Table of Contents

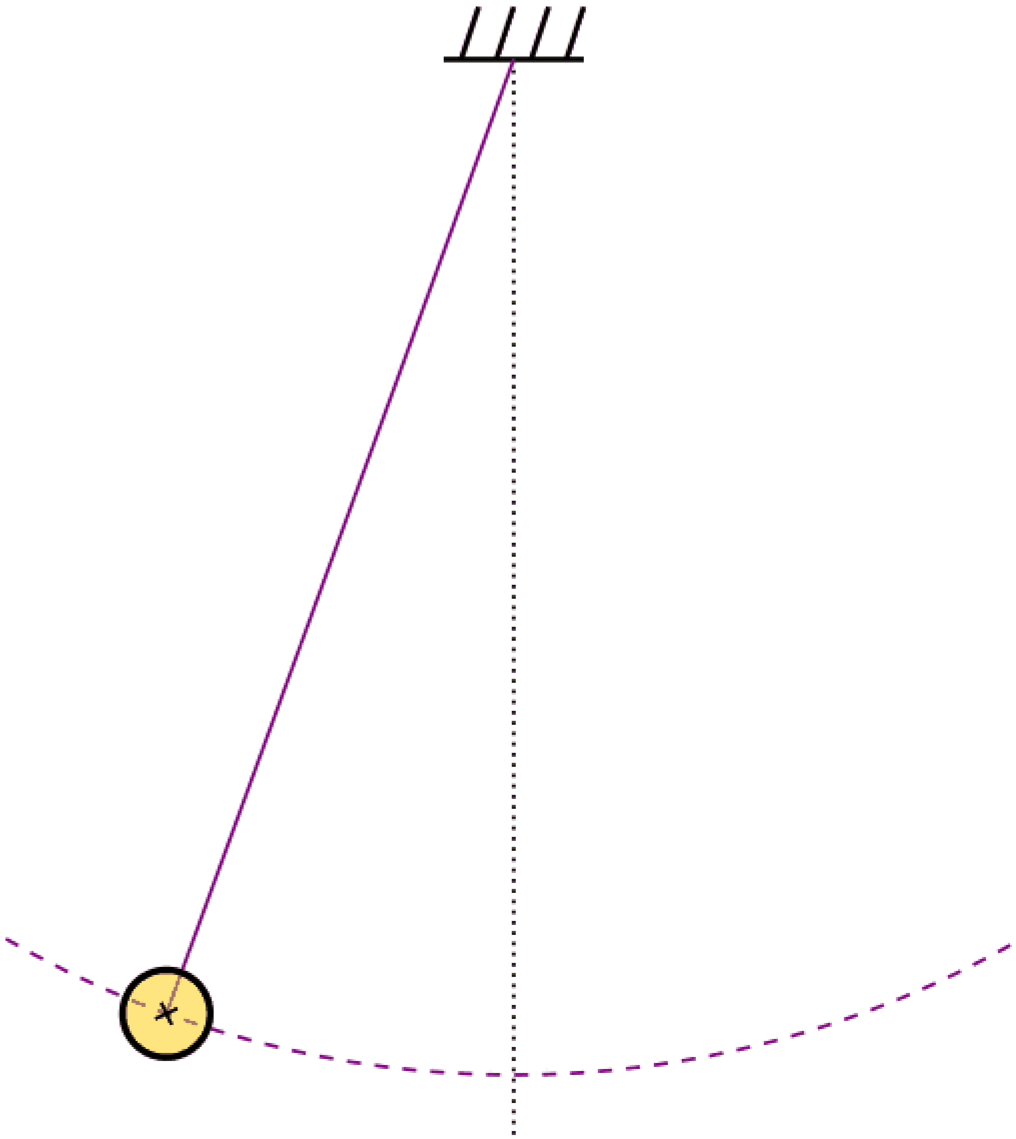
Essential numerical tools and

perturbation analysis (2.a)

Day 2: Perturbation Analysis

Pablo Winant

African Econometric Society Workshop 14/12/23



When doing modeling (not only in econonomics), we often encounters nonlinear systems that have no closed form but a steady-state.

The solution then consists in characterizing the variables in a neighbourhood of the steady-state and solve for them using a *linear* approximation of the model.

The result is a *linear* approximation of the solution.

Tip

In principle, the same approach can be carried out for higher orders of approximation.

Implicit Function Theorem

The perturbation approach is closely related to the Implicit Functions Theorem.

Assume we know the relation between two variables $x \in \mathbf{R}^n$ and $y \in \mathbf{R}^n$: $f(x, y) = 0$.

Assume we know that a particular pair satisfies this relation $f(x_0, y_0)$.

Then, if $f_y'(x_0, y_0)$ is invertible, it is possible to construct a local approximation of a function φ such that $y = \varphi(x)$ at least in the vincinity of x_0.

In practice, one *applies* the method of unknown coefficients.

Tip

Under suitable assumptions on f, the I.F.T. actually implies the *existence* of a such a function on a global definition space.

▶ unknown coefficients

Two very topical applications in (macro)-economics

DSGE Model

The perturbation approach works so well it has spurred the development of a full class of models, the Dynamic Stochastic General Equilibrium Models (DSGE).

These models were able to include all elements from the New Keynesian synthesis and the availability of an *easy-to-use solution method*, made it possible to:

- incorporate new theories into the models
- interpret models predictions using impulse response functions
- estimate models using statistical tools

Nowadays all central banks have some form of DSGE model

- Generally based on *midsize model* from Smets & Wouters (10 equations)
    - (IMF/GIMF, EC/Quest, ECB/, NYFed/FRBNY)
- but have grown up a lot (>>100 equations)
- Institutions are (slowly) diversifying their model portfolios
    - CGEs
    - Agent-based
    - Semi-structural models (again)
    - Heterogenous Agents

Tip

This paper argues that a big determinant in the development of DSGE models is easy availability of modeling tools like Dynare.

Heterogeneous Agents Modeling

Nowadays there are lots of models featuring a continuous distribution of agents (firms, households, banks)

These models are fully nonlinear (possibly with kinks) when it comes to the idiosyncratic decision variables of each agent.

But the dependence in aggregate shocks is highly non-tractable.

A common approach consists in:

- computing a fully-nonlinear stationary distributinon of agents (the steady-state)
- **perturbing it with respect to the aggregate shocks**

💡 Said perturbation has a lot in common with simple DSGEs

DSGE Modeling

Dynare has popularized the following modeling approach:

- write a model:
  - equations
  - calibration of paramters
  - steady-state guess
- check that steady-state is correct
  - if not try to find one numerically
  - if no luck: go back to the model
- check that model is well-specified (Blanchard-Kahn)
  - if not: go back to modeling
- enjoy the simulations...

Tip

Dynare

- an opensource tool to solve DSGE models
- a modelling language
- primary Matlab based but
  - Fortran, Gauss, in the past
  - versions in Octave
  - a WIP in Julia

The RBC Model

The planner version:

- objective: $\max_{\substack{c_t,n_t \\ c_t\geq 0, y_t\geq c_t, n_t\geq 0, 1\geq n_t}} \mathbb{E}_0\left[\sum \beta^t \left(\frac{c_t^{1-\sigma}}{1-\sigma} - \chi \frac{n_t^{\eta-1}}{\eta-1}\right)\right]$

Under the following constraints:

- production: $q_t = \exp(z_t) k_{t-1}^{\alpha} n_t^{1-\alpha}$
- investment: $i_t = q_t - c_t$
- capital law of motion: $k_t = (1-\delta) k_{t-1} + i_t$
- productivity law of motion: $z_t = (1-\rho) z_{t-1} + \epsilon_t$

Where ϵ_t is an i.i.d shock, normally distributed.

RBC Model First Order Conditions

The maximization program yields the following first order conditions:

- optimal investment: $(c_t)^{-\gamma} = \beta\mathbb{E}_t\left[(c_{t+1})^{-\gamma}\left((1-\delta) + \alpha e^{z_{t+1}}{\boldsymbol{k_t}}^{\alpha-1}n_{t+1}^{1-\alpha}\right)\right]$
- optimal labour supply: $\chi n_t^{\eta} = (1-\alpha)e^{z_t}{\boldsymbol{k_{t-1}}}^{\alpha}(n_t)^{-\alpha}(c_t)^{-\gamma}$
- production: $q_t = \exp(z_t){\boldsymbol{k_{t-1}}}^{\alpha}n_t^{1-\alpha}$
- investment: $i_t = q_t - c_t$
- capital law of motion: $k_t = (1-\delta)\boldsymbol{k_{t-1}} + i_t$
- productivity law of motion: $z_t = (1-\rho)\boldsymbol{z_{t-1}} + \boldsymbol{\epsilon_t}$

Where ϵ_t is an i.i.d shock, normally distributed with standard deviation σ.

Conventions

Note that in this set of equations we follow the dynare conventions:

- no distinction between states and controls:
  - **endogenous** variables k_t, y_t, n_t, i_t, z_t
    - can appear at $t-1$, t, $t+1$
  - **exogenous** variables ϵ_t at date t
- variables have subscript t if they are first known at date t
  - New information arrives with the innovations ϵ_t.
  - i.e. information set is spanned by $\mathcal{F}_t = \mathcal{F}(\cdots, \epsilon_{t-3}, \epsilon_{t-2}, \epsilon_{t-1}, \epsilon_t)$

Warning:

These conventions are different from ones typically used in optimal control. To check that your timing is correct, reason about which variables are *predetermined*. For instance, when producing $y_t = k_{t-1}^{\alpha} n_t^{\alpha}$, the level of capital cannot be adjusted to the productivity innovation to produce in period t hence it appears with date $t-1$.

Abstract Representation

Denote by y_t the vector of endogenous variables. Denote by e_t the vector exogenous variables.

The model can be represented by a function f such that:

$$\forall t \; \mathbf{E}_t \left[f(y_{t-1}, y_t, y_{t+1}, e_t) \right] = 0$$

We look for a *recursive* solution φ in the form $y_t = \varphi(y_{t-1}, e_t)$

Tip

In the RBC model we have:

- endogenous: $y_t = (k_t, q_t, n_t, i_t, z_t)$
- exogenous: $e_t = (\epsilon_t)$
- equations: each equation corresponds to a component of f

Remark: expectations taken on variables at t or $t-1$ can be ignored.
$E_t \left[k_t - (1-(1-\delta)k_{t-1}) - i_t \right]$ is simply $k_t - (1-(1-\delta)k_{t-1}) - i_t$

Steady-state

The *deterministic* steady-state is a value of endogenous variables, which solve the equations with $y_{t-1} = y_t = y_{t-1}$ an in the absence of shock (i.e. $e_t = 0$).

It satisfies

$$f(\overline{y}, \overline{y}, \overline{y}, 0) = 0$$

Tip

For the RBC model, the steady-state can be computed as:

$$\begin{aligned} n &= 0.33 \\ z &= 0 \\ r_k &= 1/\beta - 1 + \delta \\ w &= (1-\alpha) * exp(z) * (k/n)^{\alpha} \\ k &= n/(r_k\, \alpha)^{(1/(1-\alpha))} \\ q &= exp(z) * k^{\alpha} * n^{1-\alpha} \\ i &= \delta * k \\ c &= q - i \end{aligned}$$

which implies $\chi = w/c^{\sigma}/n^{\eta}$

Coding the RBC Model

Let's code it up

```julia
using LabelledArrays
```

First we need to provide **the calibration values and the steady-state** values.

In the case of the RBC, it is easier to do both at once.

```julia
# parameters
#p = (;...)
```

Then we define a function representing the model equations

```julia
# LabelledVectors are useful here:
# - they behave like namedtuples, and like vectors
```

```
f (generic function with 1 method)
```

```julia
function f(v_f, v, v_p, e, p)

end
```

Check that the steady-state conditions are indeed met.

```julia
# if they are not, change initial guess and/or model
```

If the steady-state is not right, but you are sure about the model, you can also look for the steady-state numerically.

```julia
using NLsolve
```

```
Enter cell code...
```

Differentiating the Model

To solve the model, we we replace it by a first order approximation. To do so we replace

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

where the solution is: $y_t = \varphi(y_{t-1}, e_t)$

by

$$\mathbf{E}_t\left[A\hat{y}_{t+1} + B\hat{y}_t + C\hat{y}_{t-1} + De_t\right] = 0$$

where

- variable $\hat{y} = y_t - \overline{y}$ is in deviation to the steady-state
- solution is approximated by $\hat{y} = G_y\hat{y}_{t-1} + G_e e_t$
    - i.e. $y_t \approx \overline{y} + G_y\hat{y}_{t-1} + G_e e_t$

The next steps

We still need to:

- compute:

$$A = f^{\prime}_{y_{t+1}}(\overline{y},\overline{y},\overline{y}, 0)$$

$$B = f^{\prime}_{y_{t}}(\overline{y},\overline{y},\overline{y}, 0)$$

$$C = f^{\prime}_{y_{t-1}}(\overline{y},\overline{y},\overline{y}, 0)$$

$$D = f^{\prime}_{e_{t}}(\overline{y},\overline{y},\overline{y}, 0)$$

- solve the equation in G_y, G_e

```
md"""## The next steps

We still need to:

- compute:
$A = f^{\prime}_{y_{t+1}}(\overline{y},\overline{y},\overline{y}, 0)$
$B = f^{\prime}_{y_{t}}(\overline{y},\overline{y},\overline{y}, 0)$
$C = f^{\prime}_{y_{t-1}}(\overline{y},\overline{y},\overline{y}, 0)$
$D = f^{\prime}_{e_{t}}(\overline{y},\overline{y},\overline{y}, 0)$

- solve the equation in $G_y, G_e$

"""
```