of stochastic elements! No expectancy term, no probability distribution

The general problem

Deterministic, perfect foresight, case:

$$f(y_{t+1}, y_t, y_{t-1}, u_t) = 0$$

y : vector of endogenous variables

u : vector of exogenous shocks

Identification rule: as many endogenous (y) as equations (f)

Steady state

- A steady state, $\bar{y}$, for the model satisfies

$$f(\bar{y}, \bar{y}, \bar{y}, \bar{u}) = 0$$

- Note that a steady state is conditional to:
  - The steady state values of exogenous variables $\bar{u}$
  - The value of parameters (implicit in the above definition)
- Even for a given set of exogenous and parameter values, some (nonlinear) models have several steady states
- The steady state is computed by Dynare with the `steady` command
- That command internally uses a nonlinear solver

What if more than one lead or one lag?

- A model with more than one lead or lag can be transformed in the form with one lead and one lag using auxiliary variables
- Transformation done automatically by Dynare
- For example, if there is a variable with two leads x_{t+2}:
  - create a new auxiliary variable a
  - replace all occurrences of x_{t+2} by a_{t+1}
  - add a new equation: $a_t = x_{t+1}$
- Symmetric process for variables with more than one lag

Return to the neoclassical growth model

$$y_t = \begin{pmatrix} c_t \\ k_t \end{pmatrix}$$

$$u_t = A_t$$

$$f(y_t) = \begin{pmatrix} c_t^{-\sigma} - \beta c_{t+1}^{-\sigma} \left(\alpha A_{t+1} k_t^{\alpha-1} + 1 - \delta \right) \\ c_t + k_t - A_t k_{t-1}^{\alpha} + (1-\delta) k_{t-1} \end{pmatrix}$$

Solution of deterministic models

- Approximation: impose return to equilibrium in finite time instead of asymptotically
- However possible to return to another point than the steady state
- Useful to study full implications of nonlinearities
- Computes the trajectory of the variables numerically
- Uses a Newton-type method on the stacked system

A two-boundary value problem

Approximation of an infinite horizon model by a finite horizon one

The stacked system for a simulation over T periods:

$$\begin{cases} f(y_2, y_1, y_0, u_1) = 0 \\ f(y_3, y_2, y_1, u_2) = 0 \\ \vdots \\ f(y_{T+1}, y_T, y_{T-1}, u_T) = 0 \end{cases}$$

for y_0 and $y_{T+1} = \bar{y}$ given.

Compact representation:

$$F(Y) = 0$$

where $Y = \begin{bmatrix} y_1' & y_2' & \dots & y_T' \end{bmatrix}'$.

Outline

A Newton approach

- Start from an initial guess $Y^{(0)}$
- Iterate. Updated solutions $Y^{(k+1)}$ are obtained by solving:

$$F(Y^{(k)}) + \left[\frac{\partial F}{\partial Y}\right]\left(Y^{(k+1)} - Y^{(k)}\right) = 0$$

- Terminal condition:

$$||Y^{(k+1)} - Y^{(k)}|| < \varepsilon_Y \text{ and/or } ||F(Y^{(k)})|| < \varepsilon_F$$

A practical difficulty

The size of the Jacobian is very large. For a simulation over T periods of a model with n endogenous variables, it is a matrix of order $n \times T$.
3 ways of dealing with it:

- 15 years ago, it was more of a problem than today: LBJ (the default method in Dynare ≤ 4.2) exploited the particular structure of this Jacobian using relaxation techniques
- Handle the Jacobian as one large, sparse, matrix (now the default method in Dynare ≥ 4.3)
- Block decomposition (divide-and-conquer methods) implemented by Mihoubi

Shape of the Jacobian

$$\frac{\partial F}{\partial Y} = \begin{pmatrix} B_1 & C_1 & & & & & \\ A_2 & B_2 & C_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & A_t & B_t & C_t & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & A_{T-1} & B_{T-1} & C_{T-1} \\ & & & & & A_T & B_T \end{pmatrix}$$

where

$$A_s = \frac{\partial f}{\partial y_{t-1}}(y_{s+1}, y_s, y_{s-1})$$

$$B_s = \frac{\partial f}{\partial y_t}(y_{s+1}, y_s, y_{s-1})$$

$$C_s = \frac{\partial f}{\partial y_{t+1}}(y_{s+1}, y_s, y_{s-1})$$

Relaxation (1/5)

The idea is to triangularize the stacked system:

$$\begin{pmatrix} B_1 & C_1 & & & & \\ A_2 & B_2 & C_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & A_{T-1} & B_{T-1} & C_{T-1} \\ & & & & A_T & B_T \end{pmatrix} \Delta Y = - \begin{pmatrix} f(y_2, y_1, y_0, u_1) \\ f(y_3, y_2, y_1, u_2) \\ \vdots \\ \vdots \\ f(y_T, y_{T-1}, y_T, u_{T-1}) \\ f(y_{T+1}, y_T, y_{T-1}, u_T) \end{pmatrix}$$

Relaxation (2/5)

First period is special:

$$\begin{pmatrix} I & D_1 & & & & \\ & B_2 - A_2 D_1 & C_2 & & & \\ & A_3 & B_3 & C_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & A_{T-1} & B_{T-1} & C_{T-1} \\ & & & & A_T & B_T \end{pmatrix} \Delta Y = - \begin{pmatrix} d_1 \\ f(y_3, y_2, y_1, u_2) + A_2 d_1 \\ f(y_4, y_3, y_2, u_3) \\ \vdots \\ f(y_T, y_{T-1}, y_T, u_{T-1}) \\ f(y_{T+1}, y_T, y_{T-1}, u_T) \end{pmatrix}$$

where

- $D_1 = B_1^{-1} C_1$
- $d_1 = B_1^{-1} f(y_2, y_1, y_0, u_1)$

Relaxation (3/5)

Normal iteration:

$$\begin{pmatrix} I & D_1 & & & & \\ & I & D_2 & & & \\ & & B_3 - A_3 D_2 & C_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & A_{T-1} & B_{T-1} & C_{T-1} \\ & & & & A_T & B_T \end{pmatrix} \Delta Y = - \begin{pmatrix} d_1 \\ d_2 \\ f(y_4, y_3, y_2, u_3) + A_3 d_2 \\ \vdots \\ f(y_T, y_{T-1}, y_T, u_{T-1}) \\ f(y_{T+1}, y_T, y_{T-1}, u_T) \end{pmatrix}$$

where

- $D_2 = (B_2 - A_2 D_1)^{-1} C_2$
- $d_2 = (B_2 - A_2 D_1)^{-1} (f(y_3, y_2, y_1, u_2) + A_2 d_1)$

Relaxation (4/5)

Final iteration:

$$\begin{pmatrix} I & D_1 & & & & \\ & I & D_2 & & & \\ & & I & D_3 & & \\ & & & \ddots & \ddots & \\ & & & & I & D_{T-1} \\ & & & & & I \end{pmatrix} \Delta Y = - \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{T-1} \\ d_T \end{pmatrix}$$

where

$$d_T = (B_T - A_T D_{T-1})^{-1}(f(y_{T+1}, y_T, y_{T-1}, u_T) + A_T d_{T-1})$$

Relaxation (5/5)

- The system is then solved by backward iteration:

$$y_T^{k+1} = y_T^k - d_T$$
$$y_{T-1}^{k+1} = y_{T-1}^k - d_{T-1} - D_{T-1}(y_T^{k+1} - y_T^k)$$
$$\vdots$$
$$y_1^{k+1} = y_1^k - d_1 - D_1(y_2^{k+1} - y_2^k)$$

- No need to ever store the whole Jacobian: only the D_s and d_s have to be stored
- Relaxation was the default method in Dynare $\leq$ 4.2, since it was memory efficient

Sparse matrix algebra

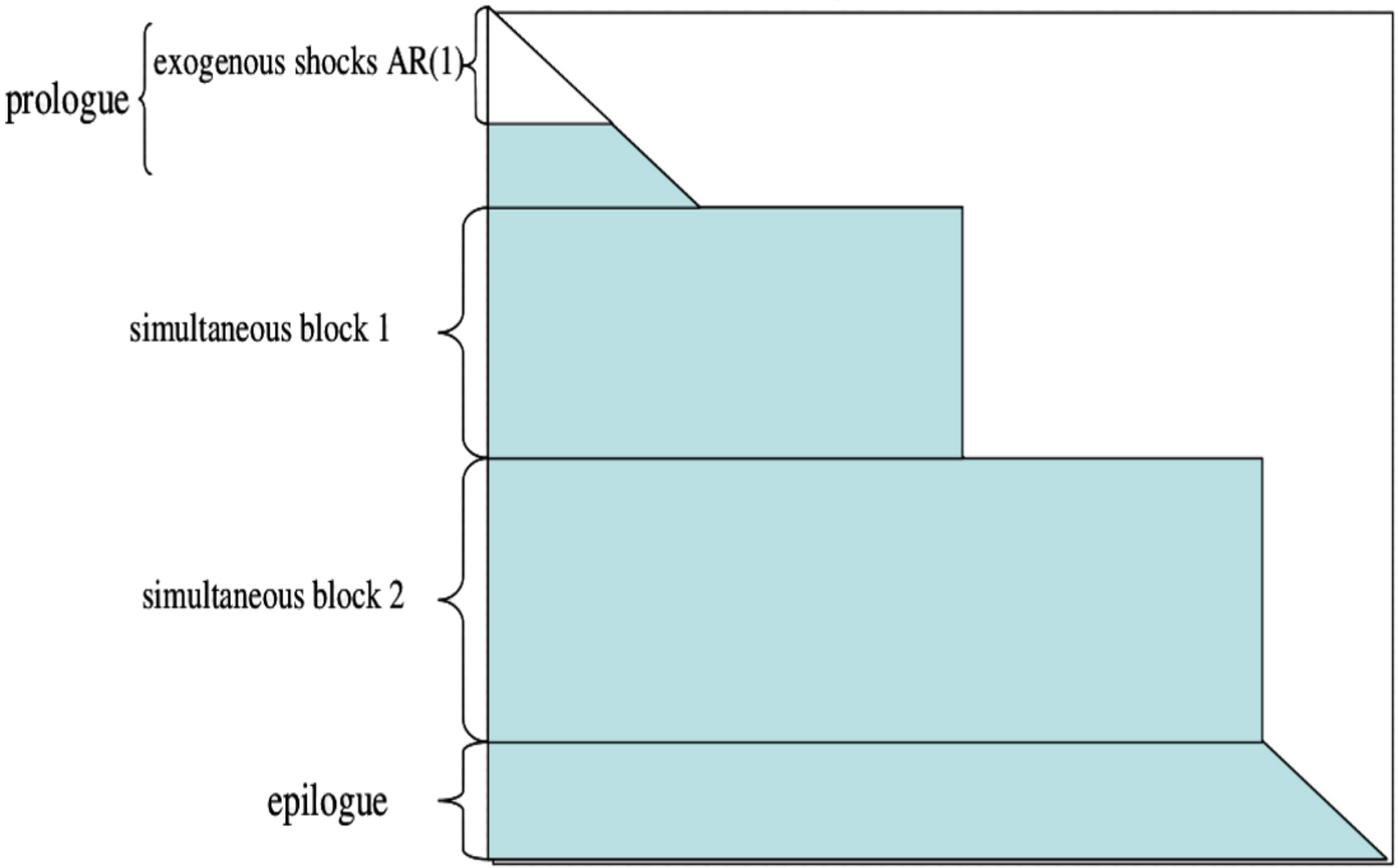
- A sparse matrix is a matrix where most entries are zero
- The Jacobian of the deterministic problem is a sparse matrix:
  - Lots of zero blocks
  - The A_s, B_s and C_s are themselves sparse
- More efficient storage possible than storing all entries
- Usually stored as a list of triplets (i, j, v) where (i, j) is a matrix coordinate and v a non-zero value
- Family of optimized algorithms for such matrices (including matrix inversion for our Newton algorithm)
- Available as native objects in MATLAB/Octave
- Works well for medium size deterministic models
- Nowadays more efficient than relaxation, even though it does not exploit the particular structure of the Jacobian $\Rightarrow$ default method in Dynare $\geq$ 4.3

Block decomposition (1/3)

- Idea: apply a divide-and-conquer technique to model simulation
- Principle: identify recursive and simultaneous blocks in the model structure
- First block (prologue): equations that only involve variables determined by previous equations; example: AR(1) processes
- Last block (epilogue): pure output/reporting equations
- In between: simultaneous blocks, that depend recursively on each other
- The identification of the blocks is performed through a matching between variables and equations (normalization), then a reordering of both

Block decomposition (2/3)

Form of the reordered Jacobian



Block decomposition (3/3)

- Can provide a significant speed-up on large models
- Implemented in Dynare by Ferhat Mihoubi
- Available as option `block` to the `model` command
- Bigger gains when used in conjunction with `bytecode` options

Outline

Example: neoclassical growth model with investment

The social planner problem is as follows:

$$\max_{\{c_{t+j},\ell_{t+j},k_{t+j}\}_{j=0}^{\infty}} \mathbb{E}_t \sum_{j=0}^{\infty} \beta^j u(c_{t+j}, \ell_{t+j})$$

s.t.

$$y_t = c_t + i_t$$
$$y_t = A_t f(k_{t-1}, \ell_t)$$
$$k_t = i_t + (1-\delta)k_{t-1}$$
$$A_t = A^\star e^{a_t}$$
$$a_t = \rho\, a_{t-1} + \varepsilon_t$$

where ε_t is an exogenous shock.

Specifications

- Utility function:

$$u(c_t, \ell_t) = \frac{\left[c_t^{\theta}(1-\ell_t)^{1-\theta}\right]^{1-\tau}}{1-\tau}$$

- Production function:

$$f(k_{t-1}, \ell_t) = \left[\alpha k_{t-1}^{\psi} + (1-\alpha)\ell_t^{\psi}\right]^{\frac{1}{\psi}}$$

First order conditions

- Euler equation:

$$u_c(c_t, \ell_t) = \beta \, \mathbb{E}_t \left[u_c(c_{t+1}, \ell_{t+1}) \Big(A_{t+1} f_k(k_t, \ell_{t+1}) + 1 - \delta \Big) \right]$$

- Arbitrage between consumption and leisure:

$$\frac{u_\ell(c_t, \ell_t)}{u_c(c_t, \ell_t)} + A_t f_l(k_{t-1}, \ell_t) = 0$$

- Resource constraint:

$$c_t + k_t = A_t f(k_{t-1}, \ell_t) + (1 - \delta) k_{t-1}$$

Calibration

Weight of consumption in utility	θ	0.357
Risk aversion	τ	2.0
Share of capital in production	α	0.45
Elasticity of substitution capital/labor (fct of...)	ψ	-0.1
Discount factor	β	0.99
Depreciation rate	δ	0.02
Autocorrelation of productivity	ρ	0.8
Steady state level of productivity	$A^\star$	1

Scenario 1: Return to equilibrium

Return to equilibrium starting from $k_0 = 0.5\bar{k}$.

Fragment from `rbc_det1.mod`

```
...
steady;

ik = varlist_indices('Capital',M_.endo_names);
CapitalSS = oo_.steady_state(ik);

histval;
Capital(0) = CapitalSS/2;
end;

simul(periods=300);
```

Scenario 2: A temporary shock to TFP

- The economy starts from the steady state
- There is an unexpected negative shock at the beginning of period 1: $\varepsilon_1 = -0.1$

Fragment from `rbc_det2.mod`

```
...
steady;

shocks;
var EfficiencyInnovation;
periods 1;
values -0.1;
end;

simul(periods=100);
```

Scenario 3: Pre-announced favorable shocks in the future

- The economy starts from the steady state
- There is a sequence of positive shocks to A_t: 4% in period 5 and an additional 1% during the 4 following periods

Fragment from `rbc_det3.mod`

```
...
steady;

shocks;
var EfficiencyInnovation;
periods 4, 5:8;
values 0.04, 0.01;
end;
```

Scenario 4: A permanent shock

- The economy starts from the initial steady state ($a_0 = 0$)
- In period 1, TFP increases by 5% permanently (and this was unexpected)

Fragment from rbc_det4.mod

```
...
initval;
EfficiencyInnovation = 0;
end;

steady;

endval;
EfficiencyInnovation = (1-rho)*log(1.05);
end;

steady;
```

Scenario 5: A pre-announced permanent shock

- The economy starts from the initial steady state ($a_0 = 0$)
- In period 6, TFP increases by 5% permanently
- A `shocks` block is used to maintain TFP at its initial level during periods 1–5

Fragment from `rbc_det5.mod`

```
...
// Same initval and endval blocks as in Scenario 4
...

shocks;
var EfficiencyInnovation;
periods 1:5;
values 0;
end;
```

Outline

Zero nominal interest rate lower bound

- Implemented by writing the law of motion under the following form in Dynare:

$$i_t = \max\left\{0, (1-\rho_i)i^* + \rho_i i_{t-1} + \rho_\pi(\pi_t - \pi^*) + \varepsilon_t^i\right\}$$

- *Warning:* this form will be accepted in a stochastic model, but the constraint will not be enforced in that case!

Irreversible investment

Same model than above, but the social planner is constrained to positive investment paths:

$$\max_{\{c_{t+j},\ell_{t+j},k_{t+j}\}_{j=0}^{\infty}} \sum_{j=0}^{\infty} \beta^j u(c_{t+j}, \ell_{t+j})$$

s.t.

$$y_t = c_t + i_t$$
$$y_t = A_t f(k_{t-1}, \ell_t)$$
$$k_t = i_t + (1-\delta)k_{t-1}$$
$$i_t \geq 0$$
$$A_t = A^\star e^{a_t}$$
$$a_t = \rho\, a_{t-1} + \varepsilon_t$$

where the technology (f) and the preferences (u) are as above.

First order conditions

$$u_c(c_t, \ell_t) - \mu_t = \beta \mathbb{E}_t \Big[u_c(c_{t+1}, \ell_{t+1}) \left(A_{t+1} f_k(k_t, \ell_{t+1}) + 1 - \delta \right) - \mu_{t+1}(1 - \delta) \Big]$$

$$\frac{u_\ell(c_t, \ell_t)}{u_c(c_t, \ell_t)} + A_t f_l(k_{t-1}, \ell_t) = 0$$

$$c_t + k_t = A_t f(k_{t-1}, \ell_t) + (1 - \delta) k_{t-1}$$

$$\mu_t \left(k_t - (1 - \delta) k_{t-1} \right) = 0$$

where $\mu_t \geq 0$ is the Lagrange multiplier associated to the non-negativity constraint for investment.

Writing this model in Dynare

Fragment from rbcii.mod

```
mu = max(0,(((c^theta)*((1-l)^(1-theta)))^(1-tau))/c
           - expterm(1)+beta*mu(1)*(1-delta));

(i<=0)*(k - (1-delta)*k(-1))
 + (i>0)*((((c^theta)*((1-l)^(1-theta)))^(1-tau))/c
         - expterm(1)+beta*mu(1)*(1-delta)) = 0;

expterm = beta*((((c^theta)*((1-l)^(1-theta)))^(1-tau))/c)
          *(alpha*((y/k(-1))^(1-psi))+1-delta);
```

Outline

Extended path (EP) algorithm

- Algorithm for creating a stochastic simulated series
- At every period, compute endogenous variables by running a *deterministic* simulation with:
  - the previous period as initial condition
  - the steady state as terminal condition
  - a random shock drawn for the current period
  - but no shock in the future
- Advantages:
  - shocks are unexpected *at every period*
  - nonlinearities fully taken into account
- Inconvenient: solution under certainty equivalence (Jensen inequality is violated)
- Method introduced by Fair and Taylor (1983)
- Implemented in Dynare 4.3 by Stéphane Adjemian under the command `extended_path`

k-step ahead EP

- Accuracy can be improved by computing conditional expectation by quadrature, computing next period endogenous variables with the previous algorithm
- Approximation: at date t, agents assume that there will be no more shocks after period $t + k$ (hence k measures the degree of future uncertainty taken into account)
- If $k = 1$: one-step ahead EP; no more certainty equivalence
- By recurrence, one can compute a k-step ahead EP: even more uncertainty taken into account
- Difficulty: computing complexity grows exponentially with k
- k-step ahead EP currently implemented in (forthcoming) Dynare 4.4; triggered with option `order = k` of `extended_path` command