Introduction and Linearization

Solution Methods for Macroeconomic Models

Petr Sedláček

Solution Methods for Macroeconomic Models

- Monday - Tuesday: Solving models with “representative agents”
  - Linearization in theory and practice: Dynare
  - Non-linear solutions methods: value function iteration, projection
  - Analyzing models: parameterization/estimation, simulation/IRFs
- Wednesday - Thursday: Solving models with “heterogeneous agents”
  - Models without aggregate uncertainty: basic algorithm
  - Models with aggregate uncertainty: key issues and alternatives
- Friday: “Final assignment”
  - Solve/estimate model with heterogeneous firms and aggregate uncertainty

Introduction

Overview for today

Introduction into numerical methods

Perturbation

- main idea
- first-order perturbation and certainty equivalence
- implementation in Dynare

DIY linearization

- main idea and algorithm

Overview for today

1. Introduction
2. A DSGE model
3. Perturbation
4. Perturbation in Dynare
5. DIY linearization

Introduction

Why DSGE's?

Why (DSGE) models?

Why not only use tons of data?

- even with super-cool techniques like machine learning?

DSGE models give

- more discipline than reduced-form methods
- discipline comes from “cross-equation” restrictions
  - stochastics of exogenous variables
  - together with forward-looking behavior of agents
  - result in implication for evolution of endogenous variables

Prior to DSGE models...

- long tradition of large macroeconometric models
- these reduced-form systems have certain drawbacks
  - no “GE”
  - no forward-looking behavior
- changes after Kydland and Prescott (1982)
  - other critical contributions by Hansen, Lucas, Sargent and Sims
- a nice discussion of current state of macro (and identification)
  - Jón Steinsson: A New Macroeconomics?

Introduction

What will we cover?

What will we cover?

Computational tools for “Rep-Agent models”

- what do we need to solve for?
  - policy rules (functions)
- why is this a tough problem?
  - forward looking behavior
  - dynamics today depend on expectations of future dynamics
  - focus on recursive problems
  - even then
    - analytical solutions are rare
    - “S” in DSGE necessitates computation of expectations

What will we cover?

1) Tools for solving (rep-agent) DSGE models
   - characterize unknown functions (in several ways)
     - perturbation
     - projection
     - value function iteration

2) Tools for parameterizing DSGE models
   - discuss calibration, estimation, matching moments
   - quick intro into Maximum Likelihood estimation

3) Tools for solving heterogeneous-agent DSGE models
   - builds on the above + algorithm to solve for equilibrium
   - alternative methods for solutions with aggregate uncertainty

Note on "solving" models

Solving models is not the end goal!

- can you use steady state comparisons "only"?
  - many interesting questions don't involve business cycles!
- if you're set on business cycles, how important is non-linearity?
  - solving models linearly is *always* a good idea (at least initially)
  - can you "rephrase" your model to fit your solution strategy?
  - often much easier than sticking to hard-to-solve non-linear model

Overview for today

1. ~~Introduction~~
2. A DSGE model
3. Perturbation
4. Perturbation in Dynare
5. DIY linearization

Neoclassical Growth Model

Neoclassical growth model

- representative household maximizing expected lifetime utility
- household owns production technology
- capital is the only factor of production
- resources spent on consumption and investment into capital
- each period existing capital depreciates at certain rate
- production subject to exogenous fluctuations in productivity

Production

$$y_t = Z_t k_t^{\alpha}$$

$$Z_t = 1 - \rho + \rho Z_{t-1} + \epsilon_t$$

$$\mathbb{E}\epsilon_t = 0$$

$$\mathbb{E}\epsilon_t^2 = \sigma_z^2$$

$$\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$$

$$\text{s.t.} \quad c_t + k_{t+1} = y_t + (1-\delta)k_t$$

$$k_0 \quad \text{given}$$

$$Z_0 \quad \text{given}$$

Neoclassical Growth Model

Solution

Solution

What is the solution?

- a sequence $\{c_t, k_{t+1}\}_{t=0}^{\infty}$
- maximizing the expected discounted sum of per-period utilities

Sounds like a tough problem!

- different $k_0 \rightarrow$ optimal sequences different!
- different realizations of $Z_t \rightarrow$ optimal sequences different!

Solution

Trick is to

- replace sequential problem with 2-period decisions $\rightarrow$ recursiveness
- must make sure that each 2-period decision is globally optimal
  - Principle of Optimality (Bellman)
- decisions depend on state variables
  - agents facing the same state variables make the same decisions
    - independent of the time period they are in
  - realizations of exogenous shocks, pre-determined variables (e.g. capital stock)
  - not always easy to know what the state variables are!

Policy rules

- what are the state variables?
  - beginning-of-period capital and productivity
- what are the policy rules?

$$c_t = c(k_t, Z_t)$$

$$k_{t+1} = k(k_t, Z_t)$$

- how are they determined?

$$u_c(c_t) = \beta \mathbb{E}_t u_c(c_{t+1}) \left(\alpha Z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta \right)$$

$$c_t + k_{t+1} = y_t + (1 - \delta) k_t$$

Neoclassical Growth Model

Use of computational tools

Where do we use our computational tools?

- analytical solutions rarely exist
- $\rightarrow$ need to approximate the policy functions (perturbation or projection)

$$c_t \approx \widetilde{c}(k_t, Z_t; \psi_c)$$

$$k_{t+1} \approx \widetilde{k}(k_t, Z_t; \psi_k)$$

- what are we solving for?
  - the coefficients of the approximations: ψ_c and ψ_k
  - requires specifying a domain (needs to be bounded)
  - $\rightarrow$ consider "stationarized" models (no growth)

Alternatively, specify the recursive problem using the Bellman equation

$$V(k_{-1}, Z) = \max_{c,k} u(c) + \mathbb{E}\beta V(k, Z_{+1}) \quad \text{s.t.}$$

$$c + k = Zk_{-1}^{\alpha} + (1 - \delta)k_{-1}$$

$$Z_{+1} = 1 - \rho + \rho Z + \epsilon$$

- approximate the value function (Bellman eq. evaluated at optimal choices)
- $\rightarrow$ value function iteration
- what are we solving for?
  - maximized values of $V(.,.)$ at different points of the state-space
  - requires specifying a domain (needs to be bounded)
  - $\rightarrow$ consider "stationarized" models (no growth)

Special case of analytical solution

- assume log utility ($\gamma = 1$)
- and full depreciation ($\delta = 1$)
- this is the Brock-Mirman model

Turns out that this version has an analytical solution:

$$k_{t+1} = \alpha\beta Z_t k_t^{\alpha}$$

$$c_t = (1 - \alpha\beta) Z_t k_t^{\alpha}$$

Neoclassical Growth Model

Taking stock

Neoclassical growth model

- workhorse DSGE model which we'll encounter throughout the course
- solution consists of policy functions
- computational tools necessary to approximate such policy functions

Overview for today

1. ~~Introduction~~
2. ~~A DSGE model~~
3. Perturbation
4. Perturbation in Dynare
5. DIY linearization

Perturbation

Perturbation: basic idea

- Perturbation is a way to approximate a function
  - more generally, it is a way of taking derivatives
  - as such it has broad applications
- it uses Taylor's theorem
- it also uses the Implicit function theorem

Perturbation

Theoretical Underpinning

Theorem *Let $k \geq 1$ be an integer and let function $f : \mathbb{R} \to \mathbb{R}$ be k times differentiable at point $a \in \mathbb{R}$. Then there exists a function $h_k : \mathbb{R} \to \mathbb{R}$ such that*

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(k)}(a)}{k!}(x-a)^k + h_k(x)(x-a)^k,$$

and $\lim_{x \to a} h_k(x) = 0$.

Theorem *Let $f : \mathbb{R}^{n+m} \to \mathbb{R}^m$ be a continuously differentiable function and let $\mathbb{R}^{n+m}$ have coordinates (x, y). Fix a point $(\overline{x}, \overline{y})$ with $f(\overline{x}, \overline{y}) = 0$. If the Jacobian matrix $\mathcal{J}_{f,y}(\overline{x}, \overline{y})$ is invertible, then there exists an open set U of $\mathbb{R}^n$ containing $\overline{x}$ such that there exists a unique continuously differentiable function $g : U \to \mathbb{R}^m$ such that*

$$g(\overline{x}) = \overline{y}$$

and

$$f(x, g(x)) = 0 \quad \text{for all } x \in U.$$

Moreover, the partial derivatives of g in U are given by the matrix product

$$\frac{\partial g}{\partial x_j}(x) = -[\mathcal{J}_{f,y}(x, g(x))]^{-1} \left[\frac{\partial f}{\partial x_j}(x, g(x)) \right]$$

Perturbation

Details

Back to the neoclassical model

- the above is all very nice
- but at this point a bit abstract
- lets see if we can write the neoclassical growth model
- in a way that looks like the notation we just used...

$$c_t^{-\gamma} = \beta \mathbb{E}_t c_{t+1}^{-\gamma} \alpha Z_{t+1} k_{t+1}^{\alpha-1}$$

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

$$Z_t = (1-\rho) + \rho Z_{t-1} + \sigma \epsilon_t$$

- σ controls the degree of uncertainty

What are we after?

- rewrite the above equations as

$$\mathbb{E}_t F[c_{t+1}, c_t, k_{t+1}, Z_{t+1}, k_t, Z_t] = 0$$

- what are the states (x) and "policy" variables ($g(x)$)?

$$x_t = [k_t, Z_t]$$

$$x_{t+1} = h(x_t, \sigma) + \sigma\widetilde{\epsilon}_{t+1}$$

$$c_t = g(x_t, \sigma)$$

  - notice that uncertainty (σ) explicitly enters the policy function!

Rewrite the system

$$\mathbb{E}_t F\Big(g(h(x_t,\sigma)+\sigma\widetilde{\epsilon}_{t+1},\sigma), g(x_t,\sigma), h(x_t,\sigma)+\sigma\widetilde{\epsilon}_{t+1}, x_t\Big) = 0$$

Perturbation

1st order perturbation and certainty equivalence

Perturbing the system

- perturbation methods find a local approximation of g and h
- it is local around a certain point $(\overline{x}, \overline{\sigma})$
- in particular, a Taylor approximation around $(\overline{x}, \overline{\sigma})$ gives

$$\begin{aligned} g(x,\sigma) \approx & g(\overline{x},\overline{\sigma}) + g_x(\overline{x},\overline{\sigma})(x-\overline{x}) + g_\sigma(\overline{x},\overline{\sigma})(\sigma-\overline{\sigma}) \\ & + 1/2[g_{xx}(\overline{x},\overline{\sigma})(x-\overline{x})^2 + 2g_{x\sigma}(\overline{x},\overline{\sigma})(x-\overline{x})(\sigma-\overline{\sigma}) \\ & + g_{\sigma\sigma}(\overline{x},\overline{\sigma})(\sigma-\overline{\sigma})^2] + \cdots \end{aligned}$$

$$\begin{aligned} h(x,\sigma) \approx & h(\overline{x},\overline{\sigma}) + h_x(\overline{x},\overline{\sigma})(x-\overline{x}) + h_\sigma(\overline{x},\overline{\sigma})(\sigma-\overline{\sigma}) \\ & + 1/2[h_{xx}(\overline{x},\overline{\sigma})(x-\overline{x})^2 + 2h_{x\sigma}(\overline{x},\overline{\sigma})(x-\overline{x})(\sigma-\overline{\sigma}) \\ & + h_{\sigma\sigma}(\overline{x},\overline{\sigma})(\sigma-\overline{\sigma})^2] + \cdots \end{aligned}$$

What are we solving for?

- we approximate the policy functions with a polynomial
- the unknown coefficients are the n-order derivatives at $(\overline{x}, \overline{\sigma})$
- how do we solve for them?
- recall that $F[x_t, \sigma] = 0$ for any value of x and σ
- $\rightarrow$ derivatives (of any order) of F also 0!

$$F_{x^k,\sigma^j}[x_t, \sigma] = 0 \qquad \forall x, \sigma, j, k$$

Where are we approximating?

- particularly convenient point is the non-stochastic steady state
  - i.e. $\sigma = 0$ and $x_t = \bar{x}$
  - $\bar{c} = g(\bar{x}, 0)$ and $\bar{x} = h(\bar{x}, 0)$
- why is so convenient?
- in principle you can approximate around any point

Getting the policy function derivatives

- under 1st order perturbation we have

$$g(x,\sigma) \approx g(\bar{x},0) + g_x(\bar{x},0)(x-\bar{x}) + g_\sigma(\bar{x},0)\sigma$$

$$h(x,\sigma) \approx h(\bar{x},0) + h_x(\bar{x},0)(x-\bar{x}) + h_\sigma(\bar{x},0)\sigma$$

- we also know that

$$g(\bar{x},0) = \bar{c}$$

$$h(\bar{x},0) = \bar{x}$$

- solve for the derivatives (coefficients of approximating Taylor polynomial)

$$F_{x^k,\sigma^j}[x_t,\sigma] = 0 \qquad \forall x,\sigma,j,k$$

For simplicity, substitute out consumption to get $F[x_{t+2}, x_{t+1}, x_t] = 0$

$$F_x = \frac{\partial F}{\partial x_{t+2}} \frac{\partial x_{t+2}}{\partial x_{t+1}} \frac{\partial x_{t+1}}{\partial x_t} + \frac{\partial F}{\partial x_{t+1}} \frac{\partial x_{t+1}}{\partial x_t} + \frac{\partial F}{\partial x_t}$$

$$= \overline{F}_1 \frac{\partial x_{t+2}}{\partial x_{t+1}} \frac{\partial x_{t+1}}{\partial x_t} + \overline{F}_2 \frac{\partial x_{t+1}}{\partial x_t} + \overline{F}_3$$

$$= \overline{F}_1 h_x^2 + \overline{F}_2 h_x + \overline{F}_3 = 0$$

- $\frac{\partial F(x_{t+2}, x_{t+1}, x_t, \sigma)}{\partial x_{t+i}} \Big|_{x_{t+2} = x_{t+1} = x_t = \overline{x}, \sigma = 0} = \overline{F}_{3-i}$
- $\frac{\partial h(x_t, \sigma)}{\partial x_t} \Big|_{x_t = \overline{x}, \sigma = 0} \ \forall t = h_x$

Perturbation

Uncertainty

Back to 1st order case

$$h(x, \sigma) = h(\overline{x}, 0) + h_x(\overline{x}, 0)(x - \overline{x}) + h_\sigma(\overline{x}, 0)(\sigma - \overline{\sigma})$$

- we can find h_x from a 2nd order system
- further higher-order terms can be solved from linear systems
- but what about h_σ?

Getting 1st-order derivative w.r.t. σ

$$\mathbb{E}_t F\Big(h(h(x_t,\sigma)+\sigma\widetilde{\epsilon}_{t+1},\sigma)+\sigma\widetilde{\epsilon}_{t+2}, h(x_t,\sigma)+\sigma\widetilde{\epsilon}_{t+1}, x_t\Big) =$$
$$= \mathbb{E}_t F\big(x'', x', x\big) = 0$$

$$\mathbb{E}_t F_\sigma(x'', x', x, \sigma)|_{x=\bar{x},\sigma=0} =$$

$$= \mathbb{E}_t\left[F_{x''}[h_\sigma + h_x(\widetilde{\epsilon}_{t+1} + h_\sigma) + \widetilde{\epsilon}_{t+2}] + F_{x'}(h_\sigma + \widetilde{\epsilon}_{t+1})\right]$$

$$= F_{x''}h_\sigma(1+h_x) + F_{x'}h_\sigma = 0$$

Certainty equivalence

Certainty equivalence result

- the variance of shocks does not matter for policy rules
- important limitation of 1st order approximation
  - what economic questions cannot be studied in this case?
- what about higher order approximations?

Getting 2-order derivative w.r.t. σ

- only $g_{\sigma\sigma}$ and $h_{\sigma\sigma}$ matter for policy function
- this affects the constant in the policy rule
- can still have important implications
  - certain economic questions can be addressed
  - can have indirect effect on dynamics (how?)
- need 3rd order to capture effect of uncertainty on “slopes”

Perturbation

Accuracy

Local approximation?

- perturbation is also known as local approximation
- when does the question of accuracy arise?
- what could go wrong?

Accuracy of perturbation

The theory guarantees local convergence

- global convergence *could* be good, but depends on approximated function
- e.g. if true function is a polynomial $\rightarrow$ approximations converge to truth

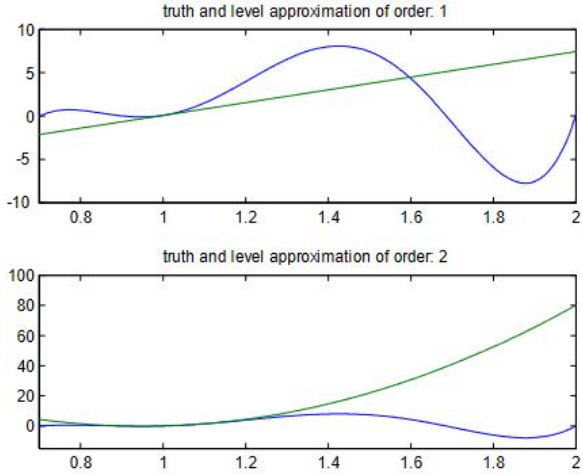
Theory doesn't say anything about convergence properties

- e.g. not clear whether 2nd order is better than 1st
- nonlinear higher-order polynomials always have "weird" shapes
- this can occur close or far away from the steady state!

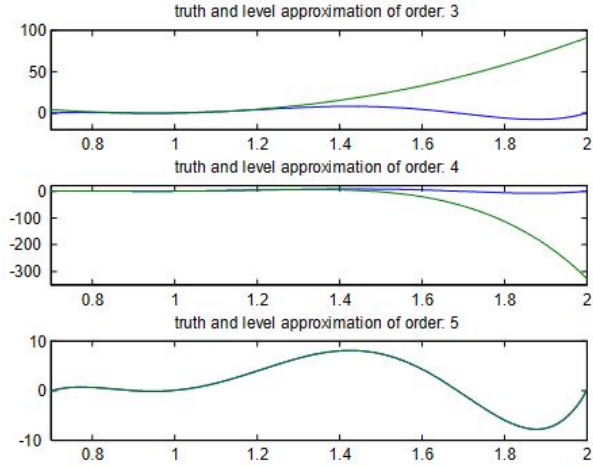
Wouter's example: Consider the true function to be defined on $x \in [0.7, 2]$ s.t.

$$f(x) = -690.59 + 3202.4x - 5739.45x^2 + 4954.2x^3 - 2053.6x^4 + 327.1x^5$$

Wouter's example: all kinds of wild things can happen



Wouter's example: all kinds of wild things can happen



a bit like this...

Perturbation

Taking stock

Taking stock

Perturbation:

- (in our context) means of approximating policy rules
- relies on Taylor polynomial and Implicit function theorem

Pros:

- easy to implement (you'll see)
- can handle large state-space (heterogeneity)

Cons:

- can't handle certain features (non-differentiabilities)
- "local" solution method

Overview for today

1. ~~Introduction~~
2. ~~A DSGE model~~
3. ~~Perturbation~~
4. Perturbation in Dynare
5. DIY linearization

Dynare

What is Dynare and why use it?

- (free) software for perturbation solutions and more
  - also estimation: ML, Bayesian
  - many options
- you MUST know what it is doing
- once you do, its a very useful tool

Where/how to get Dynare

- download at `www.dynare.org`
- install *and*
- in Matlab set path to `.../Dynare/Matlab`
- read the documentation

What does Dynare do?

Dynare implements a perturbation solution to your model

- model described by $\mathbb{E}_t[F(y_t, x_t)] = \mathbb{E}_t[F(g(x_t), x_t)] = 0$
  - where state variables are denoted by $x_t = [x_{1,t}, ..., x_{n,t}]$
  - and choice variables are denoted by $y_t = g(x_t)$
    - $g(.)$ denote policy rules
- Dynare approximates policy rules, $g(.)$
  - that satisfy first order conditions $\mathbb{E}_t[F(g(x_t), x_t)] = 0$

Result of approximation (e.g. 1st order perturbation)

$$y_t = g(x_t) \approx \bar{y} + (x_t - \bar{x})'a$$

- "bars" indicate steady states
- a coefficient of approximating (Taylor) polynomial

Dynare

Notation

Dynare's main file

- main file type is a `*.mod` file
- into this file you specify
  - variables of your model
  - parameters and their values
  - model equations (linearized or not)
  - initial values (ideally steady state)
  - solution method (1st or higher order)
    - many other options (IRFs, simulations, moments etc.)
    - you can also estimate models

Notation in Dynare

- variables known at the beginning of period
- are dated as $t-1$!
  - k_t: capital *choice* in period t
  - k_{t-1}: capital stock available in t

Policy rules

- Dynare produces perturbation approximation to policy rules
- for now consider linear approximations
- linear in what?!
  - Dynare doesn't know that "k" means capital
  - k could be
    - level of capital
    - log of capital
- its up to you to decide
- Dynare will produce policy rules for specified variables

Policy rules

- in neoclassical growth model
- Dynare generates following policy rules

$$k_t = \overline{k} + a_{kk}(k_{t-1} - \overline{k}) + a_{kz}(z_{t-1} - \overline{z}) + a_{k\epsilon}\epsilon_t$$

- i.e. it splits structural shocks into
  - past value and
  - innovation
  - i.e. if $z_t = 1 - \rho + \rho z_{t-1} + \epsilon_t$ then $a_{kz} = \rho a_{k\epsilon}$

Dynare

Blocks

Dynare blocks

A Dynare file has several blocks:

1. list of variables
2. list of exogenous shocks
3. list of model parameters and their values
4. model block (optimality conditions)
5. shock properties
6. initial values
7. solution (and other) commands

Definitions and parametrization

1. Specify variables
   - specified by typing “var” and then listing variables
2. Specify exogenous shocks
   - specified by typing “varexo” and then listing shocks
3. Specify parameters and their values
   - specified by typing “parameters” and then listing parameters
   - each parameter must then be assigned a value
     - either directly in Dynare file
     - or by loading it from outside Dynare file
     - the latter is more convenient for calibration

4. Model block contains equilibrium conditions

- initialize block by typing “model;”
- end it by typing “end;”
- in between simply write your model equations

Specifics

- Dynare figures out there are expectations when you write $(+1)$
- e.g. the Euler equation:

```
c^(-gamma)=beta*c(+1)^(-gamma)*(alpha*Z(+1)k^(alpha-1)+1-delta)
```

5. Shock properties
   - initialize the block by typing “shocks;”
   - end it by typing “end;”
   - in between specify shock properties
     - e.g. “var e; stderr sigZ;”
     - can specify more, like correlations etc.

6. Initial values
   - initialize block by typing “initval;”
   - end it by typing “end;”
   - inbetween list the initial values of all variables
     - ideally give Dynare the steady state
     - often difficult to compute, so supply it yourself

7. Give Dynare the green light to solve the model
   - "`stoch_simul(options)`"
   - options include
     - order of perturbation: e.g. "order=1" for linear
     - length of IRFs: e.g. IRF=20
     - many, many more

To actually run Dynare type `dynare filename.mod`

Other useful features

- "resid" command shows equation errors
  - it plugs initial values into model equations
  - they should all be zero in steady state
  - useful for finding out typos

Dynare

Example code

RBC model in Dynare

```
// neoclassical growth model solution and simulation

var c, k, y, z;
varexo e;
parameters alpha, beta, delta, nu, rhoz, sigz, kss, css;

load params;

set_param_value('alpha'     ,par.alpha);   // returns to scale parameter
set_param_value('beta'      ,par.beta);    // discount factor
set_param_value('delta'     ,par.delta);   // depreciation rate
set_param_value('nu'        ,par.nu);      // relative risk aversion coefficient
set_param_value('rhoz'      ,par.rhoz);    // autocorrelation of productivity shock
set_param_value('sigz'      ,par.sigz);    // standard deviation of productivity shock

set_param_value('kss'       ,par.k);       // steady state capital
set_param_value('css'       ,par.c);       // steady state consumption

model;
c^(-nu) = beta*c(+1)^(-nu)*(alpha*z(+1)*k^(alpha-1) + 1 - delta);
c + k   = y + k(-1)*(1-delta);
y       = z*k(-1)^alpha;
z       = 1 - rhoz + rhoz*z(-1) + e;
end;
```

RBC model in Dynare

```
initval;
k   = kss;
c   = css;
y   = kss^alpha;
z   = 1;
end;

shocks;
var e; stderr sigz;
end;

resid;
steady;

stoch_simul(order=1,nomoments, irf=0, periods = 5000);
```

Dynare's output (coefficients of policy rules)

- remember the quirks of Dynare!

POLICY AND TRANSITION FUNCTIONS

	c	k	y	z
Constant	2.754327	37.989254	3.704059	1.000000
k(-1)	0.044825	0.965276	0.035101	0
z(-1)	0.798702	2.720154	3.518856	0.950000
e	0.840739	2.863320	3.704059	1.000000

Now let's increase the size of shocks (from $\sigma = 0.01$ to $\sigma = 0.1$)

- what happens to the solution?

POLICY AND TRANSITION FUNCTIONS

	c	k	y	z
Constant	2.754327	37.989254	3.704059	1.000000
k(-1)	0.044825	0.965276	0.035101	0
z(-1)	0.798702	2.720154	3.518856	0.950000
e	0.840739	2.863320	3.704059	1.000000

What about a log-linear approximation?

```
var c, k, y, z;
varexo e;
parameters alpha, beta, delta, nu, rhoz, sigz, kss, css;

load params;

set_param_value('alpha'     ,par.alpha);   // returns to scale parameter
set_param_value('beta'      ,par.beta);    // discount factor
set_param_value('delta'     ,par.delta);   // depreciation rate
set_param_value('nu'        ,par.nu);      // relative risk aversion coefficient
set_param_value('rhoz'      ,par.rhoz);    // autocorrelation of productivity shock
set_param_value('sigz'      ,par.sigz);    // standard deviation of productivity shock

set_param_value('kss'       ,par.k);       // steady state capital
set_param_value('css'       ,par.c);       // steady state consumption

model;
c^(-nu) = beta*c(+1)^(-nu)*(alpha*z(+1)*k^(alpha-1) + 1 - delta);
c + k   = y + k(-1)*(1-delta);
y       = z*k(-1)^alpha;
z       = 1 - rhoz + rhoz*z(-1) + e;
end;
```

What about a log-linear approximation?

```
var c, k, y, z;
varexo e;
parameters alpha, beta, delta, nu, rhoz, sigz, kss, css;

load params;

set_param_value('alpha'     ,par.alpha);   // returns to scale parameter
set_param_value('beta'      ,par.beta);    // discount factor
set_param_value('delta'     ,par.delta);   // depreciation rate
set_param_value('nu'        ,par.nu);      // relative risk aversion coefficient
set_param_value('rhoz'      ,par.rhoz);    // autocorrelation of productivity shock
set_param_value('sigz'      ,par.sigz);    // standard deviation of productivity shock

set_param_value('kss'       ,par.k);       // steady state capital
set_param_value('css'       ,par.c);       // steady state consumption

model;
exp(c)^(-nu) = beta* exp(c(+1))^(-nu)*(alpha* exp(z(+1))* exp(k)^(alpha-1) + 1 - delta);
exp(c) + exp(k)   = exp(y) + exp(k(-1))*(1-delta);
exp(y)       = exp(z)* exp(k(-1))^alpha;
exp(z)       = 1 - rhoz + rhoz* exp(z(-1)) + e;
end;
```

What about a log-linear approximation?

Dynare's output (coefficients of policy rules): linear

POLICY AND TRANSITION FUNCTIONS

	c	k	y	z
Constant	2.754327	37.989254	3.704059	1.000000
k(-1)	0.044825	0.965276	0.035101	0
z(-1)	0.798702	2.720154	3.518856	0.950000
e	0.840739	2.863320	3.704059	1.000000

Dynare's output (coefficients of policy rules): log-linear

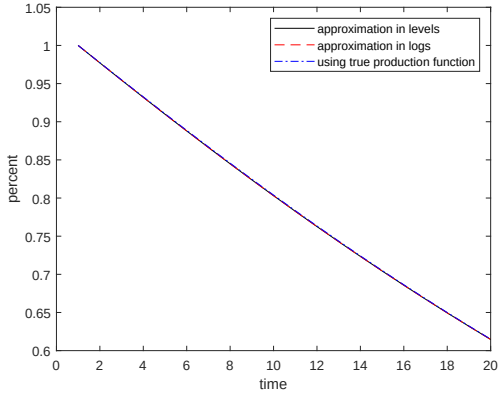
POLICY AND TRANSITION FUNCTIONS

	c	k	y	z
Constant	1.013173	3.637303	1.309429	0
k(-1)	0.618247	0.965276	0.360000	0
z(-1)	0.289981	0.071603	0.950000	0.950000
e	0.305243	0.075372	1.000000	1.000000

What about a log-linear approximation?

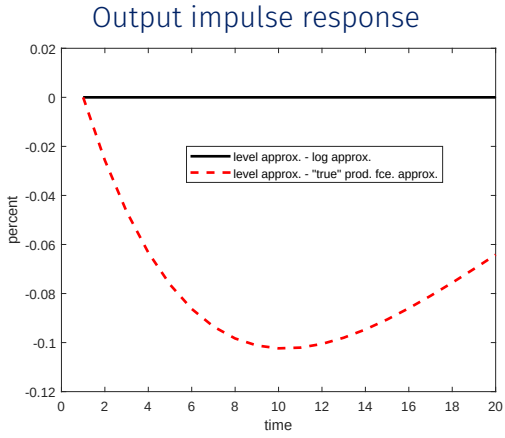
Does it matter for the dynamics?

Output impulse response



What about a log-linear approximation?

Does it matter for the dynamics?



Dynare

Tips and tricks

Incorporating Dynare into a broader code

Often very useful to have the `.mod` file as a part of bigger code

- e.g. when calibrating a model, conducting your own simulation, IRFs etc
- to make this efficient, it requires a few tricks

Tips/tricks

- keeping variables in memory
- loading, instead of setting, parameter values
- saving solution in a separate file
- the idea of homotopy

Keeping variables in memory

As a default, Dynare clears all variables from memory

- to over-ride this, include `noclearall` after your Dynare command
- e.g. `dynare neoclassModel.mod noclearall`

Setting parameter values

In the "parameter block" of the `.mod` file, you need to specify all parameter values

- either you set them directly, e.g. `beta=0.99`
- or you can load parameter values

Loading parameter values

- In a "standard" Matlab program, you can set all your parameter values

```
%% 1. Parametrization
%================================================================================

par.beta    = 0.99;         % discount factor
par.alpha   = 0.36;         % returns to scale in production
par.delta   = 0.025;        % depreciation rate
par.rhoz    = 0.95;         % autocorrelation of productivity shock
par.sigz    = 0.01;         % standard deviation of productivity shock
par.nu      = 1;            % relative risk aversion coefficient (1=log utility)
```

Setting parameter values

In the "parameter block" of the `.mod` file, you need to specify all parameter values

- either you set them directly, e.g. `beta=0.99`
- or you can load parameter values

Loading parameter values

- in a "standard" Matlab program, you can set all your parameter values
- then, save all the parameter values as e.g. `save params par`
- in your `.mod` file, load those parameters as `load params`
- finally, set parameters to loaded values using `set_param_value('alpha', par.alpha);`

```
set_param_value('alpha'     ,par.alpha);   // returns to scale parameter
set_param_value('beta'      ,par.beta);    // discount factor
set_param_value('delta'     ,par.delta);   // depreciation rate
set_param_value('nu'        ,par.nu);      // relative risk aversion coefficient
set_param_value('rhoz'      ,par.rhoz);    // autocorrelation of productivity shock
set_param_value('sigz'      ,par.sigz);    // standard deviation of productivity shock
```

Wouter's dynarerocks file

All Dynare output is saved in `oo_`

- e.g. IRFs of capital to a productivity shock are in `oo_.irfs.k_e`
- decision rule coefficients are in `oo_.dr.ghx`, in a particular order

Wouter's `disp_dr.m` function

- includes command that saves decision rules in format you see on screen

POLICY AND TRANSITION FUNCTIONS

	c	k	y	z
Constant	2.754327	37.989254	3.704059	1.000000
k(-1)	0.044825	0.965276	0.035101	0
z(-1)	0.798702	2.720154	3.518856	0.950000
e	0.840739	2.863320	3.704059	1.000000

- unfortunately, specific to Dynare versions and Wouter got tired of updating

WOUTER'S `dynarerocks` FILE

All Dynare output is saved in `oo_`

- e.g. IRFs of capital to a productivity shock are in `oo_.irfs.k_e`
- decision rule coefficients are in `oo_.dr.ghx`, in a particular order

Wouter's `disp_dr.m` function

- includes command that saves decision rules in format you see on screen
- matrix is conveniently called `dynarerocks.mat`
- i.e. in order to load decision rules, simply type `load dynarerocks`

All the above is super-useful when calibrating a model

- often, you can solve a model under “some” parametrization
- but getting to your preferred parametrization is harder
  - you might not know what it is
  - you might not have good initial values (steady state)
- in both of the above cases, it is useful to use the homotopy idea
  - move slowly from what you know to where you want to be

Example: suppose you want to solve $[F(x; \alpha_1)] = 0$

- and suppose you know the solution to $[F(x; \alpha_0)] = 0$
- using solution of $[F(x; \alpha_0)] = 0$ as initial guess in $[F(x; \alpha_1)] = 0$ may not work!

Instead, solve a sequence of "intermediate" cases $[F(x; \omega\alpha_0 + (1 - \omega)\alpha_1)] = 0$

- where $\omega \in [0, 1]$
- allows transition between what you know (α_0) to where you're heading (α_1)
  1. solve model for $\omega_0 = 1$, save x
  2. use x from 1 as initial conditions for case where $\omega_1 < \omega_0$ and save x again
  3. repeat 2 until $\omega_J = 0$

Dynare

Taking stock

Dynare

- incredibly useful software for perturbation solutions of DSGE models
- can solve, estimate (ML, Bayesian), simulate, produce IRFs etc.
- read documentation for specific syntax

Overview for today

1. ~~Introduction~~
2. ~~A DSGE model~~
3. ~~Perturbation~~
4. ~~Perturbation in Dynare~~
5. DIY linearization

DIY Linearization

DIY linearization: Starting point

Linearization of models is great

- fast and can deal with large state-spaces
- models can be estimated “easily”
- great starting point to see if model is reasonable

But, linearization also has important drawbacks

- accuracy only guaranteed around approximation point
- certainty equivalence!
- can’t handle some features, e.g. *occasionally* binding constraints

DIY linearization: Purpose

In its basic form:

- opens up “blackbox” of Dynare
- easy to implement and fast

Allows for important extensions

- linearization around an arbitrary point
- solving of regime-switching models

Developed by Pontus, see Rendahl (2017), “Linear Time Iteration”

DIY Linearization

General Formulation

General formulation of DSGE models

As before, we can write a DSGE model in the following form

$$\mathbb{E}_t[F(x_{t-1}, x_t, x_{t+1})] = 0$$

- $F[.]$: system of equilibrium conditions
- x: vector of endogenous and exogenous (possibly stochastic) variables
- x^*: corresponding steady state values

1st order perturbation solution based on Taylor expansion around x^*:

$$F(x^*, x^*, x^*) + J_{x_{t-1}}(x_{t-1} - x^*) + J_{x_t}(x_t - x^*) + J_{x_{t+1}}\mathbb{E}_t(x_{t+1} - x^*) = 0$$

$$\underbrace{J_{x_{t-1}}}_{A}\underbrace{(x_{t-1} - x^*)}_{u_{t-1}} + \underbrace{J_{x_t}}_{B}\underbrace{(x_t - x^*)}_{u_t} + \underbrace{J_{x_{t+1}}}_{C}\underbrace{\mathbb{E}_t(x_{t+1} - x^*)}_{u_{t+1}} = 0$$

General formulation of DSGE models

$$Au_{t-1} + Bu_t + Cu_{t+1} = 0$$

- arbitrarily general (e.g. many states)
- easy to solve
- can quickly check uniqueness/stability (Blanchard/Khan conditions)

So, how to solve the above (without shocks first)?

- we know we're looking for a linear solution, i.e. $u_t = Fu_{t-1}$
  - u_{t-1}: state variables, u_t: choice variables, F: same dimension as J's
- start with a guess, F_0, plug it into the above and iterate

DIY Linearization

Solution

$$Au_{t-1} + Bu_t + Cu_{t+1} = 0$$

Main idea of solution:

- guess how you will act tomorrow ($u_{t+1} = F_0 u_t$)
- and use the above system to solve for how you act now

$$Au_{t-1} + Bu_t + CF_0 u_t = 0$$

- we now have a new relationship between u_t and u_{t-1}
- use this as the new guess for F, i.e. F_1:

$$\begin{aligned} Au_{t-1} + (B + CF_0)u_t &= 0 \\ (B + CF_0)u_t &= -Au_{t-1} \\ u_t &= (B + CF_0)^{-1}(-A)u_{t-1} \end{aligned}$$

Solution algorithm: Basics

Ultimately, all we need to do is iterate on

$$F_{n+1} = (B + CF_n)^{-1}(-A)$$

- n indicates iteration number
- continue until "convergence", e.g.

$$||BF_n + CF_n^2 + A|| \approx 0$$

- since the algorithm is very fast, can use tight criteria, e.g. $1e(-12)$

Consider allowing for stochastic shocks:

$$Au_{t-1} + Bu_t + C\mathbb{E}_t u_{t+1} + \epsilon_t = 0$$

- notice, we need to keep linear structure!
- therefore, we are looking for $u_t = Fu_{t-1} + Q\epsilon_t$
- otherwise all the same as before

$$Au_{t-1} + Bu_t + C\mathbb{E}_t[Fu_t + Q\epsilon_{t+1}] + \epsilon_t = 0$$

$$Au_{t-1} + Bu_t + C \quad Fu_t \qquad\qquad + \epsilon_t = 0$$

Solution algorithm: Introducing shocks

Now we have a solution given by

$$u_t = \underbrace{(B + CF)^{-1}(-A)}_{F} u_{t-1} + \underbrace{(B + CF)^{-1}}_{-Q}(-\epsilon_t)$$

- iteration for F is unchanged!

$$F_{n+1} = (B + CF_n)^{-1}(-A)$$

- once converged to an F, can solve for Q:

$$Q = -(B + CF_n)^{-1}$$

- how come we don't need to know Q for F?

DIY Linearization

Solution around an arbitrary point

Solution around an arbitrary point

What about solving the model around $\bar{x} \neq x^*$?

- key difference, introduce an "intercept", $F(\bar{x}, \bar{x}, \bar{x}) = D$
- note that now Jacobians also evaluated at $\bar{x}$, not x^* and $u_t = x_t - \bar{x}$!
- we're now looking for the following solution: $u_t = E + Fu_{t-1}$
- otherwise, all is the same as before

$$D + Au_{t-1} + Bu_t + C(E + Fu_t) = 0$$

$$u_t = \underbrace{(B + CF)^{-1}(-D - CE)}_{E} + \underbrace{(B + CF)^{-1}(-A)}_{F} u_{t-1}$$

- once again, iteration for F unchanged
- after solving for F, can compute $E = (B + C + CF)^{-1}(-D)$

DIY Linearization

Taking stock

Taking stock

- using an iterative scheme instead of solving non-linear equations
- DIY linearization simple to implement and more flexible

Overview for today

1. ~~Introduction~~
2. ~~A DSGE model~~
3. ~~Perturbation~~
4. ~~Perturbation in Dynare~~
5. ~~DIY linearization~~


That's all Folks!