Perfect foresight models

Stéphane Adjemian

stepan@adjemian.eu

June, 2021

Introduction

- We abstract from the effects of future uncertainty by assuming that the agents can form perfect foresights.
- For instance, if the current productivity departs from its equilibrium level, all the agents can perfectly anticipate the productivity's return path to its steady state.
- Possible future shocks are anticipated (deterministic).
- **Pros:**
  - Nonlinear deterministic models are much easier to solve.
  - Can handle models with kinks or OBC.
- **Cons:**
  - What if future uncertainty really matters?
  - Can we simulate time series?... (yes)

Outline

Deterministic RBC model (equations)

As an example, consider a deterministic version of the RBC model, where the dynamics of consumption, physical capital and productivity are given by:

$$\frac{c_{t+1}}{c_t} = \beta\left(\alpha e^{a_{t+1}} k_{t+1}^{\alpha-1} + 1 - \delta\right) \qquad \text{(Euler)}$$

$$k_{t+1} = e^{a_t} k_t^{\alpha} + (1-\delta)k_t - c_t \qquad \text{(Transition)}$$

$$a_t = \rho a_{t-1} \qquad \text{(Productivity)}$$

- No zero mean innovations in the exogenous productivity process.
- $\Rightarrow$ No conditional expectation in the Euler equation.
- Later we will consider the possibility of perfectly anticipated shocks.

The equations given in the previous slide are the first order condition of a Central planner, whose behavior is characterized by:

$$\begin{aligned} & \max_{\{c_{t+j},k_{t+1+j}\}_{j=0}^{\infty}} \sum_{j=0}^{\infty} \beta^j \log c_{t+j} \\ s.t. \quad & c_{t+j} + k_{t+1+j} = e^{a_{t+j}} k_{t+j}^{\alpha} + (1-\delta)k_{t+j} \quad \forall j \geq 0 \\ & a_{t+j} = \rho a_{t+j-1} \end{aligned} \tag{1}$$

- $\beta \in (0,1)$ is the discount factor,
- $\alpha \in (0,1)$ is the elasticity of production with respect to physical capital,
- $\delta \in (0,1)$ is the depreciation rate of physical capital stock,
- Autoregressive parameter ρ is assumed to be less than one in absolute value,
- Usual notations: c_t, k_t and a_t respectively stand for consumption, physical capital stock and (logged) exogenous efficiency.

For simplicity, we state the central planner formulation of the problem. We know that, without any departure from the perfect competition assumptions, we would obtain exactly the same equilibrium dynamics by stating the decentralized problem. The Lagrangian associated to this problem is given by:

$$\mathscr{L} = \sum_{j=0}^{\infty} \beta^j \left(\log c_{t+j} + \lambda_{t+j} \left\{ e^{a_{t+j}} k_{t+j}^{\alpha} + (1-\delta)k_{t+j} - c_{t+j} - k_{t+1+j} \right\} \right)$$

where $\lambda_{t+j} \geq 0$ is the Lagrange multiplier for the resource constraint.

The same Lagrangian can be alternatively written in the following manner:

$$\begin{aligned} \mathscr{L} = \log c_t + \lambda_t \left\{ e^{a_t} k_t^{\alpha} + (1-\delta)k_t - c_t - k_{t+1} \right\} \\ \beta \left(\log c_{t+1} + \lambda_{t+1} \left\{ e^{a_{t+1}} k_{t+1}^{\alpha} + (1-\delta)k_{t+1} - c_{t+1} - k_{t+2} \right\} \right) \\ + \ldots \\ \beta^s \left(\log c_s + \lambda_s \left\{ e^{a_s} k_s^{\alpha} + (1-\delta)k_s - c_s - k_{s+1} \right\} \right) \\ + \ldots \end{aligned}$$

The first order conditions are obtained by setting to zero the first partial derivatives of $\mathscr{L}$ with respect to c_t and k_{t+1}. We obtain:

$$\lambda_t = \frac{1}{c_t} \tag{a}$$

and

$$\lambda_t = \beta \lambda_{t+1} \left(\alpha e^{a_{t+1}} k_{t+1}^{\alpha - 1} + 1 - \delta \right) \tag{b}$$

The Lagrange multiplier, λ_t, is also called the shadow price of capital. λ_t says how much the household is willing to pay for an additional unit of physical capital tomorrow. Condition (a) relates this implicit price to the marginal utility of consumption (an additional unit of physical capital tomorrow is at the cost of one unit of consumption today).

Equation (b) states that, in equilibrium, what is lost by postponing consumption today (λ_t) has to be exactly compensated by the discounted gains obtained tomorrow (β times λ_{t+1} times the future real gross return to physical capital).

Substituting (a) into (b) we obtain the Euler equation which, completed with the resource constraint and the law of motion for productivity, characterizes the dynamics of the economy.

Substituting the transition equation in the Euler equation, one can see that the first order conditions implicitly define a second order nonlinear recurrent equation for the physical capital stock (with k_t, k_{t+1} and k_{t+2}). This recurrent equation admits an infinity of solution given the initial condition k_0. We need to add another boundary condition to pin down a unique solution. In an infinite horizon problem we use a transversality condition for that purpose:

$$\lim_{T \to \infty} \beta^T \lambda_T k_{T+1} = 0$$

where λ_T is the previously defined Lagrange multiplier (the marginal utility of consumption). This condition states that asymptotically the detention of capital is not valued.

In the sequel, when solving perfect foresight models, we will not explicitely referring to a transversality condition. To pin down a unique path for the endogenous variables we will impose that these variable have to converge to a steady state. Provided that the steady state has good properties (determinacy), this additional boundary condition is enough to identify a unique solution. This terminal condition, the steady state, has to satisfy the transversality condition. A careful analysis of this model

Deterministic RBC model (steady state)

The steady state of the model is given by:

$$a^\star = 0$$

$$k^\star = \left(\frac{\alpha}{b + \delta} \right)^{\frac{1}{1-\alpha}}$$

$$c^\star = k^{\star\,\alpha} - \delta k^\star$$

- The steady state is unstable in all directions except one, for all admissible values of the deep parameters (saddle path property).
- We can illustrate this property, that will be shared by all the DSGE models that we will deal with in this course, with a phase diagram.
- We assume that $a_0 = a^\star$, so that we get rid of the productivity variable.

The steady state level of productivity is necessarily zero because it is the only possible value for $a^\star$ such that $a^\star = \rho a^\star$ for any value of the autoregressive parameter ρ.

Evaluating the Euler equation at the (unknown) steady state, we obtain:

$$\beta^{-1} = \alpha \ k^{\star \ \alpha - 1} + 1 - \delta$$

because the constant levels of consumption cancel out. Defining b a positive real number such that $\beta = \frac{1}{1+b}$, we have equivalently:

$$\frac{b + \delta}{\alpha} = k^{\star \ \alpha - 1}$$

or

$$k^\star = \left(\frac{\alpha}{b + \delta} \right)^{\frac{1}{1-\alpha}}$$

Substituting $k^\star$ in the transition equation, we obtain the steady state level of consumption:

$$c^\star = k^{\star \ \alpha} - \delta k^\star$$

Deterministic RBC model (phase diagram, 1/3)

- Describe how k in c move in each region of the plan (k, c).

- The physical capital stock is expected to decrease ($\Delta k_{t+1} \leq 0$) if and only if (k_t, c_t) is such that:

$$c_t \geq k_t^{\alpha} - \delta k_t \equiv \varphi_{\Delta k}(k_t)$$

  where $\varphi_{\Delta k}$ is an increasing function of k as long as the marginal productivity of capital is greater than the depreciation rate ($k < \bar{k}$). For higher levels k, $\varphi_{\Delta k}$ is a decreasing function of k.

- The consumption is expected to decrease ($\Delta c_{t+1} \leq 0$) if and only if (k_t, c_t) is such that:

$$c_t \leq k_t^{\alpha} + (1 - \delta)k_t - k^{\star} \equiv \varphi_{\Delta c}(k_t)$$

  where $\varphi_{\Delta c}$ is an increasing function of k.

- The transition equation can be equivalently rewritten as:

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

$\Delta k_{t+1} \leq 0$ is equivalent to:

$$k_t^{\alpha} - \delta k_t - c_t \leq 0$$

$$\Leftrightarrow c_t \geq k_t^{\alpha} - \delta k_t \qquad (\varphi_{\Delta k})$$

One can show that $\varphi'_{\Delta k}(k) \geq 0$ if and only if $k \leq \bar{k} \equiv (\alpha/\delta)^{\frac{1}{1-\alpha}}$ and that $\bar{k} > k^{\star}$.

- If $\Delta c_{t+1} \leq 0$, then $c_{t+1}/c_t \leq 1$ and from the Euler equation we have:

$$\alpha k_{t+1}^{\alpha-1} + 1 - \delta \leq \beta^{-1}$$

$$k_{t+1} \leq k^{\star}$$

Substituting the law of motion for the physical capital stock:

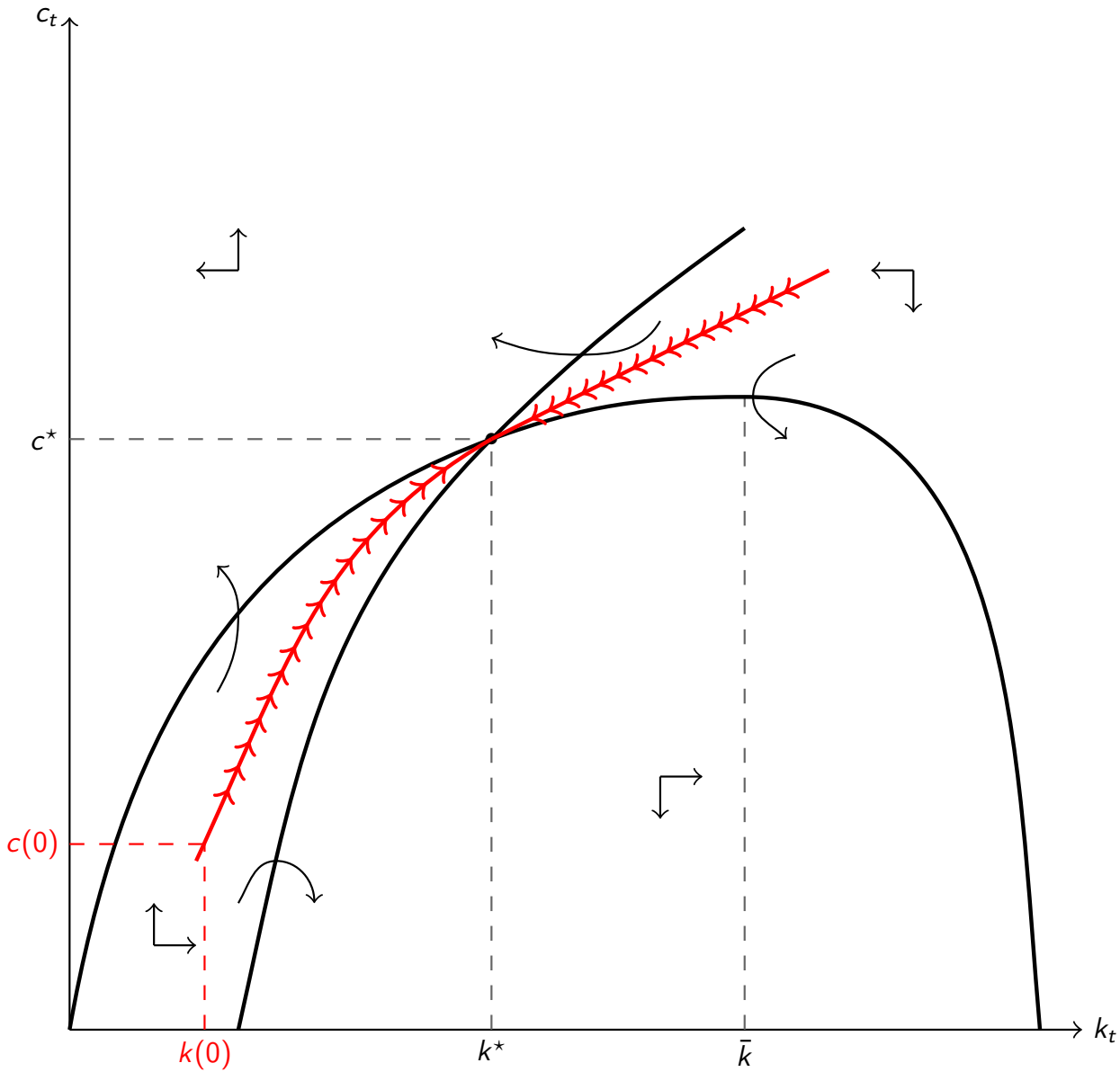
$$k_t^{\alpha} + (1 - \delta)k_t - c_t \leq k^{\star}$$

$$\Leftrightarrow c_t \geq k_t^{\alpha} + (1 - \delta)k_t - k^{\star} \qquad (\varphi_{\Delta c})$$

$\varphi'_{\Delta c}(k) \geq 0$ for any level of physical capital.

- The representative curves for $\varphi_{\Delta c}$ and $\varphi_{\Delta k}$ divide the plan (k, c) if four regions. In each region, the vertical and horizontal arrows indicate how consumption and physical capital stock are expected to move. Curved arrows on the frontiers show how a path can go from a region to another. Clearly it is not possible to go from the North-West region to the South-West region, since in the North-West region $\Delta c_{t+1} > 0$ and $\Delta k_{t+1} < 0$. On the contrary a path can go from the South-West region to the North-West region, the frontier has to be crossed vertically since along $\varphi_{\Delta k}(k)$ the physical capital stock is not expected to move.

Deterministic RBC model (phase diagram, 2/3)



Deterministic RBC model (phase diagram, 3/3)

- Any trajectory in the (k, c) plan satisfies the Euler and transition equations... But, given an initial condition for the physical capital stock, only one path leads to the steady state in the long run.

- If $k(0)$ is the initial state of the economy, the central planner chooses the initial level of the control variable, $c(0)$, such that the economy will converge to the steady state in the long run $\Rightarrow$ $c(0)$ has to be on the saddle path (the red curve)...

- Otherwise the economy will move away permanently from its steady state, and the economy will eventually disappear in finite time or violate the transversality condition.

Deterministic RBC model (solution strategies)

- To study the properties and implications of the model, we need to solve it.
  - A first approach is to solve for policy rules, *ie* time invariant functional forms that relate the control variables to the state variables.
  - A second approach is to solve for the paths of all the endogenous variables.
- The second approach, which will be emphasized in this course, is less general because the solution is obtained for a specific initial condition and/or set of shocks.
- But this approach
  - offers a better control over the accuracy of the solution
  - is often easier to compute (especially in the presence of non differentiabilities induced by OBC)
  - suffers to a less extent from the curse of dimensionality

Deterministic RBC model (solving for the policy rules)

- We are looking for a time invariant function $c_t = \psi(k_t)$ satisfying the Euler and transition equations:

$$\frac{\psi\left(k_t^{\alpha} + (1-\delta)k_t - \psi(k_t)\right)}{\psi\left(k_t\right)} =$$

$$\beta\left[\alpha\left(k_t^{\alpha} + (1-\delta)k_t - \psi(k_t)\right)^{\alpha-1} + (1-\delta)k_t - \psi(k_t)\right]$$

- In general there is no closed form solution to this functional equation...

- We postulate a parametric solution $\widehat{\psi}(k_t, \mathbf{a})$, and look for the vector of reduced form parameters $\mathbf{a}$ such that the residuals of the previous equation are satisfied or arbitrarily small for some values of k_t.

Deterministic RBC model (solving for the paths, 1/2)

- We assume that the economy reaches the steady state in finite time ($T < \infty$). This is obviously an approximation.
- Paths for c and k must satisfy the following system (Euler and transition equations for $t = 0, \ldots, T$):

$$
\begin{aligned}
\frac{c_1}{c_0} - \beta\left(\alpha k_1^{\alpha-1} + 1 - \delta\right) &= 0 \\
k_1 - k_0^{\alpha} - (1-\delta)k_0 + c_0 &= 0 \\
\frac{c_2}{c_1} - \beta\left(\alpha k_2^{\alpha-1} + 1 - \delta\right) &= 0 \\
k_2 - k_1^{\alpha} - (1-\delta)k_1 + c_1 &= 0 \\
&\vdots \\
\frac{c_{t+1}}{c_t} - \beta\left(\alpha k_{t+1}^{\alpha-1} + 1 - \delta\right) &= 0 \\
k_{t+1} - k_t^{\alpha} - (1-\delta)k_t + c_t &= 0 \\
&\vdots \\
\frac{c_T}{c_{T-1}} - \beta\left(\alpha k_T^{\alpha-1} + 1 - \delta\right) &= 0 \\
k_T - k_{T-1}^{\alpha} - (1-\delta)k_{T-1} + c_{T-1} &= 0
\end{aligned}
$$

with k_0 give and $c_T = c^\star$ the known boundary conditions.

Deterministic RBC model (solving for the paths, 2/2)

- In the numerical analysis literature this kind of problem is known as a two boundary value problem. The boundaries are the initial condition for the states and the terminal condition (steady state) for the control variables.

- At least two class of methods are available to solve these problems: the *shooting method* (find the initial condition such that the endogenous variables converge to the steady state) and the *relaxation method* that will be explored in this course.

- The relaxation method is generally found to be faster and more accurate (because, as shown below, we exploit all the equations at all the periods between 0 and $T-1$).

- Dynare solves the perfect foresight models using the relaxation approach.

How to solve a nonlinear equation? (1/4)

- We are looking for a real value $x^\star$ of x such that $f(x) = 0$, where f is a real nonlinear function in $\mathcal{C}_1$.

- This problem may have more than one solution, or no solution at all!

- Assuming we know such a solution $x^\star$, we can approximate the function f around $x^\star$:

$$f(x) = \underbrace{f(x^\star) + (x - x^\star)f'(x^\star)}_{h(x)} + \mathcal{O}\left((x - x^\star)^2\right)$$

- $h(x)$ is the tangent of f in $(x^\star, 0)$.

- Note that $x^\star$ is the unique solution of $h(x) = 0$, since $f(x^\star) = 0$.

- $h(x) = 0$ is much simpler to solve, it is trivial to compute the intersection of an affine function and the x-axis.

- But since we do not know $x^\star$ in practice, and *a fortiori* the tangent $h(x)$, this does not help a lot...

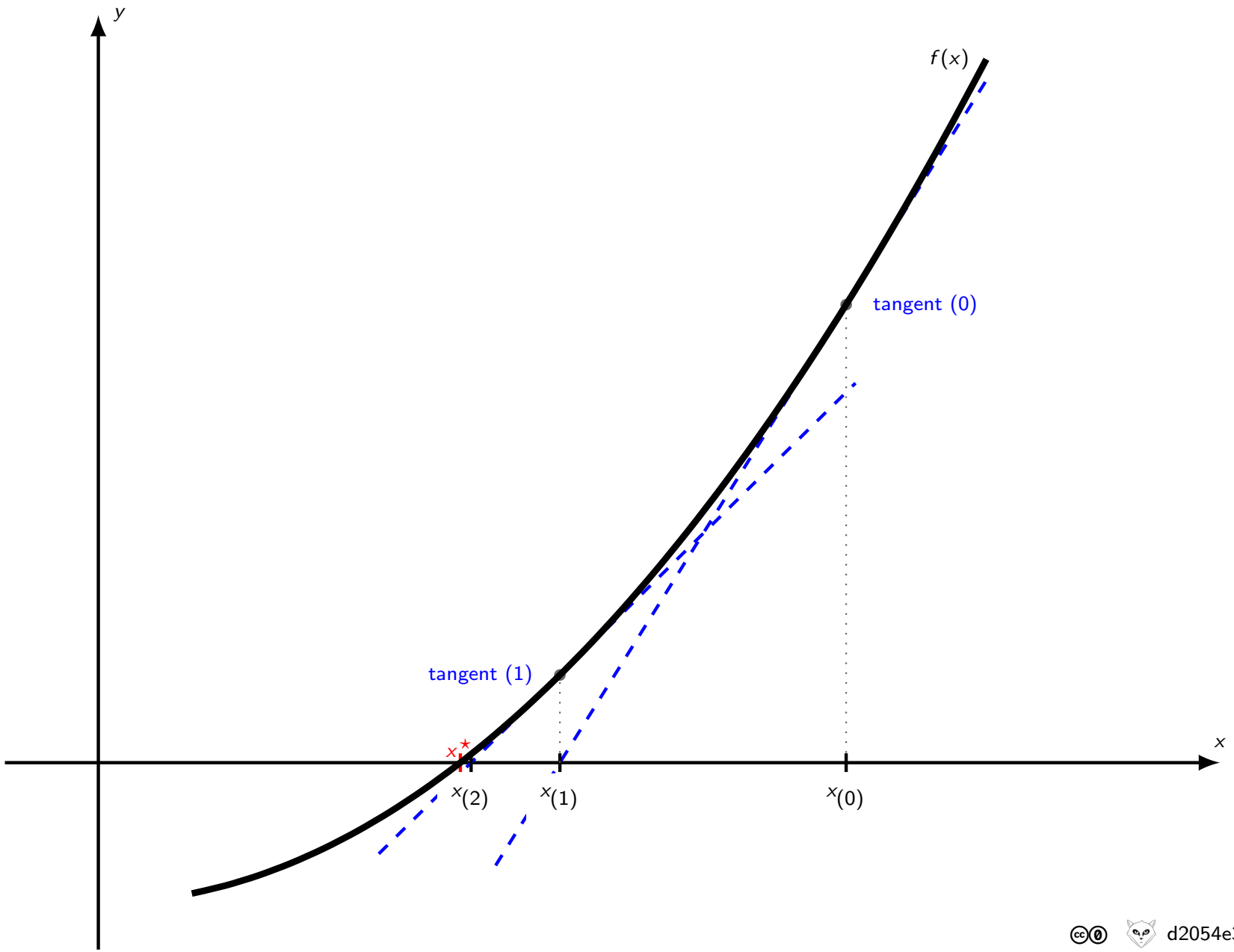
How to solve a nonlinear equation? (2/4)

Newton-Raphson algorithm

- Solve $f(x) = 0$ by solving a sequence of linear approximations of f.
- Suppose we have an initial guess $x_{(k)}$ such that $f(x_{(k)}) \neq 0$, and solve:
$$f(x_{(k)}) + (x - x_{(k)})f'(x_{(k)}) = 0$$
- The solution is:
$$\tilde{x} = x_{(k)} - \frac{f(x_{(k)})}{f'(x_{(k)})}$$
- Set $x_{(k+1)} = \tilde{x}$.
- Iterate over $k = 0, 1, 2, \ldots$.
- If $|x_{(k)} - x_{(k+1)}|$ or $|f(\tilde{x})|$ small enough, stop.
- If $x_{(0)}$ is *not too far* from $x^\star$ the Newton-Raphson algorithm will converge quadratically

How to solve a nonlinear equation? (3/4)

Newton-Raphson algorithm



How to solve a nonlinear equation? (4/4)

Pitfalls

- The Newton-Raphson may fail because:
  - The equation does not admit a solution,
  - The initial guess is too far from a solution.
- Except if we can show that a solution must exist, it is not possible to discriminate between these two sources of failure.
- See highly-instructive-examples-for-the-newton-raphson-method.
- If the equation admits more than one solution, the Newton-Raphson returns only one solution (depending on the initial guess).
- The derivative of the function f must exist.
- An accurate evaluation of the derivative is required.
- Preferrably the derivative has to be non nil everywhere.

How to solve a system of nonlinear equations?

- We are looking for a real $n \times 1$ vector $\mathbf{X}^\star$ such that $F(\mathbf{X}) = 0$, where $F : \mathbb{R}^n \to \mathbb{R}^n$ is a class $\mathcal{C}_1$ function.

- Same approach but the derivative $f'(x)$ is replaced by the Jacobian matrix:

$$J_F(\mathbf{X}) = \begin{pmatrix} \frac{\partial F_1(\mathbf{X})}{\partial X_1} & \frac{\partial F_1(\mathbf{X})}{\partial X_2} & \cdots & \frac{\partial F_1(\mathbf{X})}{\partial X_n} \\ \frac{\partial F_2(\mathbf{X})}{\partial X_1} & \frac{\partial F_2(\mathbf{X})}{\partial X_2} & \cdots & \frac{\partial F_2(\mathbf{X})}{\partial X_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial F_n(\mathbf{X})}{\partial X_1} & \frac{\partial F_n(\mathbf{X})}{\partial X_2} & \cdots & \frac{\partial F_n(\mathbf{X})}{\partial X_n} \end{pmatrix}$$

- The solution, $\mathbf{X}^{(k+1)}$ $(k = 0, 1, \dots)$ is updated, by solving the following linear system of equations:

$$F\left(\mathbf{X}^{(k)}\right) + J_F\left(\mathbf{X}^{(k)}\right)\left(\mathbf{X}^{(k+1)} - \mathbf{X}^{(k)}\right) = 0$$

- Stop the iterations as soon as

$$||F\left(\mathbf{X}^{(k)}\right)|| < \epsilon$$

Back to the deterministic RBC model (1/3)

- We have a system of $2T$ equations for $2T$ unknowns: $c_0, c_1, \ldots, c_{T-1}$ and $k_1, k_2, \ldots, k_T$, the paths for the endogenous variables.

- k_0 and c_0 are given by the initial and terminal conditions.

- We stack the unknown variables in a column vector $\mathbf{Y}$ as follows:

$$\mathbf{Y} = (c_0, k_1, c_1, k_2, c_2, \ldots, k_{t-1}, c_{t-1}, k_t)$$

  The boundary conditions (k_0 and c_T) are not in the vector.

- The previous system of equations can be represented by defining a function $F : \mathbb{R}_+^{2T} \to R^{2T}$ such that:

$$F(\mathbf{Y}) = 0$$

  This function is parameterized by the deep parameters (α, β and δ) and the boundary conditions (k_0 and c_T).

Back to the deterministic RBC model (2/3)

- At time t the Euler and transition equations depend only on c_t, c_{t+1}, k_t and k_{t+1}.

$\Rightarrow$ The jacobian of F, denoted $J_F(\mathbf{Y})$, is a sparse matrix.

- Specialized inversion algorithms are available for sparse matrices.
- Using Newton iterations, we can solve for the paths:

$$\mathbf{Y}^{(n)} = \mathbf{Y}^{(n-1)} - J_F\left(\mathbf{Y}^{(n-1)}\right)^{-1} F\left(\mathbf{Y}^{(n-1)}\right)$$

- As an initial guess for the unknown paths, $\mathbf{Y}^0$, we consider that all the variables at all periods are at the steady state.
- Matlab code for simulating this model are available here.

We adopt the following notations for the residuals of the Euler equation and transition equation at time t:

$$f^t = \frac{c_{t+1}}{c_t} - \beta\left(\alpha k_{t+1}^{\alpha-1} + 1 - \delta\right)$$
$$g^t = k_{t+1} - k_t^{\alpha} - (1-\delta)k_t + c_t$$

The partial derivatives that will appear in the Jacobian matrix are:

$$f_c^t = -\frac{c_{t+1}}{c_t^2}, \qquad f_{c_+}^t = \frac{1}{c_t}, \qquad f_k^t = 0, \qquad f_{k_+}^t = \beta\alpha(1-\alpha)k_{t+1}^{\alpha-2}$$

$$g_c^t = 1, \qquad g_{c_+}^t = 0, \qquad g_k^t = -(1-\delta) - \alpha k_t^{\alpha-1}, \qquad g_{k_+}^t = 1$$

The functions `eulerequation` and `transitionequation` available here return f^t and g^t with all the partial derivatives. Using these notations, the jacobian of the stacked model F is given by:

$$J_F(\mathbf{Y}) = \begin{pmatrix}
f_c^0 & f_{k_+}^0 & f_{c_+}^0 & 0 & \dots & \dots & \dots & 0 \\
g_c^0 & g_{k_+}^0 & g_{c_+}^0 & 0 & \dots & \dots & \dots & 0 \\
0 & f_k^1 & f_c^1 & f_{k_+}^1 & f_{c_+}^1 & 0 & \dots & 0 \\
0 & g_k^1 & g_c^1 & g_{k_+}^1 & g_{c_+}^1 & 0 & \dots & 0 \\
 & & \ddots & \ddots & \ddots & \ddots & & \\
0 & \dots & 0 & f_k^{T-2} & f_c^{T-2} & f_{k_+}^{T-2} & f_{c_+}^{T-2} & 0 \\
0 & \dots & 0 & g_k^{T-2} & g_c^{T-2} & g_{k_+}^{T-2} & g_{c_+}^{T-2} & 0 \\
0 & \dots & \dots & \dots & 0 & f_k^{T-1} & f_c^{T-1} & f_{k_+}^{T-1} \\
0 & \dots & \dots & \dots & 0 & g_k^{T-1} & g_c^{T-1} & g_{k_+}^{T-1}
\end{pmatrix}$$

This matrix has $4T^2$ elements and only $10 + (T-2) \times 6$ of them are nonzero. Note that the total number of elements grows quadratically while the number of nonzero elements grows linearly. Consequently the percentage of nonzero elements, *ie* the sparsity of the jacobian matrix, is a decreasing function of T. For instance for $T = 200$, the percentage of nonzero elements is 0.74875%. The function `modelevaluation` returns the residuals and sparse jacobian of the stacked model.

The function `modelsimulation` computes a transition to the steady state using the nonlinear solver described above.

Exercise 1. Use the provided matlab code to represent graphically the saddle path for $k \in k^\star \pm k^\star \times 40\%$.

Exercise 2. Adapt the provided matlab by adding the dynamic of efficiency (*ie* we do not assume $a_0 = a^\star$). Simulate the model with $k_0 = .6k^\star$ and $a_0 = -.1$ and plot the dynamic of output.

Exercise 3. We considered $c_T = c^\star$ as a terminal condition. An alternative would be to impose that the variation of consumption is zero in the two last periods. Adapt the provided matlab code to implement this terminal condition, and compare the simulation obtained with the two terminal conditions.

Back to the deterministic RBC model (3/3)

- The Newton algorithm may fail to converge (typically if $k^\star - k_0$ is too large).
- For instance, replace `.5kstar` by `.05kstar` in the last line of example1.m.
- The matlab's script example2.m shows how we can overcome this issue using an homotopy approach:
  - Suppose we want to solve $f(x) = 0$ for x but that the solver fails.
  - Define $F_\lambda(x) = \lambda f(x) + (1 - \lambda)x$, with $\lambda \in [0, 1]$.
  - For $\lambda = 0$ the solution of $F_\lambda(x) = 0$ is obvious.
  - We can use the solution $x^\star_\lambda$ as an initial guess for $F_{\lambda+\epsilon}(x) = 0$.
  - Iterating with an increasing sequence of λs we can reach the solution we are looking for (noting that $F_\lambda(x) \to f(x)$ as $\lambda \to 1$).

The general problem

We consider perfect foresight (PF) models that can be cast in the following form:

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

for $t = 1, \ldots, T$, with initial condition y_0 given for the states and $y_T = y^\star$ (the steady state) for the control variables.

- y is an $n \times 1$ vector of endogenous variables.
- u is a $q \times 1$ vector of perfectly anticipated innovations.
- $f : \mathbb{R}^{3n+q} \to \mathbb{R}^n$ must be a continuous function.
- The steady state $y^\star$ is such that $f(y^\star, y^\star, y^\star, u^\star) = 0$.

The shocks (u) can occur at any period. A shock in period 1 is a surprise, while a shock in period $t > 1$ is perfectly anticipated in period 1.

The general problem

Certainty equivalence

- A rational expectation (RE) version of the same model would be:

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

  where innovations u_t is a zero mean random variable. The future realizations of u are unknown at time t, only the associated CDF is in the time t information set.

- The RE and PF models would be equivalent if it was possible to pass the conditional expectation inside the f function.

- This is possible if and only if f is linear.

$\Rightarrow$ **Certainty equivalence property**: the size of the innovations does not affect agents behavior.

- For a linear model the IRFs obtained from a RE model solver are identical to the ones obtained with a PF model solver.

The general problem

Remarks

- The steady state, $y^\star$ depends implicitly on the deep parameters **and** the steady state level of the innovations (usually zero). If the model is well behaved the steady state exists, but nothing guaranties its unicity.

- We did not impose the differentiability of f. The model may admit kinks.

- Models with more than one lead and/or lag can be considered by adding auxiliary variables.

  - If a variable with two leads, x_{t+2}, is needed:
    + Create an auxiliary variable $a_t = x_{t+1}$.
    + Replace all occurrences of x_{t+2} by a_{t+1}.

  - If a variable with three leads, x_{t+3}, is needed:
    + Create auxiliary variables $a_t = x_{t+1}$ and $b_t = a_{t+1}$.
    + Replace all occurrences of x_{t+2} by b_{t+1}.

  - Same trick for variables with more than one lag

Solution of Perfect Foresight models

Stacked system

- **Approximation:** Impose return to equilibrium in finite time.

- We need to solve the stacked system of nonlinear equations:

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

  where the boundary conditions, y_0 and $y_{T+1} = y^\star$ are given.

- This system can be written $F(\mathbf{Y}) = 0$ with $\mathbf{Y} = (y_1', y_2', \ldots, y_T')'$, where the function F is parameterized by the deep parameters, y_0 and $y^\star$.

Solution of Perfect Foresight models

Newton algorithm

- Set an initial guess $\mathbf{Y}^{(0)}$. Usually the steady state: $y^\star \otimes \vec{e}_T$.
- Update the solution paths, by solving:

$$F\left(\mathbf{Y}^{(i)}\right) + J_F\left(\mathbf{Y}^{(i)}\right)\left(\mathbf{Y}^{(i+1)} - \mathbf{Y}^{(i)}\right) = 0$$

  for $\mathbf{Y}^{(i+1)}$ where $J_F\left(\mathbf{Y}\right) = \frac{\partial F(\mathbf{Y})}{\partial \mathbf{Y}'}$ is the jacobian matrix of F.
- Stop the iterations if

$$||F\left(\mathbf{Y}^{(i)}\right)|| < \epsilon$$

- The Newton iteration step exposed here is very basic. It may be useful to consider variable length steps (when $\mathbf{Y}$ moves in the direction defined by the jacobian matrix).
- Different methods are available to solve the systems of linear equations (we do not need to explicitely inverse the jacobian matrix).

Solution of Perfect Foresight models

Solving the system of linear equations

- The size of the jacobian is very large. If we have a model with $n = 100$ endogenous variables and $T = 400$, we must solve systems of 40000 linear equations!

- This jacobian matrix is sparse:

$$J_F(\mathbf{Y}) = \begin{pmatrix} f_y^1 & f_{y_+}^1 & 0 & \dots & \dots & \dots & \dots & 0 \\ f_{y_-}^2 & f_y^2 & f_{y_+}^2 & 0 & \dots & \dots & \dots & 0 \\ 0 & f_{y_-}^3 & f_y^3 & f_{y_+}^2 & 0 & \dots & \dots & 0 \\ & & \ddots & \ddots & \ddots & \ddots & & \\ 0 & \dots & \dots & 0 & f_{y_-}^{T-2} & f_y^{T-2} & f_{y_+}^{T-2} & 0 \\ 0 & \dots & \dots & \dots & 0 & f_{y_-}^{T-1} & f_y^{T-1} & f_{y_+}^{T-1} \\ 0 & \dots & \dots & \dots & \dots & 0 & f_{y_-}^{T} & f_y^{T} \end{pmatrix}$$

with $f_x^t = \frac{\partial F(\mathbf{Y}_t)}{\partial x'}$ for x equal to $y = y_t$, $y_- = y_{t-1}$, $y_+ = y_{t+1}$

- We have to exploit the sparsity when solving the systems of linear equations.

- Laffargue, Boucekine and Juillard (LBJ) propose to solve each Newton step:

$$\begin{pmatrix} f_y^1 & f_{y_+}^1 & & & & & \\ f_{y_-}^2 & f_y^2 & f_{y_+}^2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & f_{y_-}^{T-1} & f_y^{T-1} & f_{y_+}^{T-1} & \\ & & & & f_{y_-}^T & f_y^T \end{pmatrix} \Delta Y = - \begin{pmatrix} f(y_2, y_1, y_0, u_1) \\ f(y_3, y_2, y_1, u_2) \\ \vdots \\ \vdots \\ f(y_T, y_{T-1}, y_T, u_{T-1}) \\ f(y_{T+1}, y_T, y_{T-1}, u_T) \end{pmatrix}$$

by triangularizing the system of linear equations.

- Note that the first and last blocks of rows need a special treatment.
  1. Pre-multiply the $t = 1$ rows by $\left(f_y^1\right)^{-1}$: $f_y^1 \rightarrow I_n$
  2. Substract $f_{y_-}^2$ times the new $t = 1$ rows to $t = 2$ rows: $f_{y_-}^2 \rightarrow O_n$

$$\begin{pmatrix} I_n & g^1 & & & & & \\ O_n & f_y^2 - f_{y_-}^2 g^1 & f_{y_+}^2 & & & & \\ & f_{y_-}^3 & f_y^3 & f_{y_+}^3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & f_{y_-}^{T-1} & f_y^{T-1} & f_{y_+}^{T-1} & \\ & & & & f_{y_-}^T & f_y^T \end{pmatrix} \Delta Y = - \begin{pmatrix} d^1 \\ f(y_3, y_2, y_1, u_2) + f_{y_-}^2 d^1 \\ f(y_4, y_3, y_2, u_3) \\ \vdots \\ f(y_T, y_{T-1}, y_T, u_{T-1}) \\ f(y_{T+1}, y_T, y_{T-1}, u_T) \end{pmatrix}$$

where

$$\begin{aligned} g^1 &= \left(f_y^1\right)^{-1} f_{y_+}^1 \\ d^1 &= \left(f_y^1\right)^{-1} f(y_2, y_1, y_0, u_1) \end{aligned}$$

- For $t = 2, \ldots, T-1$:
  1. Pre-multiply the t rows by $\left(f_y^t - f_{y_-}^t\, g^{t-1}\right)^{-1}$
  2. Substract $f_{y_-}^{t+1}$ times the new t rows to $t+1$ rows

$$\begin{pmatrix} I_n & g^1 & & & & & \\ & I_n & g^2 & & & & \\ & & f_y^3 - f_{y_-}^3\, g^2 & f_{y_+}^3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & f_{y_-}^{T-1} & f_y^{T-1} & f_{y_+}^{T-1} & \\ & & & & f_{y_-}^T & f_y^T & \end{pmatrix} \Delta Y = - \begin{pmatrix} d_1 \\ d_2 \\ f(y_4, y_3, y_2, u_3) + f_{y_-}^3\, d_2 \\ \vdots \\ f(y_T, y_{T-1}, y_T, u_{T-1}) \\ f(y_{T+1}, y_T, y_{T-1}, u_T) \end{pmatrix}$$

where

$$g^t = \left(f_y^t - f_{y_-}^t\, g^{t-1}\right)^{-1} f_{y_+}^t$$

$$d_t = \left(f_y^t - f_{y_-}^t\, g^{t-1}\right)^{-1} \left(f(y_{t+1}, y_t, y_{t-1}, u_t) + f_{y_-}^t\, d_{t-1}\right)$$

Note that for the last block of rows we only need to apply the first transformation.

- In the end the system of linear equations looks like:

$$\begin{pmatrix} I_n & g^1 & & & & \\ & I_n & g^2 & & & \\ & & I_n & g^3 & & \\ & & & \ddots & \ddots & \\ & & & & I_n & g^{T-1} \\ & & & & & I_n \end{pmatrix} \Delta Y = - \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{T-1} \\ d_T \end{pmatrix}$$

with

$$d_T = \left(f_y^T - f_{y\_}^T \, g^{T-1} \right)^{-1} \left(f(y^\star, y_T, y_{T-1}, u^\star) + f_{y\_}^T \, d_{T-1} \right)$$

- The system is then solved by backward iteration:

$$\begin{aligned} y_T^{k+1} &= y_T^k - d_T \\ y_{T-1}^{k+1} &= y_{T-1}^k - d_{T-1} - g^{T-1}(y_T^{k+1} - y_T^k) \\ &\vdots \\ y_1^{k+1} &= y_1^k - d_1 - g^1(y_2^{k+1} - y_2^k) \end{aligned}$$

- We do not need to ever store the whole jacobian: only the g^s and d_s have to be stored.

- We compute the inverse of T $n \times n$ matrices, f_y^1 and $f_y^t - f_{y\_}^t \, g^{t-1}$ for $t = 2, \ldots, T$, instead of the inverse of an $nT \times nT$ matrix (remember that the number of flops for solving a system of linear equations is a third order polynomial of the number of equations).

- This approach was the default method in Dynare $\leq$ 4.2.

Solution of Perfect Foresight models

Sparse matrix algebra

- Sparse matrix algebra libraries are now widely available.
- The jacobian of the PF model is a sparse matrix because:
  - We have a lots of zero blocks (see previous slides).
  - The $f^t_{y_+}$, f^t_y and $f^t_{y_-}$ are themselves sparse because.
- Sparse matrices are stored as a list of triplets (i, j, v) where (i, j) is a matrix coordinate and v a non-zero value.

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 3 & 1 \\ 3 & 1 & 1 \end{pmatrix}$$

  The percentage of non zero elements has to be very small, otherwise there is no advantage in using sparse matrices.
- A lot of optimized algorithms for such matrices (including solution for system of linear equations, $\mathbf{A}x = b$).

Typology of simulations

One can imagine a lot of possible scenarii:

- Impulse Response Functions: paths of the endogenous variables after a transitory/permanent shock in period 1.
- Transitions from one steady state to another.
- Transitory/permanent expected shock in period $t > 1$ (for instance, an anounced fiscal reform would be an expected permanent shock).
- Simulation conditional on an expected path for the exogenous variables (for instance, the future demographic structure can be expected at time 1).
- Simulation with non expected shocks (surprises).

Except the last case (which requires more work), these scenarii can be easily handled by Dynare.

Perfect foresight models in Dynare (1/4)

- Declaration of the endogenous variables (`var` command).
- Declaration of the exogenous variables (`varexo` command).
- Declaration of the parameters (`parameters` command).
- Calibration of the parameters (as in matlab).
- Definition of the model (in a `model` block).

⚠ *Timing convention*

Variables decided at time t must be indexed by t.

For instance, the capital stock used in production at time t is decided at time $t-1$, thus output at time t has to be written $y_t = k_{t-1}^{\alpha}$.

$\Rightarrow$ Thanks to this timing convention, Dynare is able to identify the predetermined (state) variables.

Perfect foresight models in Dynare (2/4)

Example

Content of rbc.mod

```
var Consumption, Capital, LoggedProductivity;

varexo LoggedProductivityInnovation;

parameters beta, alpha, delta, rho;

beta = .985;
alpha = 1/3;
delta = alpha/10;
rho = .9;

model;

[name='Euler equation'] // This is an equation tag!
1/Consumption = beta/Consumption(1)*(alpha*exp(LoggedProductivity(1))*Capital^(alpha-1)+1-delta)
      ;

[name='Physical capital stock law of motion']
Capital = exp(LoggedProductivity)*Capital(-1)^alpha+(1-delta)*Capital(-1)-Consumption;

[name='Logged productivity law of motion']
LoggedProductivity = rho*LoggedProductivity(-1)+LoggedProductivityInnovation;

end;
```

Leads. If `X` is a variable, X_t, then `X(1)` is X_{t+1} and more generally `X(p)` with `p` a positive integer is X_{t+p}.

Lags. If `X` is a variable, X_t, then `X(-1)` is X_{t-1} and more generally `X(-p)` with `p` a positive integer is X_{t-p}.

Note that the "paper" version of the transition equation, $k_{t+1} = e^{at} k_t^{\alpha} + (1 - \delta)k_t - c_t$, translates into:

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

in the Dynare language, due to the timing convention. Thanks to this timing convention, Dynare understands that the physical capital used at time t in production is given at time t, *ie* that the capital stock is a predetermined variable. At time t the central planner (or household) can choose the capital stock that will be used tomorrow (through its consumption/investment decision) but not the capital stock currently used. Some Dynare users claim that Dynare is able to identify the predetermined variables (states) and non predetermined variables (controls). This statement is wrong, because the status of the variables is dictated by the timing of the variables (there is a perfect mapping between the timing and status of the variables). It is equivalent to declare the nature of a variable (predetermined vs. non predetermined) and to decide its timing.

An alternative interpretation of the timing in Dynare is that Dynare is uses a "stock at the end of the period" concept instead of a "stock at the beginning of the period" convention.

If you really do not like the Dynare's timing convention (*ie* if you prefer to adopt the same timing in your paper and the `mod` file), you have to declare the list of the predetermined variables using the `predetermined_variables` command. For instance, after the first line in rbc.mod, we should add

```
predetermined_variables Capital;
```

and change the model's equations as follows:

```
model;

[name='Euler equation'] // This is an equation tag!
1/Consumption=beta/Consumption(1)*(alpha*exp(LoggedProductivity(1))*Capital(1)^(alpha-1)+1-delta
      );

[name='Physical capital stock law of motion']
Capital(1)=exp(LoggedProductivity)*Capital^alpha+(1-delta)*Capital-Consumption;

[name='Logged productivity law of motion']
LoggedProductivity=rho*LoggedProductivity+LoggedProductivityInnovation;

end;
```

The variables and parameters used in the model block must be declared as such before. Note however that it is possible to declare local variables inside the `model` block using the # symbol. Suppose we have a CES production function of the form:

$$y = \left(\alpha k^{\rho} + (1 - \alpha) l^{\rho}\right)^{\frac{1}{\rho}}$$

in the model. Instead of calibrating ρ we may prefer to calibrate the elasticity of substitution between k and l, *ie* $\epsilon = 1/1-\rho$. In this case, we declare ϵ as a parameter (*ie* after the `parameters` keyword) and at the top of the model block we write the definition of rho:

```
model;
#rho = (epsilon-1)/epsilon;
...

end;
```

Note that `rho` is unknown outside of the scope defined by the `model` block. Behind the scene, Dynare replaces all occurences of `rho` by `(epsilon-1)/epsilon`.

Perfect foresight models in Dynare (3/4)

Steady state (numerical routine)

- Dynare can compute the steady state of the model...
- Using a Newton like solver, so we need to define an initial guess. The `initval` block can be used for that purpose (see rbc1.mod):

```
initval;
Consumption = 2;
Capital = 15;
LoggedProductivity = 0;
LoggedProductivityInnovation = 0;
end;

steady;
```

- Different algorithms are available to compute the steady state, see section 4.10 in the manual.
- But if the initial guess is too far from the solution, the solution algorithms may fail (see rbc2.mod for instance). n case of failure, Dynare returns the residuals of the static equations, this may help to identify the problem.
- The analytical steady state should be provided when available.

Perfect foresight models in Dynare (3/4)

Steady state (closed form, 1)

- The simplest way to define the steady state is to use the `steady_state_model` block.

Content of rbc4.mod

```
steady_state_model;
  LoggedProductivity = 0;
  Capital = (alpha/(1/beta-1+delta))^(1/(1-alpha));
  Consumption = Capital^alpha-delta*Capital;
end;
```

- Parameters can be updated according to steady state constraints.
- External matlab routines may be called in this block, but it is not allowed to use loops or conditional structures.
- If the analytical steady state is only available for a subset of variables conditional on the others, a solver can be called in this block to solve the concentrated static model.

Perfect foresight models in Dynare (3/4)

Steady state (closed form, 2)

- A more flexible approach is to write a matlab routine that computes the steady state. This routine must be saved in a file called `<BASE_NAME_OF_THE_MOD_FILE>_steadystate.m`

Content of rbc5_steadystate.m

```
function [ys, check] = rbc5_steadystate(ys, exe)
    global M_
    check = 0;
    beta = M_.params(1); alpha = M_.params(2); delta = M_.params(3); rho = M_.params(4);
    LoggedProductivity = 0;
    Capital = (alpha/(1/beta-1+delta))^(1/(1-alpha));
    Consumption = Capital^alpha-delta*Capital;
    ys = [Consumption; Capital; LoggedProductivity];
```

- The routine has to return `check=0` if the steady state exist. A nonzero value means that there is a problem with the steady state.
- All legal matlab statements are allowed, contrary to the previous approach with the `steady_state_model`.
- But flexibility comes at a price: this approach is slower than the previous one (why?).

Perfect foresight models in Dynare (4/4)

Simulation of a transition (1)

- Initial condition is different from the steady state.
- Simulate transition to the steady state.
- Use the `initval` block

Content of rbc6.mod

```
initval;
  LoggedProductivityInnovation = 0;
  LoggedProductivity = .05;
  Capital = 17.5;
end;

endval;
  LoggedProductivityInnovation = 0;
end;

steady;

perfect_foresight_setup(periods=200);
perfect_foresight_solver;
```

- The `initval` block sets the initial condition for the states, while the `endval` block with the `steady` command set the terminal condition.

Perfect foresight models in Dynare (4/4)

Simulation of a transition (2)

- Initial condition is the steady state.
- A permanent shock shifts upward the productivity.
- Change the value of the innovation (> 0) in the `endval` block and compute the new steady state with the `steady` command.

Content of rbc7.mod

```
initval;
  LoggedProductivityInnovation = 0;
end;

steady;

endval;
  LoggedProductivityInnovation = .01;
end;

steady;

perfect_foresight_setup(periods=200);
perfect_foresight_solver;
```

- With these commands we implicitly set the innovations equal to .01 from period 1 to period 200. Initial and terminal conditions are different steady states.

Perfect foresight models in Dynare (4/4)

Transition induced by a shock in period 1

- Initial condition is the steady state.
- A shock in period 1 temporarily increases productivity.
- We use the `shocks` block.

Content of rbc8.mod

```
initval;
  LoggedProductivityInnovation = 0;
end;

steady;

endval;
  LoggedProductivityInnovation = 0;
end;

steady;

shocks;
var LoggedProductivityInnovation;
periods 1;
values .1;
end;

perfect_foresight_setup(periods=200);
perfect_foresight_solver;
```

- With these commands we implicitly set the innovations equal to $(.1, 0, 0, \dots)$

Perfect foresight models in Dynare (4/4)

Transition induced by a sequence of expected shocks

- Initial condition is the steady state.
- Shocks in periods 1 to 5 temporarily hit productivity.

Content of rbc9.mod

```
initval;
  LoggedProductivityInnovation = 0;
end;

steady;

endval;
  LoggedProductivityInnovation = 0;
end;

steady;

sequence_of_shocks = [.1; .2; .2; .2^2; .2^4];

shocks;
var LoggedProductivityInnovation;
periods 1:5;
values (sequence_of_shocks);
end;

perfect_foresight_setup(periods=200);
perfect_foresight_solver;
```

- With these commands we implicitly set the innovations equal to $(.1, .2, -.2, -.2^2, -.5, 0, \dots)$

Perfect foresight models in Dynare (4/4)

Non expected shocks (1)

- It is also possible to simulate paths with an unexpected sequence of shocks. But there is no interface for that in Dynare so we need to add some matlab code in the mod file.
- Basically, the idea it to solve the perfect foresight model for each new shock with an initial condition given by the solution obtained for the previous shock. If we have p non expected shocks:

In period 1 $a_0^1 = a_0$ and $k_0^1 = k_0$ are given. Conditionally on the observed productivity shock, $u_1^1 = \epsilon_1$, we solve a perfect foresight model assuming that no shock will hit the economy tomorrow, and save $c_1 = c_1^1$, $k_1 = k_1^1$ and $a_1 = a_1^1$.

In period 2 $a_0^2 = a_1$ and $k_0^2 = k_1$ are given. Conditionally on the observed productivity shock, $u_1^2 = \epsilon_2$, we solve a perfect foresight model assuming that no shock will hit the economy tomorrow, and save $c_2 = c_1^2$, $k_2 = k_1^2$ and $a_2 = a_1^2$.

...

In period p $a_0^p = a_{p-1}$ and $k_0^p = k_{p-1}$ are given. Conditionally on the observed productivity shock, $u_1^p = \epsilon_p$, we solve a perfect foresight model assuming that no shock will hit the economy tomorrow, and save $c_{p\ldots} = c_{1\ldots}^p$, $k_{p\ldots} = k_{1\ldots}^p$ and $a_{p\ldots} = a_{1\ldots}^p$.

Perfect foresight models in Dynare (4/4)

Non expected shocks (2)

- This algorithm is related to the extended path approach that we will present in the next section.

Content of rbc10.mod

```
1  initval; LoggedProductivityInnovation = 0; end; steady;
2
3  endval; LoggedProductivityInnovation = 0; end; steady;
4
5  sequence_of_shocks = [.1; .2; .2; .2^2; .2^4]; // [.1; .2; -.2; -.2^2; -.5];
6
7  shocks;
8      var LoggedProductivityInnovation; periods 1; values (sequence_of_shocks(1));
9  end;
10
11 yy = oo_.steady_state;
12 // First period
13 perfect_foresight_setup(periods=200);
14 perfect_foresight_solver;
15 yy = [yy, oo_.endo_simul(:,2)];
16
17 // Following periods
18 for i=2:length(sequence_of_shocks)
19   oo_.exo_simul(2) = sequence_of_shocks(i);
20   oo_.endo_simul(:,1) = yy(:,end);
21   perfect_foresight_solver;
22   yy = [yy, oo_.endo_simul(:,2)];
23 end;
24
25 yy = [yy, oo_.endo_simul(:,3:end)]; // Complete the paths with the last simulation.
```

Some Remarks about the code in rbc10.mod:

Line 5. The sequence of non expected innovations, defined in `sequence_of_shocks`, must be a column vector. This vector can be the result returned by a matlab routine. Note that if some of the innovations are too large, Dynare may fail in solving the model. The code is valid for any number of non expected shocks.

Line 8. The value of the innovation must be between parenthesis because it is defined as a matlab expression (here the element of an array).

Line 11. Initialization of the matrix that will hold the generated paths. The initial condition is the steady state, stored in `oo_.steadystate`. The global variable `oo_` is a global matlab structure containing all the results.

Line 15. The results of the perfect foresight solver are stored in `oo_.endo_simul`, a $n \times (T+2)$ matrix (where n is the number of endogenous variables and T the simulation horizon). The variables (rows) are ordered consistently with the order of declaration in the mod file. So in our example (see rbc.mod) the first, second and third rows are respectively for `Consumption`, `Capital` and `LoggedProductivity`. The first column is for the initial condition (y_0) and the last column for the terminal condition (y_{T+1}) $\Rightarrow$ The first period of the simulation is stored in the second column, and more generally period t is stored in column $t+1$. Note also that the storing is consistent with Dynare's timing convention. In our example, the second row second column is k_1, the capital stock decided in period one and used in period two.

The innovations are stored in `oo_.exo_simul`, a $(T+2) \times q$ matrix, where q is the number of declared shocks (`varexo`) in the model. This matrix is not an output of the perfect foresight solver but an input, where the expected shocks are defined. The second row contains the shocks values declared for period 1, and more generally the time t shocks values are stored in row $t+1$. The last row $(T+1)$ is filled by the content of the `endval` block (if values are assigned to the exogenous variables as it is the case when we consider permanent shocks, see rbc7.mod). By default the elements of this matrix are all zeros. Non zero values are defined by the content of the `shocks` block.

In line 15, we append the second column of `oo_.endo_simul`, corresponding to the choices of the agents (or central planner) in reaction to the surprise on productivity, to `yy` (see the description of the algorithm in the previous page).

Perfect foresight models in Dynare (4/4)

Expected versus Surprise shocks

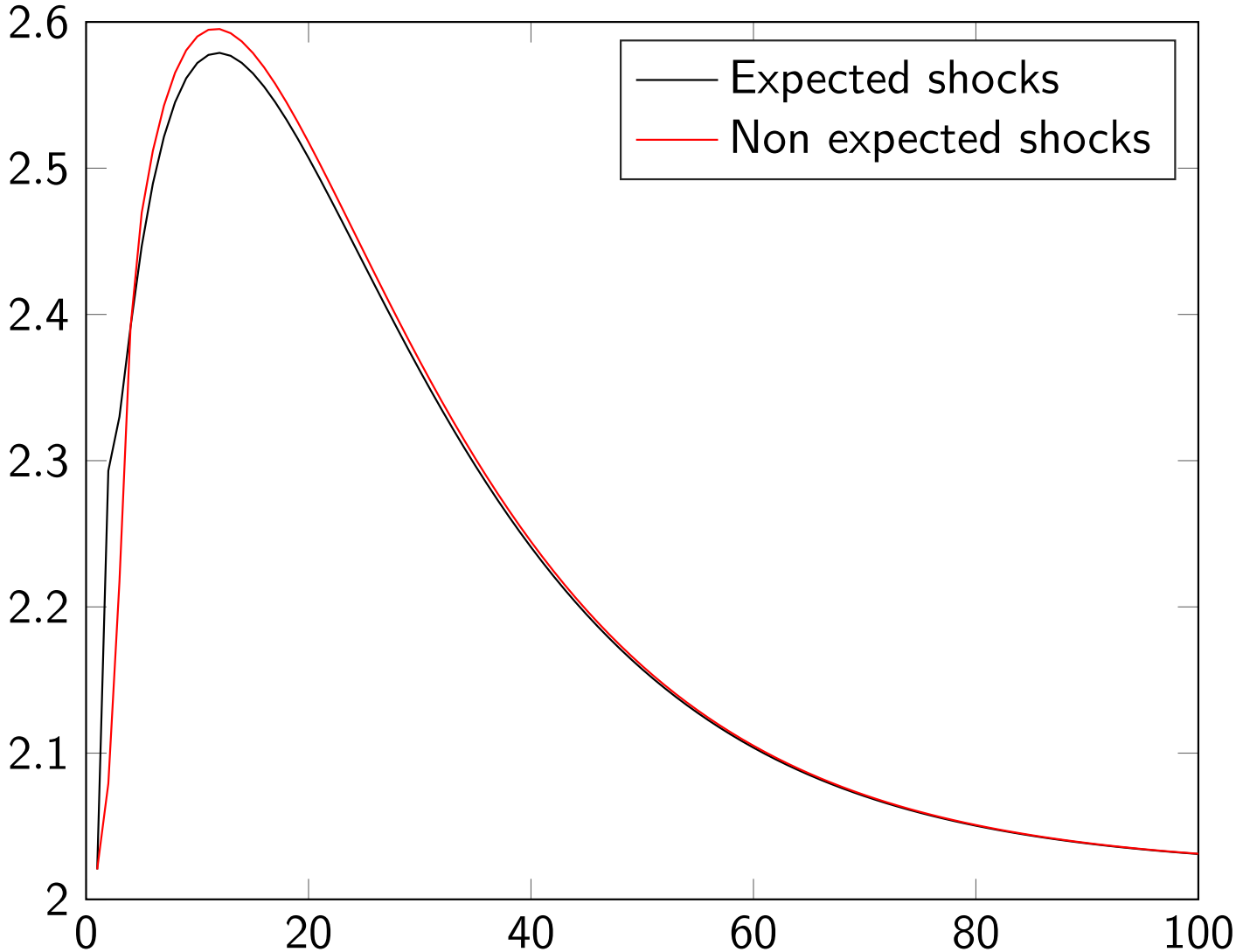
- In the two next slides we compare the paths for the capital stock and consumption when the sequence of shocks is expected or non expected.
- For this exercise, we considered the following sequence of shocks:

$$.1, .2, .2, .2^2, .2^4, 0, 0, \ldots$$

- The data are generated with rbc9.mod and rbc10.mod
- If the (positive) productivity shocks are expected the consumption reacts upward more aggressively (during the first periods) to benefit from the anticipated growth of the physical capital return.
- Consequently, in the first period the physical capital stock is almost flat when the shocks are expected. The path of the physical capital stock has a smaller amplitude when the shocks are expected ($\rightarrow$ smoothing).

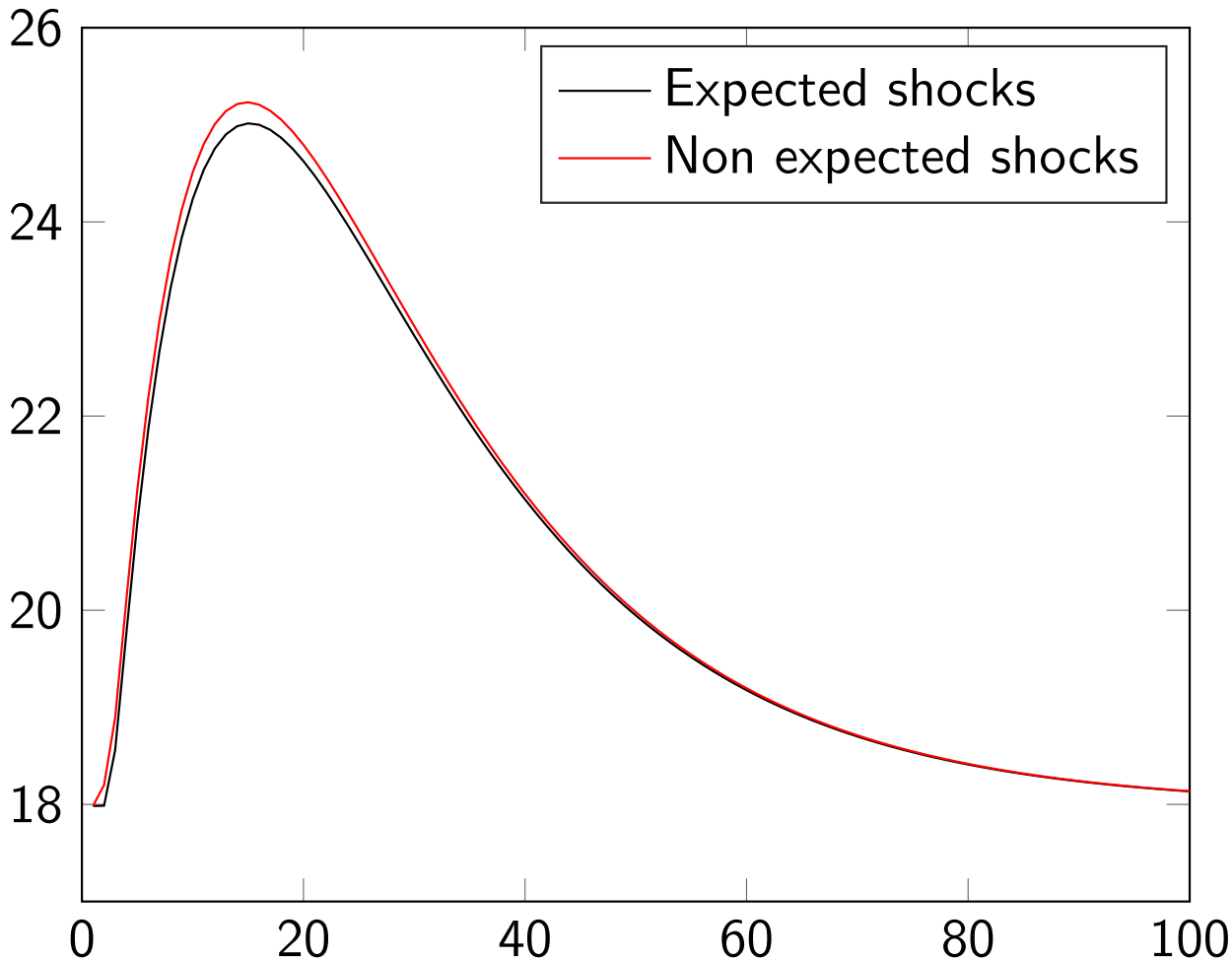
Perfect foresight models in Dynare (4/4)

Expected versus Surprise shocks (Consumption)



Perfect foresight models in Dynare (4/4)

Expected versus Surprise shocks (Capital)



Extended path

Introduction

- What if we need to generate time series?
- Extended path approach introduced by Fair and Taylor (Econometrica, 1983).
- Use the same strategy than for simulating the perfect foresight model with a sequence of unexpected shocks.

$\Rightarrow$ We just need to take the sequence of unexpected shocks from a random number generator.

Extended path

Model to be solved

$$s_t = \mathrm{Q}(s_{t-1}, u_t) \tag{2a}$$
$$\mathrm{F}\left(y_t, x_t, s_t, \mathbb{E}_t\left[\mathscr{E}_{t+1}\right]\right) = 0 \tag{2b}$$
$$\mathrm{G}(y_t, x_{t+1}, x_t, s_t) = 0 \tag{2c}$$
$$\mathscr{E}_t = \mathrm{H}(y_t, x_t, s_t) \tag{2d}$$

s_t is a $n_s \times 1$ vector of exogenous state variables, $u_t \sim \mathrm{BB}(0, \Sigma_u)$ is a $n_u \times 1$ multivariate innovation, x_t is a $n_x \times 1$ vector of endogenous state variables, y_t is a $n_y \times 1$ vector of non predetermined variables and $\mathscr{E}_t$ is a $n_{\mathscr{E}} \times 1$ vector of auxiliary variables. $\mathscr{E}_t$ is a vector gathering the nonlinear functions of the endogenous variables that appear under the condition expectation.

Functions Q, F, G and H are assumed to be continuous functions (but not necessarily differentiable). They respectively collect the equations for the exogenous variables, the Euler equations and static equilibrium conditions, the transition equations, and the definitions of the auxiliary variables.

Obviously, the basic RBC model can be cast into the previous form:

(2a) $\rightarrow a_t - \rho a_{t-1} - \epsilon_t = 0$ where ϵ_t, usually a Gaussian white noise, is for the random unexpected shocks.

(2b) $\rightarrow \frac{1}{c_t} - \beta^{-1}\mathbb{E}_t\left[\mathscr{E}_{t+1}\right] = 0$, the Euler equation.

(2c) $\rightarrow k_{t+1} - e^{a_t} k_t^{\alpha} - (1-\delta)k_t + c_t = 0$, the transition equation.

(2d) $\rightarrow$, $\mathscr{E}_t = \frac{\alpha e^{a_t} k_t^{\alpha-1}+1-\delta}{c_t}$, the definition of auxiliary variable $\mathscr{E}_t$.

Note the presence of the conditional expectation, appearing because of the non expected shocks. The model defined in (2a)-(2d) is a rational expectation model. In the next slides we will see how we deal with the conditional expectation (the extended path approach basically removes the conditional expectation, while its extension, the stochastic extended path method, uses numerical integration methods to compute the expectations).

Note also that we do not use the timing convention of Dynare here and in the description of the algorithms.

Extended path

Rational

- The extended path approach creates a stochastic simulation as if only the shocks of the current period were random.

- Substituting (2a) in (2d), define:

$$\mathscr{E}_t = \mathscr{E}(y_t, x_t, s_{t-1}, u_t) = \mathrm{H}(y_t, x_t, \mathrm{Q}(s_{t-1}, u_t))$$

- The Euler equations (2b) can then be rewritten as:

$$\mathrm{F}\left(y_t, x_t, s_t, \mathbb{E}_t\left[\mathscr{E}(y_{t+1}, x_{t+1}, s_t, u_{t+1})\right]\right) = 0$$

- The Extended path algorithm consists in replacing the previous Euler equations by:

$$\mathrm{F}\left(y_t, x_t, s_t, \mathscr{E}(y_{t+1}, x_{t+1}, s_t, 0)\right) = 0$$

$\Rightarrow$ Set the future innovations to their expectation (0). As if it was legal to pass the expectation inside the $\mathscr{E}$ function (**certainty equivalence**).

Extended path

Algorithm

Algorithm 1 Extended path algorithm

1. $H \leftarrow$ Set the horizon of the perfect foresight (PF) model.
2. $(x^\star, y^\star) \leftarrow$ Compute steady state of the model
3. $(s_0, x_1) \leftarrow$ Choose an initial condition for the state variables
4. **for** $t = 1$ to T **do**
5. $\quad u_t \leftarrow$ Draw random shocks for the current period
6. $\quad (y_t, x_{t+1}, s_t) \leftarrow$ Solve a PF with $y_{t+H+1} = y^\star$
7. **end for**

Extended path

Time t nonlinear problem

For each period we need to solve the following non linear problem with respect to the paths for the endogenous variables:

$$
\begin{aligned}
s_t &= \mathrm{Q}(s_{t-1}, u_t) \\
0 &= \mathrm{F}(y_t, x_t, s_t, \mathscr{E}(y_{t+1}, x_{t+1}, s_t, 0)) \\
0 &= \mathrm{G}(y_t, x_{t+1}, x_t, s_t) \\
s_{t+1} &= \mathrm{Q}(s_t, 0) \\
0 &= \mathrm{F}(y_{t+1}, x_{t+1}, s_{t+1}, \mathscr{E}(y_{t+2}, x_{t+2}, s_{t+1}, 0)) \\
0 &= \mathrm{G}(y_{t+1}, x_{t+2}, x_{t+1}, s_{t+1}) \\
&\vdots \\
s_{t+h} &= \mathrm{Q}(s_{t+h-1}, 0) \\
0 &= \mathrm{F}(y_{t+h}, x_{t+h}, s_{t+h}, \mathscr{E}(y_{t+h+1}, x_{t+h+1}, s_{t+h}, 0)) \\
0 &= \mathrm{G}(y_{t+h}, x_{t+h+1}, x_{t+h}, s_{t+h}) \\
&\vdots \\
s_{t+H} &= \mathrm{Q}(s_{t+H-1}, 0) \\
0 &= \mathrm{F}(y_{t+H}, x_{t+H}, s_{t+H}, \mathscr{E}(y^\star, x_{t+H+1}, s_{t+H}, 0)) \\
0 &= \mathrm{G}(y_{t+H}, x_{t+H+1}, x_{t+H}, s_{t+H})
\end{aligned}
$$

We create the time series by concatenating the solutions for y_t and x_{t+1} $(t = 1, \ldots, T)$.

Extended path

Discussion

- This approach takes full account of the *deterministic* non linearities...
- ... But neglects the Jensen inequality by setting future innovations to zero (the expectation).
- We do not solve the rational expectation model! We solve a model where the agents believe that the economy will not be perturbed in the future. They observe new realizations of the innovations at each date but do not update this belief...
- Uncertainty about the future does not matter here (for instance, an hypothetical economy simulated with this approach would not save for precautionary motives).
- EP $>$ First order perturbation (which shares the certainty equivalence property with the EP approach).

Extended path

EP with Dynare

- Declare the variance of the innovations using the `shocks` block.
- Use the `extended_path` command.

Content of rbc11.mod

```
steady;

shocks;
    var LoggedProductivityInnovation = .01^2;
end;

extended_path(periods=1000);

plot(Simulated_time_series.Capital,Simulated_time_series.Consumption,'ok')
```

- `periods` is the size of the generated sample.
- By default, the horizon, H, is set equal to 400 (see option `solver_periods` in the reference manual).

Stochastic extended path

Introduction

- It is not possible to think about the importance of future uncertainty with the extended path approach.
- To circumvent this issue Dynare proposes an extension: the stochastic extended path.
- The strong assumption about future uncertainty can be relaxed by approximating the expected terms in the Euler equations (2b).
- We assume that, at time t, agents perceive uncertainty about realizations of $u_{t+1}, \ldots, u_{t+k}$ but not about the realizations of $u_{t+\tau}$ for all $\tau > k$ (which, again, are set to zero).
- Under this assumption, the expectations are approximated using numerical integration.

Stochastic extended path

Gaussian quadrature (univariate)

- Let X be a Gaussian random variable with mean zero and variance $\sigma_x^2 > 0$, and suppose that we need to evaluate $\mathbb{E}[\varphi(X)]$, where φ is a continuous function.

- By definition we have:

$$\mathbb{E}[\varphi(X)] = \frac{1}{\sigma_x\sqrt{2\pi}} \int_{-\infty}^{\infty} \varphi(x) e^{-\frac{x^2}{2\sigma_x^2}} \mathrm{d}x$$

- It can be shown that this integral can be approximated by a finite sum using the following result:

$$\int_{-\infty}^{\infty} \varphi(z) e^{-z^2} \mathrm{d}x = \sum_{i=1}^{n} \omega_i \varphi(z_i) + \frac{n!\sqrt{n}}{2^n} \frac{\varphi^{(2n)}(\xi)}{(2n)!}$$

where z_i $(i = 1, \ldots, n)$ are the roots of an order n Hermite polynomial, and the weights ω_i are positive and summing up to one (the error term is zero iff φ is a polynomial of order at most $2n - 1$).
$\rightarrow x_i = {}^{z_i}/_{\sigma_x\sqrt{2}}$

Stochastic extended path

Gaussian quadrature (multivariate)

- Let X be a multivariate Gaussian random variable with mean zero and unit variance, and suppose that we need to evaluate

$$\mathbb{E}[\varphi(X)] = (2\pi)^{-\frac{p}{2}} \int_{\mathbb{R}^p} \varphi(\boldsymbol{x}) e^{-\frac{1}{2}\boldsymbol{x}'\boldsymbol{x}} \mathrm{d}\boldsymbol{x}$$

- Let $\{(\omega_i, z_i)\}_{i=1}^n$ be the weights and nodes of an order n univariate Gaussian quadrature.

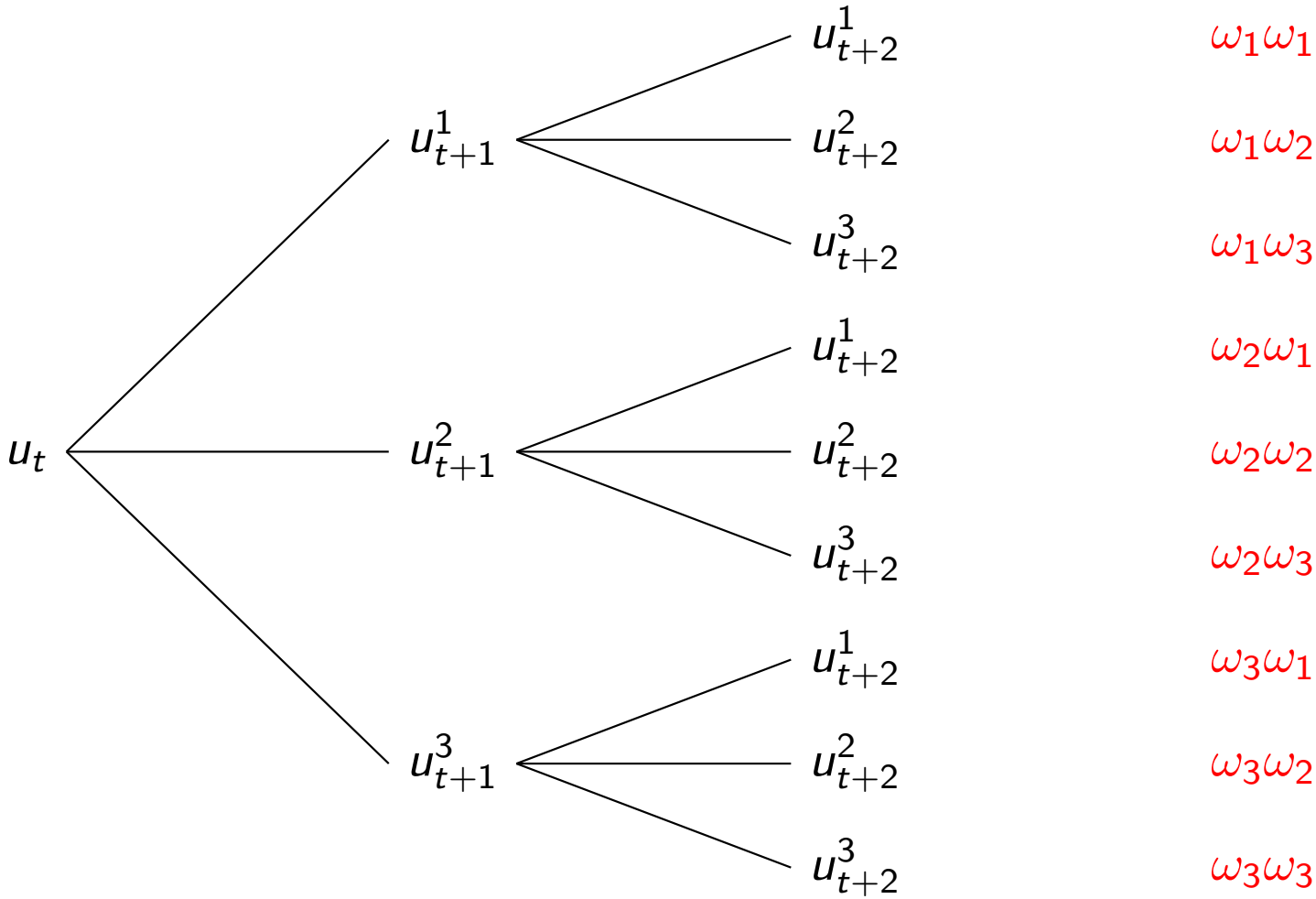
- This integral can be approximated using a tensor grid:

$$\int_{\mathbb{R}^p} \varphi(\boldsymbol{z}) e^{-\boldsymbol{z}'\boldsymbol{z}} \mathrm{d}\boldsymbol{z} \approx \sum_{i_1,\ldots,i_p=1}^{n} \omega_{i_1} \ldots \omega_{i_p} \varphi(z_{i_1}, \ldots, z_{i_p})$$

- **Curse of dimensionality:** The number of terms in the sum grows exponentially with the number of shocks.

Stochastic extended path

Forward histories (one shock, three nodes, order two SEP)



$\Rightarrow$ **Curse of dimensionality:** The tree of histories grows exponentially!

Stochastic extended path

Curses of dimensionality

- We face two curses of dimensionality:
  - Number of innovations (n_u).
  - Approximation order (k).

- However, the size of the problem grow "only" polynomially (because of the inversion of the jacobian matrix) with respect to the number of state variables.

- The relative advantage of this approach (compared with global approximation methods) is in models with a large number of states.

- It is possible to use alternative numerical integration routines, which overcome the exponential growth issues. But this is ongoing research and not yet stable in Dynare.

Stochastic extended path

Algorithm

Algorithm 2 Stochastic Extended path algorithm

1. $H \leftarrow$ Set the horizon of the *stochastic* perfect foresight (SPF) models.
2. $(x^\star, y^\star) \leftarrow$ Compute steady state of the model.
3. $\{(\omega_i, \delta_i)\}_{i=1}^{m} \leftarrow$ Get weights and nodes for numerical integration
4. $(s_0, x_1) \leftarrow$ Choose an initial condition for the state variables
5. **for** $t = 1$ to T **do**
6. $\quad u_t \leftarrow$ Draw random shocks for the current period
7. $\quad (y_t, x_{t+1}, s_t) \leftarrow$ Solve a SPF model with $y_{t+H+1} = y^\star$
8. **end for**

Stochastic extended path

SEP algorithm (order 1, time t nonlinear problem)

For $i = 1, \ldots, m$

$$
\begin{aligned}
s_t &= \mathrm{Q}(s_{t-1}, u_t) \\
0 &= \mathrm{F}\left(y_t, x_t, s_t, \textstyle\sum_i \omega_i \mathscr{E}(y^i_{t+1}, x_{t+1}, s_t, \delta_i)\right) \\
0 &= \mathrm{G}(y_t, x_{t+1}, x_t, s_t) \\
s^i_{t+1} &= \mathrm{Q}(s_t, \delta_i) \\
0 &= \mathrm{F}(y^i_{t+1}, x_{t+1}, s^i_{t+1}, \mathscr{E}(y^i_{t+2}, x^i_{t+2}, s^i_{t+1}, 0)) \\
0 &= \mathrm{G}(y^i_{t+1}, x^i_{t+2}, x_{t+1}, s^i_{t+1}) \\
&\vdots \\
s^i_{t+h} &= \mathrm{Q}(s^i_{t+h-1}, 0) \\
0 &= \mathrm{F}(y^i_{t+h}, x^i_{t+h}, s^i_{t+h}, \mathscr{E}(y^i_{t+h+1}, x^i_{t+h+1}, s^i_{t+h}, 0)) \\
0 &= \mathrm{G}(y^i_{t+h}, x^i_{t+h+1}, x^i_{t+h}, s^i_{t+h}) \\
&\vdots \\
s^i_{t+H} &= \mathrm{Q}(s^i_{t+H-1}, 0) \\
0 &= \mathrm{F}(y^i_{t+H}, x^i_{t+H}, s^i_{t+H}, \mathscr{E}(y^\star, x^i_{t+H+1}, s^i_{t+H}, 0)) \\
0 &= \mathrm{G}(y^i_{t+H}, x^i_{t+H+1}, x^i_{t+H}, s^i_{t+H})
\end{aligned}
$$

Stochastic extended path

SEP algorithm (order 2, time t nonlinear problem)

For all $(i,j) \in \{1,\dots,m\}^2$

$$
\begin{aligned}
s_t &= \mathrm{Q}(s_{t-1}, u_t)\\
0 &= \mathrm{F}\left(y_t, x_t, s_t, \textstyle\sum_i \omega_i \mathscr{E}(y^i_{t+1}, x_{t+1}, s_t, \delta_i)\right)\\
0 &= \mathrm{G}(y_t, x_{t+1}, x_t, s_t)\\
s^i_{t+1} &= \mathrm{Q}(s_t, \delta_i)\\
0 &= \mathrm{F}(y^i_{t+1}, x_{t+1}, s^i_{t+1}, \textstyle\sum_j \omega_j \mathscr{E}(y^{i,j}_{t+2}, x^i_{t+2}, s^i_{t+1}, \delta_j))\\
0 &= \mathrm{G}(y^i_{t+1}, x^i_{t+2}, x_{t+1}, s^i_{t+1})\\
s^{i,j}_{t+1} &= \mathrm{Q}(s^i_{t+1}, \delta_j)\\
&\vdots\\
s^{i,j}_{t+h} &= \mathrm{Q}(s^{i,j}_{t+h-1}, 0)\\
0 &= \mathrm{F}(y^{i,j}_{t+h}, x^{i,j}_{t+h}, s^{i,j}_{t+h}, \mathscr{E}(y^{i,j}_{t+h+1}, x^{i,j}_{t+h+1}, s^{i,j}_{t+h}, 0))\\
0 &= \mathrm{G}(y^{i,j}_{t+h}, x^{i,j}_{t+h+1}, x^{i,j}_{t+h}, s^{i,j}_{t+h})\\
&\vdots\\
s^{i,j}_{t+H} &= \mathrm{Q}(s^{i,j}_{t+H-1}, 0)\\
0 &= \mathrm{F}(y^{i,j}_{t+H}, x^{i,j}_{t+H}, s^{i,j}_{t+H}, \mathscr{E}(y^\star, x^{i,j}_{t+H+1}, s^{i,j}_{t+H}, 0))\\
0 &= \mathrm{G}(y^{i,j}_{t+H}, x^{i,j}_{t+H+1}, x^{i,j}_{t+H}, s^{i,j}_{t+H})
\end{aligned}
$$

Stochastic extended path

SEP with Dynare

- Add options to the `extended_path` command.
  - `order` is the value of k (approximation order of the SEP)
  - In the current version of Dynare there is no interface for controlling the number of nodes for the Gaussian quadrature. The number of nodes has to be directly set in the `options_` global structure.

Content of rbc12.mod

```
steady;

shocks;
    var LoggedProductivityInnovation = .01^2;
end;

options_.ep.stochastic.quadrature.nodes = 3;
extended_path(periods=1000,order=1);

plot(Simulated_time_series.Capital,Simulated_time_series.Consumption,'ok')
```