Global DSGE Models

Dan Cao (Georgetown University)
Wenlan Luo (Tsinghua University)
Guangyu Nie (Shanghai University of Finance and Economics)

PHBS, July 7, 2020

Introduction

Q1: Why Global Solution?

- Dynare and other local perturbation methods provide solution around the deterministic steady state

Q1: Why Global Solution?

- Dynare and other local perturbation methods provide solution around the deterministic steady state
- Recent studies highlight the importance of nonlinearity in DSGE models:
  - financial crises in closed or open economies
  - implications of rare diasters (such as COVID-19)
  - portfolio choices models with many financial assets
  - occasionally binding constraints (borrowing constraints, ZLB etc.)
  - international finance models with portfolio choices/capital accumulation

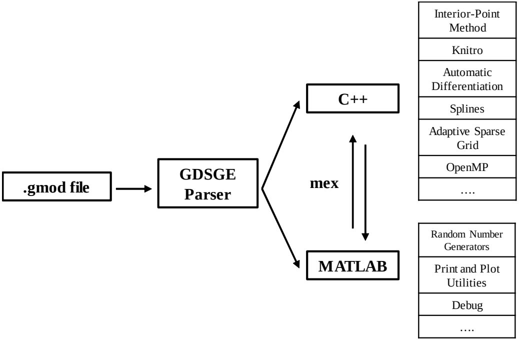
Q1: Why Global Solution?

- Dynare and other local perturbation methods provide solution around the deterministic steady state
- Recent studies highlight the importance of nonlinearity in DSGE models:
  - financial crises in closed or open economies
  - implications of rare diasters (such as COVID-19)
  - portfolio choices models with many financial assets
  - occasionally binding constraints (borrowing constraints, ZLB etc.)
  - international finance models with portfolio choices/capital accumulation
- Models calling for global solution:
  1. models intrinsically not suited for local method
  2. models with large shocks/high nonlinearity
  3. equilibrium properties in different regions are significantly different
  4. when precautionary behavior matters

Q2: Why GDSGE?

- GDSGE (available at www.gdsge.com): Dynare-like toolbox for global non-linear solutions of DSGE models
- Properties of GDSGE:
  1. easy to use: One only needs to provide model specification in a simple way.
  2. unified framework: Encompasses many well-known incomplete markets models with highly nonlinear dynamics
  3. high efficiency and accuracy: More efficient and accurate than the original solution methods of many important papers.
     Most of the examples on our website can be solved in one minute
  4. great flexibility: many options of model specification for users/can be incorporated into the whole program

Q3: What do We Provide?



1. a unified framework that allow users to describe models in simple and intuitive script files
2. efficient implementation that compiles these script files to C++ libraries parallelization, equation solvers with automatic differentiation, and various dense/sparse grid function approximation methods
3. an easy-to-use interface in MATLAB to run/debug/plot/print

Literature

- **Properties and solutions of global DSGE.** Coleman (1990); Duffie et al. (1994); Magill and Quinzii (1994); Cao (2020) among others in the GE incomplete markets literature
  New: A policy iteration method that delivers both good theoretical properties and robust numerical properties
- **Computational Toolbox.** Winschel and Kratzig (2010)... Many others by providing modularized code
  New: A unified framework to represent models in concise scripts. A parser to convert model scripts. No requirements for specific programming languages besides MATLAB
- **Dealing with endogenous state variables with implicit laws of motion. (e.g. wealth share)** Kubler and Schmedders (03), Dumas and Lyasoff (12), Elenev et al. (16)
  New: Introducing consistency equations: enabling a robust algorithm

Roadmap

Please download lecture material at http://www.gdsge.com/lectures.html

- Getting Started - A Simple RBC Model
  - Equilibrium concepts
  - Structure of gmod. file and toolbox usage
  - An extension with irreversible investment
- General GDSGE framework
- Bianchi (2011): Sudden Stops in Open Economies
  - Observe nonlinearity!
  - Initiate policy functions that involve solving non-trivial equations
  - Deal with endogenous borrowing constraint
  - Using adaptive-grid functional approximations
- Kiyotaki and Moore (1997): Collateral Constraints with Investment
  - Consistency equations for endogenous states with implicit laws of motion
  - Generalized Impulse Response Functions
- Some advice on developing models using GDSGE

Getting Started: A Simple RBC Model

Getting Started: A Simple RBC Model

- Preferences

$$\mathbb{E}\sum_{t=0}^{\infty}\beta^t\frac{c_t^{1-\sigma}}{1-\sigma}, \quad L_t = 1$$

- Technology

$$\text{Production: } Y_t = z_t K_t^{\alpha} L_t^{1-\alpha}$$
$$\text{Investment: } K_{t+1} = (1-\delta)K_t + I_t$$

- Markets clear

$$c_t + K_{t+1} = Y_t + (1-\delta)K_t$$

- Shock: $z_t \in \{z_L, z_H\}$, with Markov transition matrix $\mathrm{Pr}_{z\to z'} = \begin{pmatrix} \pi_{LL} & 1-\pi_{LL} \\ 1-\pi_{HH} & \pi_{HH} \end{pmatrix}$

Solution Concepts and Equilibrium Conditions

- Given K_0, a sequential competitive equilibrium is stochastic sequences: $\{c_t, K_{t+1}\}_{t=0}^{\infty}$ such that

$$\text{Euler equation: } c_t^{-\sigma} = \beta \mathbb{E}_t \left[\left(\alpha z_{t+1} K_{t+1}^{\alpha-1} + (1-\delta) \right) c_{t+1}^{-\sigma} \right],$$
$$\text{Budget: } c_t + K_{t+1} = z_t K_t^{\alpha} + (1-\delta) K_t.$$

- Notice that the equilibrium can be represented by the system of equations. In particular, the Euler equation is necessary and sufficient for optimality.

Solution Concepts and Equilibrium Conditions

- Given K_0, a sequential competitive equilibrium is stochastic sequences: $\{c_t, K_{t+1}\}_{t=0}^{\infty}$ such that

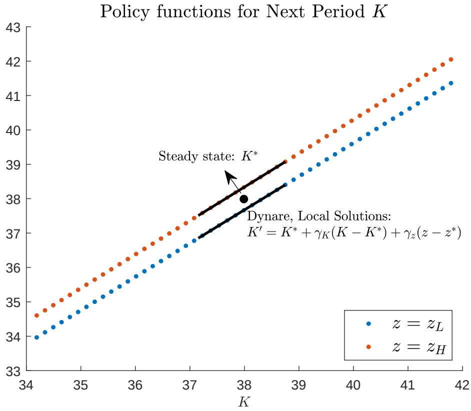
$$\text{Euler equation: } c_t^{-\sigma} = \beta \mathbb{E}_t \left[\left(\alpha z_{t+1} K_{t+1}^{\alpha-1} + (1-\delta) \right) c_{t+1}^{-\sigma} \right],$$
$$\text{Budget: } c_t + K_{t+1} = z_t K_t^{\alpha} + (1-\delta) K_t.$$

- Notice that the equilibrium can be represented by the system of equations. In particular, the Euler equation is necessary and sufficient for optimality.

- The GDSGE toolbox is looking for a recursive equilibrium: **functions** $c(z, K), K'(z, K)$ such that

$$c(z,K)^{-\sigma} = \beta \mathbb{E} \left[\left(\alpha z' [K'(z,K)]^{\alpha-1} + (1-\delta) \right) [c(z', K'(z,K))]^{-\sigma} \middle| z \right],$$
$$c(z,K) + K'(z,K) = zK^{\alpha} + (1-\delta)K.$$

Solution Concepts and Local v.s Global Solutions



- Local solutions: approximated around the steady state; implemented by Dynare
- Global solutions: solved at each collocation point
  - need to specify the domain of state variables: $K \in \{K_1, K_2, \ldots, K_N\}$, with $\underline{K} = K_1 < K_2 < \ldots < K_N = \overline{K}$.
- Local solutions approximate well for the current model

Policy Iteration Methods, A Prelim

- We solve the recursive system via policy iterations described by

$$c^{(n)}(z,K)^{-\sigma} = \beta \mathbb{E}\left[\left(\alpha z'[K'^{(n)}(z,K)]^{\alpha-1} + (1-\delta)\right)[c^{(n-1)}(z', K'^{(n)}(z,K))]^{-\sigma}\middle| z\right]$$
$$c^{(n)}(z,K) + K'^{(n)}(z,K) = zK^{\alpha} + (1-\delta)K$$

- Start from some initial conjecture $c^{(0)}$ (more on initialization later)

- At the n-th iteration, take function $c^{(n-1)}$ as given, and solve a two-equation system for unknowns (c, K') for each collocation point (z, K) to get updated functions $c^{(n)}$ and $K'^{(n)}$

- Iterate until $||c^{(n)} - c^{(n-1)}|| < \mathrm{Tol}$, for some predetermined Tol

Toolbox Code - Structure of the gmod File

```
% Parameters
parameters beta sigma alpha delta;
beta  = 0.99;       % discount factor
sigma = 2.0;        % CRRA coefficient
alpha = 0.36;       % capital share
delta = 0.025;      % depreciation rate
```

Parameters

```
% Exogenous States
var_shock z;
shock_num = 2;
z_low = 0.99; z_high = 1.01;
Pr_ll = 0.9;  Pr_hh  = 0.9;
z = [z_low,z_high];
shock_trans = [
  Pr_ll, 1-Pr_ll
  1-Pr_hh, Pr_hh
  ];
```

Exogenous States

```
% Endogenous States
var_state K;
Kss  = (alpha/(1/beta - 1 + delta))^(1/(1-alpha));
KPts = 101;
KMin = Kss*0.9;
KMax = Kss*1.1;
K    = linspace(KMin,KMax,KPts);
```

Endogenous States

```
% Interp
var_interp c_interp;
initial c_interp z.*K.^alpha+(1-delta)*K;
% Time iterations update
c_interp = c;
```

Policy Functions

```
% Endogenous variables as unknowns of equations
var_policy c K_next;
inbound c      0 z.*K.^alpha+(1-delta)*K;
inbound K_next  0 z.*K.^alpha+(1-delta)*K;
```

Unknowns

```
% Other endogenous variables
var_aux w;
```

```
model;
  % Budget constraints
  u_prime = c^(-sigma);
  mpk_next' = z'*alpha*K_next^(alpha-1) + 1-delta;

  % Evaluate the interpolation object to get future consumption
  c_future' = c_interp'(K_next);
  u_prime_future' = c_future'^(-sigma);

  % Calculate residual of the equation
  euler_residual = 1 - beta*GDSGE_EXPECT{u_prime_future'*mpk_next'}/u_prime;
  budget_residual = z*K^alpha + (1-delta)*K - c - K_next;

  % Calcualte other endogenous variables
  w = z*(1-alpha)*K^alpha;

  equations;
    euler_residual;
    budget_residual;
  end;
end;
```

Models and Equations

```
simulate;
  num_periods = 10000;
  num_samples = 100;
  initial K Kss;
  initial shock 1;
  var_simu c K w;
  K' = K_next;
end;
```

Simulations (Optional)

```
% Parameters
parameters beta sigma alpha delta;
beta  = 0.99;          % discount factor
sigma = 2.0;           % CRRA coefficient
alpha = 0.36;          % capital share
delta = 0.025;         % depreciation rate

% Exogenous States
var_shock z;
shock_num = 2;
z_low = 0.99; z_high = 1.01;
Pr_ll = 0.9;  Pr_hh  = 0.9;
z = [z_low,z_high];
shock_trans = [
  Pr_ll, 1-Pr_ll
  1-Pr_hh, Pr_hh
  ];
```

- **parameters**: parameters needed to define the model
- **var_shock**: exogenous states (e.g., productivity z here)
  - **shock_num**: number of discrete realizations
  - **shock_trans**: the full transition matrix (e.g, $\Pr(z \to z')$ here)

```
% Endogenous States
var_state K;
Kss  = (alpha/(1/beta - 1 + delta))^(1/(1-alpha));
KPts = 101;
KMin = Kss*0.9;
KMax = Kss*1.1;
K    = linspace(KMin,KMax,KPts);
```

- **var_state**: endogenous states (e.g., capital K here)
  - Need to specify a gird for each endogenous state
  - For example, here the grid is specified to be a 101-point equal-spaced grid over $[0.9 \times K^*, 1.1 \times K^*]$ where K^* is the steady state capita level
  - Generally, need the range of the grid to cover the *ergodic set* (more on this later)

```
% Interp
var_interp c_interp;
initial c_interp z.*K.^alpha+(1-delta)*K;
% Time iterations update
c_interp = c;
```

- **var_interp**: functions to be iterated over ($c(z, K)$ here)
  - Need to initialize each var_interp following keyword **initial**
  - Here $c(z, K)$ is initialized to be consuming all available resources
  - Need to specify the update of each policy function after a time step. Here it is updated to be unknown c solved out of the equation

```
% Endogenous variables as unknowns of equations
var_policy c K_next;
inbound c        0 z.*K.^alpha+(1-delta)*K;
inbound K_next   0 z.*K.^alpha+(1-delta)*K;

% Other endogenous variables
var_aux w;
```

- **var_policy**: unknowns to be solved at each collocation point, c and K' here
  - Need to specify the bounds of range over which solutions are searched for each unknown following keyword **inbound**
- **var_aux**: variables that are simple functions of other variables and need to be returned. Each **var_aux** needs to be defined in the model

```
model;
  % Budget constraints
  u_prime = c^(-sigma);
  kret_next' = z'*alpha*K_next^(alpha-1) + 1-delta;

  % Evaluate the interpolation object to get future consumption
  c_future' = c_interp'(K_next);
  u_prime_future' = c_future'^(-sigma);

  % Calculate residual of the equation
  euler_residual = 1 - beta*GDSGE_EXPECT{u_prime_future'*kret_next'}/u_prime;
  market_clear = z*K^alpha + (1-delta)*K - c - K_next;

  % Calcualte other endogenous variables
  w = z*(1-alpha)*K^alpha;

  equations;
    euler_residual;
    market_clear;
  end;
end;
```

- The system of equations for each collocation point of exogenous and endogenous states (z, K here) needs to be defined in the **model;** block
- The final system of equations is defined in the **equations;** block
- Any evaluation necessary for defining the equations is enclosed preceding the **equations;** block

```
model;
  % Budget constraints
  u_prime = c^(-sigma);
  kret_next' = z'*alpha*K_next^(alpha-1) + 1-delta

  % Evaluate the interpolation object to get future consumption
  c_future' = c_interp'(K_next);
  u_prime_future' = c_future'^(-sigma);

  % Calculate residual of the equation
  euler_residual = 1 - beta*GDSGE_EXPECT{u_prime_future'*kret_next'}/u_prime;
  market_clear = z*K^alpha + (1-delta)*K - c - K_next;

  % Calcualte other endogenous variables
  w = z*(1-alpha)*K^alpha;

  equations;
    euler_residual;
    market_clear;
  end;
end;
```

$$kret'(1) = z(1) \cdot \alpha \cdot K'^{\alpha-1} + 1 - \delta$$
$$kret'(2) = z(2) \cdot \alpha \cdot K'^{\alpha-1} + 1 - \delta$$

- Can use **parameters**, **var_shock**, **var_state** and **var_policy** in the model block
- A variable followed by a prime (') defines a vector of length **shock_num**
- A **var_shock** (z here) followed by a prime (') refers to this var_shock across realizations, which is of length shock_num
- The line defines capital return given choice K' across realizations of z

```
model;
  % Budget constraints
  u_prime = c^(-sigma);
  kret_next' = z'*alpha*K_next^(alpha-1) + 1-delta;

  % Evaluate the interpolation object to get future consumption
  c_future' = c_interp'(K_next);
  u_prime_future' = c_future'^(-sigma);

  % Calculate residual of the equation
  euler_residual = 1 - beta*GDSGE_EXPECT{u_prime_future'*kret_next'}/u_prime;
  market_clear = z*K^alpha + (1-delta)*K - c - K_next;

  % Calcualte other endogenous variables
  w = z*(1-alpha)*K^alpha;

  equations;
    euler_residual;
    market_clear;
  end;
end;
```

$$c'(1) = c^{(n-1)}(z(1), K')$$
$$c'(2) = c^{(n-1)}(z(2), K')$$

- A **var_interp** defined before (c_interp here) can be used as a function to evaluate policy functions referred by this var_interp from the last iteration
- A **var_interp** when called followed by a prime (') returns the evaluation across realizations of exogenous states ...

```
model;
  % Budget constraints
  u_prime = c^(-sigma);
  kret_next' = z'*alpha*K_next^(alpha-1) + 1-delta;

  % Evaluate the interpolation object to get future consumption
  c_future' = c_interp'(K_next);
  u_prime_future' = c_future'^(-sigma);

  % Calculate residual of the equation
  euler_residual = 1 - beta*GDSGE_EXPECT{u_prime_future'*kret_next'}/u_prime;
  market_clear = z*K^alpha + (1-de

  % Calcualte other endogenous var
  w = z*(1-alpha)*K^alpha;

  equations;
    euler_residual;
    market_clear;
  end;
end;
```

$$EulerResid = 1 - \beta \frac{\sum_{i'=1,2} kret'(i')c'(i')^{-\sigma}\Pr(i \to i')}{c^{-\sigma}}$$

- **GDSGE_EXPECT** is a built-in function that calculates the expectation of the expression conditional on the current realization of exogenous states

```
model;
  % Budget constraints
  u_prime = c^(-sigma);
  kret_next' = z'*alpha*K_next^(alpha-1) + 1-delta;

  % Evaluate the interpolation object to get future consumption
  c_future' = c_interp'(K_next);
  u_prime_future' = c_future'^(-sigma);

  % Calculate residual of the equation
  euler_residual = 1 - beta*GDSGE_EXPECT{u_prime_future'*kret_next'}/u_prime;
  market_clear = z*K^alpha + (1-delta)*K - c - K_next;

  % Calcualte other endogenous variables
  w = z*(1-alpha)*K^alpha;
  
  equations;
    euler_residual;
    market_clear;
  end;
end;
```

$w = z \cdot (1-\alpha)K^{\alpha}$

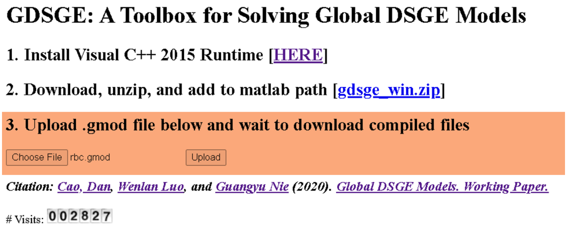
- Any **var_aux** (w here) needs to be evaluated in the model block so as to be returned
- **Notice**: Expressions in the **model;** block are executed sequentially. Do not use a variable before it's defined.

```
simulate;
  num_periods = 10000;
  num_samples = 100;
  initial K Kss;
  initial shock 1;
  var_simu c K w;
  K' = K_next;
end;
```

- The **simulate;** block specifies Monte-Carlo simulations
- Need to initiate all endogenous states (K here) following keyword **initial**, and the index of exogenous states following keyword **initial shock**
- Need to specify the transition of endogenous states (K' = K_next here)
- **var_simu** are variables recorded; **var_simu** must be in **var_policy** or **var_aux**

Parse the gmod File

- Upload the gmod file to an online compiler listed on **www.gdsge.com**
  - Also download the runtime libraries at the compiler website and add to path
  - Local compiler coming soon
  - Recompilation needed only if changing models (but not parameters or options)



- The online compiler returns a zip file (rbc.zip in this case), which contains
  - iter_*modname*.m and simulate_*modname*.m that can be called in MATLAB
  - mex_*modename*: dynamic libraries that will be called to do the actual computations

Solve Models on Local Computers: Policy Iterations

- In MATLAB run iter_rbc.m and assign results in variable IterRslt

```
>> IterRslt = iter_rbc;
Iter:10, Metric:0.385606, maxF:9.9913e-09
Elapsed time is 0.057094 seconds.
...
Iter:323, Metric:9.8918e-07, maxF:7.96884e-09
Elapsed time is 0.032591 seconds.
```

- In the printed information:
  - Iter: number of iterations
  - Metric: $||c^{(n)} - c^{(n-1)}||$ where $||\cdot||$ is the sup norm (max abs across states)
  - maxF: the max of absolute residual across all equations and all states
  - Elapsed time: time elapsed from the last print

- The returned IterRslt contains model structure, converged policy functions iterated on (i.e., var_interp), and var_policy and var_aux. For example,

```
>> IterRslt

IterRslt =

  struct with fields:

         Metric: 9.8918e-07
           Iter: 323
      shock_num: 2
    shock_trans: [2×2 double]
         params: [1×1 struct]
      var_shock: [1×1 struct]
      var_state: [1×1 struct]
     var_policy: [1×1 struct]
     var_interp: [1×1 struct]
        var_aux: [1×1 struct]
             pp: [1×1 struct]
    GNDSGE_PROB: [1×1 struct]
     var_others: [1×1 struct]
```

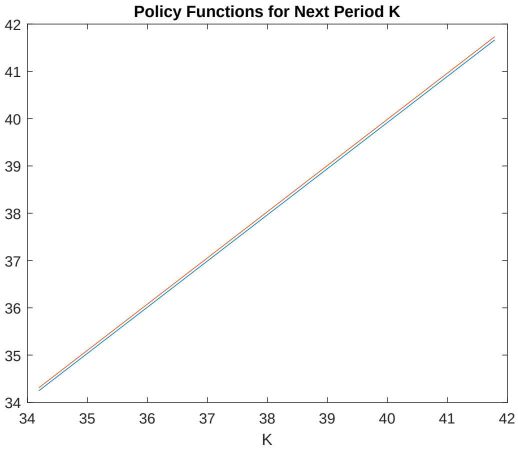
and

```
>> IterRslt.var_policy

ans =

  struct with fields:

         c: [2×101 double]
    K_next: [2×101 double]
```

- We can plot the converged policy functions or state transition functions

```
>> figure;
plot(IterRslt.var_state.K, IterRslt.var_policy.K_next);
xlabel('K'); title('Policy Functions for Next Period K');
```

which produces



Simulations using the Converged Policy Iterations

- The converged policy and transition functions can be passed to simulate_rbc.m for Monte-Carlo simulations, by calling

```
>> SimuRslt = simulate_rbc(IterRslt);
Periods: 1000
   shock       K       c       w
       2   37.89   2.755   2.392
Elapsed time is 0.818482 seconds.
...
Periods: 10000
   shock       K       c       w
       2   38.45   2.774   2.405
Elapsed time is 0.795403 seconds.
```

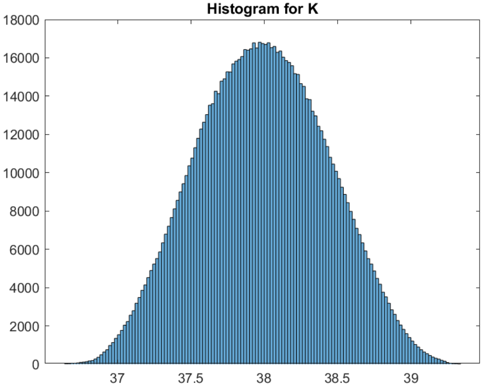
- The results (stored in SimuRslt) contain the panels of shock index and **var_simu** defined in the **simulate** block.

```
>> SimuRslt

SimuRslt =

  struct with fields:

    shock: [100×10001 double]
        K: [100×10001 double]
        c: [100×10000 double]
        w: [100×10000 double]
```

- We can inspect the ergodic distribution of the endogenous state K

```
>> histogram(SimuRslt.K); title('Histogram for K');
```

which produces



- We can inspect the simulated panels of **var_simu**, for example

```
>> plot(SimuRslt.w(1:2,1:100)'); title('Sample Paths of Wages');
```

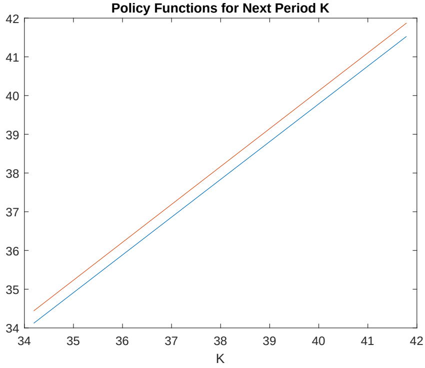
which produces the first two paths of wage for the first 100 periods

Resolving Models with Different Parameters

- The compiled code can be reused to solve models with different parameters
  For example, solve the model by increasing the size of the shock

```
>> options.z = [0.95,1.05]; % previously [0.99,1.01]
IterRslt = iter_rbc(options);
```

- Policy functions now show more visible difference across realizations of shocks



Resolving Models Starting from Converged Solutions

- A useful feature is to solve models with new parameters starting from previously converged solutions, by passing converged solutions in WarmUp

  ```
  >> options.z = [0.95,1.05]; % previously [0.99,1.01]
  options.WarmUp = IterRslt;
  IterRslt = iter_rbc(options);
  ```

  which starts from converged solutions and converge in fewer iterations

  ```
  Iter:330, Metric:0.000783625, maxF:9.89807e-09
  Elapsed time is 0.051842 seconds.
  ...
  Iter:457, Metric:9.90468e-07, maxF:7.93061e-09
  Elapsed time is 0.050782 seconds.
  ```

- This can also be used to overwrite options, for example

  ```
  >> options.PrintFreq = 100;
  options.SaveFreq = 100;
  ```

  sets the print frequency and save frequency to 100 (the default was 10)
  See the toolbox website for more options

- This can be used to overwrite the range of **var_state** to refine solutions. More on this later

Extending the RBC model with Irreversible Investment

- The RBC model can exhibit nonlinearity and state-dependence with simple extensions
- Assume the investment is partially irreversible:

$$I_t \geq \phi I_{ss},$$

- The optimality conditions now read

$$\begin{aligned} c_t^{-\sigma} - \mu_t c_t^{-\sigma} = \beta \mathbb{E}_t \left[\left(\alpha z_{t+1} K_{t+1}^{\alpha-1} + (1-\delta) \right) c_{t+1}^{-\sigma} - (1-\delta)\mu_{t+1} c_{t+1}^{-\sigma} \right] \\ \mu_t c_t^{-\sigma} \left[K_{t+1} - (1-\delta)K_t - \phi I_{ss} \right] = 0, \end{aligned}$$

in which $\mu_t c_t^{-\sigma}$ is the multiplier on the investment irreversible constraint. (Here we use $\mu_t c_t^{-\sigma}$ instead of μ_t alone to restrict its value in a box [0,1]).

Extending the RBC model with Irreversible Investment

- The RBC model can exhibit nonlinearity and state-dependence with simple extensions
- Assume the investment is partially irreversible:

$$I_t \geq \phi I_{ss},$$

- The optimality conditions now read

$$c_t^{-\sigma} - \mu_t c_t^{-\sigma} = \beta \mathbb{E}_t \left[\left(\alpha z_{t+1} K_{t+1}^{\alpha-1} + (1-\delta) \right) c_{t+1}^{-\sigma} - (1-\delta)\mu_{t+1} c_{t+1}^{-\sigma} \right]$$
$$\mu_t c_t^{-\sigma} \left[K_{t+1} - (1-\delta) K_t - \phi I_{ss} \right] = 0,$$

  in which $\mu_t c_t^{-\sigma}$ is the multiplier on the investment irreversible constraint. (Here we use $\mu_t c_t^{-\sigma}$ instead of μ_t alone to restrict its value in a box [0,1]).
- We set the values in **inbound** for the two inequalities with Kuhn-Tucker condition:

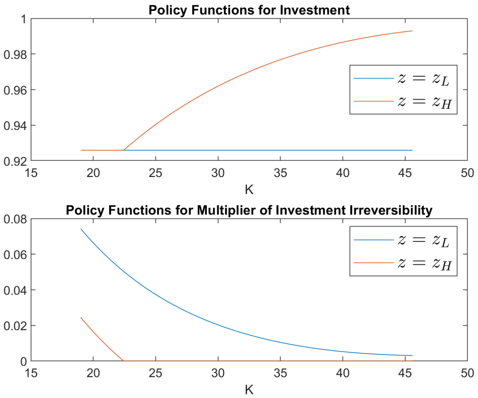
$$\mu_t \geq 0; \; K_{t+1} \geq (1-\delta) K_t + \phi I_{ss}.$$

Parameters | Exogenous States | Endogenous States | **Policy Functions** | **Unknowns** | **Models & Equations** | Simulations

```
var_interp c_interp mu_interp;
initial c_interp z.*K.^alpha+(1-delta)*K;
initial mu_interp 0;
% Time iterations update
c_interp = c;
mu_interp = mu;

% Endogenous variables as unknowns of equations
var_policy c K_next mu;
inbound c        0 z.*K.^alpha+(1-delta)*K;
inbound K_next   (1-delta)*K+phi*Iss z.*K.^alpha+(1-delta)*K;
inbound mu       0 1.0;

% Other endogenous variables
var_aux w Inv;

model;
  % Budget constraints
  u_prime = c^(-sigma);
  kret_next' = z'*alpha*K_next^(alpha-1) + 1-delta;

  % Evaluate the interpolation object to get future consumption
  c_future' = c_interp'(K_next);
  mu_future' = mu_interp'(K_next);
  u_prime_future' = c_future'^(-sigma);

  % Calculate residual of the equation
  euler_residual = 1 - beta*GDSGE_EXPECT{u_prime_future'*(kret_next'-(1-delta)*mu_future')}/(u_prime*(1-mu));
  market_clear = z*K^alpha + (1-delta)*K - c - K_next;

  % Calcualte other endogenous variables
  w = z*(1-alpha)*K^alpha;
  Inv = K_next - (1-delta)*K;

  equations;
    euler_residual;
    mu*(Inv - phi*Iss);
    market_clear;
  end;
end;
```

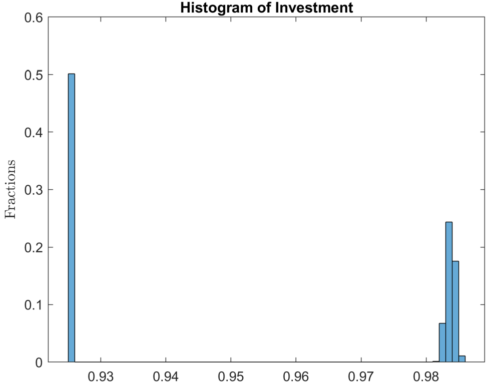
- Include $\mu(z, K)$ as **var_interp** and use it to interpolate for μ_{t+1}
- Include μ_t as **var_policy**.
- Modify the Euler equation and add the comp. slackness condition to system

Policy Functions with Irreversible Investment



- As shown, the investment irreversibility starts to bind (with multiplier $\mu_t > 0$), when z_t is low or capital K_t is low.

Occasionally Binding Irreversible Constraint at Ergodic Set



- As shown, the irreversible constraint binds when the realization of z is z_L
- Since z is a two-point process, this binding pattern seems a bit extreme
- See toolbox website on how to introduce a continuous z process (e.g., AR(1)), which generates richer binding patterns at the ergodic distribution

The GDSGE Framework

The GDSGE Framework, Summary

- With the RBC example, we are now ready to discuss the general framework.
- Many models fit in the framework and can be transformed into gmod files
- The framework also facilitates a comparison between global v.s local solutions
- Will refer back to the RBC example to discuss abstract concepts

Models with Short-run Equilibrium Conditions as Equations

- GDSGE is able to solve models with short-run equilibrium conditions represented by system of equations:

$$\boldsymbol{F}(s, x, z, \{s'(z'), x'(z')\}_{z' \in \mathcal{Z}}) = 0 \tag{1}$$

  where

  - $z \in \mathcal{Z} \subset \mathbb{R}^{d_z}$: a vector of exogenous shocks (productivity z in the RBC example)
  - $s \in \mathcal{S} \subset \mathbb{R}^{d_s}$: a vector of endogenous states variables (capital K)
  - $x \in \mathcal{X} \subset \mathbb{R}^{d_x}$: a vector of endogenous policy variables (c and K')
  - $s'(z'), x'(z')$: future states and policies that depend on the realizations of future shocks, ($K'(z') \equiv K', \forall z'$; $c'(z')$ in expectation operator); can accommodate more general dependence than expectation

- RBC example: 2 unknowns with 2 equations: Euler equation and budget

Models with Short-run Equilibrium Conditions as Equations

- GDSGE is able to solve models with short-run equilibrium conditions represented by system of equations:

$$\boldsymbol{F}(s, x, z, \{s'(z'), x'(z')\}_{z' \in \mathcal{Z}}) = 0 \tag{1}$$

where

- $z \in \mathcal{Z} \subset \mathbb{R}^{d_z}$: a vector of exogenous shocks (productivity z in the RBC example)
- $s \in \mathcal{S} \subset \mathbb{R}^{d_s}$: a vector of endogenous states variables (capital K)
- $x \in \mathcal{X} \subset \mathbb{R}^{d_x}$: a vector of endogenous policy variables (c and K')
- $s'(z'), x'(z')$: future states and policies that depend on the realizations of future shocks, ($K'(z') \equiv K', \forall z'$; $c'(z')$ in expectation operator); can accommodate more general dependence than expectation

- RBC example: 2 unknowns with 2 equations: Euler equation and budget

- Therefore, the toolbox (so far) cannot solve
  - Decision problems that are non-concave or involve discrete choices, whose optimality condition cannot be represented by equations.
  - We are working on transforming discrete-choice into continuous-choice

Accommodate Inequality Constraints

- Models with inequality constraints

$$\begin{aligned} \boldsymbol{F}\left(s, x, z, \{s'(z'), x'(z')\}_{z' \in \mathcal{Z}}\right) &= 0 \\ \boldsymbol{G}\left(s, x, z, \{s'(z'), x'(z')\}_{z' \in \mathcal{Z}}\right) &\geq 0 \end{aligned}$$

can be transformed to the general formulation (1), by writing

$$\hat{\boldsymbol{F}} = \begin{pmatrix} \boldsymbol{F} \\ \boldsymbol{G} - \eta \end{pmatrix} \tag{2}$$

with $\eta \geq 0$ being an additional policy variable and expand $\hat{x} = (x, \eta)$

Accommodate Inequality Constraints

- Models with inequality constraints

$$\boldsymbol{F}\left(s, x, z, \{s'(z'), x'(z')\}_{z' \in \mathcal{Z}}\right) = 0$$
$$\boldsymbol{G}\left(s, x, z, \{s'(z'), x'(z')\}_{z' \in \mathcal{Z}}\right) \geq 0$$

can be transformed to the general formulation (1), by writing

$$\hat{\boldsymbol{F}} = \begin{pmatrix} \boldsymbol{F} \\ \boldsymbol{G} - \eta \end{pmatrix} \tag{2}$$

with $\eta \geq 0$ being an additional policy variable and expand $\hat{x} = (x, \eta)$

- In the investment irreversible example, we add a multiplier $\mu \geq 0$ into the Euler equation and the complementary slackness condition as an additional equation
- This is how we handle occasionally binding constraints with equation solvers

Solution Concepts and the Policy Iteration Algorithm

$$\boldsymbol{F}(s, x, z, \{s'(z'), x'(z')\}_{z' \in \mathcal{Z}}) = 0 \tag{1}$$

- A *recursive equilibrium* is a solution to (1) of the form

$$x = \mathcal{P}(z, s)$$

and

$$s'(z') = \mathcal{T}(z, z', s)$$

where $\mathcal{P}$ and $\mathcal{T}$ are equilibrium policy and transition functions, respectively.

- The algorithm starts with an initial guess for policy and transition functions

$$\left\{\mathcal{P}^{(0)}(.,.), \mathcal{T}^{(0)}(.,.,.)\right\}$$

Given $\mathcal{P}^{(n)}$ and $\mathcal{T}^{(n)}$, $\mathcal{P}^{(n+1)}$ and $\mathcal{T}^{(n+1)}$ are determined by solving the following system of equations:

$$\boldsymbol{F}\left(s, x, z, \left\{s'(z'), \mathcal{P}^{(n)}\left(z', s'(z')\right)\right\}_{z' \in \mathcal{Z}}\right) = 0.$$

with unknowns x and $\{s'(z')\}_{z' \in \mathcal{Z}}$ for each

$$(s, z) \in \mathcal{C}^{(n)} \subset \mathcal{Z} \times \mathcal{S}.$$

- Mapping to the toolbox:
  - z: **var_shock** (z). s: **var_state** (K). x: **var_policy**, **var_aux** (c, w, K')
  - $s'(z')$: K'
  - $\mathcal{P}^{(n)}$: **var_interp** (c_interp)
  - $\mathcal{P}^{(0)}$: **initial**, $c^{(0)}(z, K) = zK^{\alpha} + (1-\delta)K$
  - $\boldsymbol{F}$: Euler equation residual and the market clearing condition

Bianchi (2011): Sudden Stops in Open Economies

Bianchi (2011), Summary

- A model in which the borrowing constraint depends on a (commodity) price
- A negative shock that lowers the non-tradable good price tightens the borrowing constraint, induces deleveraging and reduction of tradable consumption, and further lowers the non-tradable price, amplifying the effects
- Can generate current account reversals resembling crises in emerging markets
- The model is highly nonlinear when the borrowing constraint binds. The borrowing constraint binds occasionally, necessitating a global solution

Bianchi (2011), Summary

- A model in which the borrowing constraint depends on a (commodity) price
- A negative shock that lowers the non-tradable good price tightens the borrowing constraint, induces deleveraging and reduction of tradable consumption, and further lowers the non-tradable price, amplifying the effects
- Can generate current account reversals resembling crises in emerging markets
- The model is highly nonlinear when the borrowing constraint binds. The borrowing constraint binds occasionally, necessitating a global solution
- Use the model to illustrate how to
    - introduce endogenous borrowing constraints
    - initiate the policy function $\mathcal{P}^{(0)}(z, s)$ with **model_init** block
    - refine solutions over expanded and refined grids
    - use adaptive grids to obtain accurate solutions efficiently

The Model

- Preferences:

$$\mathbb{E}\Big[\sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}}{1-\sigma}\Big],$$

  with the composite consumption

$$c_t = \mathcal{A}\left(c_t^T, c_t^N\right) \equiv [\omega(c_t^T)^{-\eta} + (1-\omega)(c_t^N)^{-\eta}]^{-\frac{1}{\eta}},$$

  where $\eta > -1$ determines the *elasticity of substitution* between tradable consumption c_t^T and non-tradable c_t^N. $\omega \in (0,1)$ is the weight on tradables

- Endowments: $\left(y_t^T, y_t^N\right)$ follows an exogenous AR(1) process

- Incomplete-markets: saving/borrowing can only be via a *state non-contingent* bond b_{t+1} at a world (exogenous) interest rate r

- Budget constraint:

$$b_{t+1} + c_t^T + p_t^N c_t^N = b_t(1+r) + y_t^T + p_t^N y_t^N.$$

- Borrowing constraint:

$$b_{t+1} \geq -(\kappa^N p_t^N y_t^N + \kappa^T y_t^T).$$

where $\kappa^N, \kappa^T > 0$ are parameters governing the *collaterability* of non-tradable and tradable endowments

Equilibrium Conditions

- Optimality:

$$p_t^N = \Big(\frac{1-\omega}{\omega}\Big)\Big(\frac{c_t^T}{c_t^N}\Big)^{\eta+1}, \quad \text{(Tradable v.s Non-tradable)}$$

$$\lambda_t = \beta(1+r)\mathbb{E}_t\lambda_{t+1} + \mu_t, \quad \text{(Bond Euler Equation)}$$

$$\mu_t\big[b_{t+1} + (\kappa^N p_t^N y_t^N + \kappa^T y_t^T)\big] = 0, \quad \text{(Comp. Slack. for Borrowing Constraint)}$$

where

$$\lambda_t = c_t^{-\sigma}\frac{\partial\mathcal{A}(c_t^T, c_t^N)}{\partial c_t^T}.$$

- Market clearing conditions:

$$c_t^N = y_t^N,$$
$$c_t^T = y_t^T + b_t(1+r) - b_{t+1}.$$

Mapping to GDSGE Framework and the Toolbox

- Exogenous states, **var_shock**: $z = (y_t^N, y_t^T)$
- Endogenous states, **var_state**: $s = b_t$
- Policy variables (unknowns), **var_policy**: $x = (\mu_t, c_t^T, c_t^N, b_{t+1}, p_t^N)$
- Policy functions iterated over, **var_interp**: $\lambda(z, b)$
- Equations $\boldsymbol{F}$ at n-th iteration:

$$
\begin{aligned}
p_t^N &= \left(\frac{1-\omega}{\omega}\right)\left(\frac{c_t^T}{c_t^N}\right)^{\eta+1}, \\
\lambda_t &= \beta(1+r)\mathbb{E}\left[\lambda^{(n-1)}(z', b_{t+1})|z\right] + \mu_t, \\
\mu_t\left[b_{t+1} + (\kappa^N p_t^N y_t^N + \kappa^T y_t^T)\right] &= 0, \\
c_t^N &= y_t^N, \\
c_t^T &= y_t^T + b_t(1+r) - b_{t+1}.
\end{aligned}
$$

- Update $\lambda^{(n)} = \lambda_t$; need to include λ_t as a **var_aux**.

Biianchi (2011) in 100 Lines of GDSGE Code

```
% Parameters
parameters r sigma eta kappaN kappaT omega beta;
r = 0.04;
sigma = 2;
eta = 1/0.83 - 1;
kappaN = 0.32;
kappaT = 0.32;
omega = 0.31;
beta = 0.91;

% States
var_state b;
bPts = 101;
bMin=-0.5;
bMax=0.0;
b=linspace(bMin,bMax,bPts);

% Shocks
var_shock yT yN;
yPts = 4;
shock_num=16;

yTEpsilonVar = 0.00219;
yNEpsilonVar = 0.00167;
rhoYT = 0.901;
rhoYN = 0.225;

[yTTrans,yT] = markovappr(rhoYT,yTEpsilonVar^0.5,1,yPts);
[yNTrans,yN] = markovappr(rhoYN,yNEpsilonVar^0.5,1,yPts);

shock_trans = kron(yNTrans,yTTrans);
[yT,yN] = ndgrid(yT,yN);
yT = exp(yT(:)');
yN = exp(yN(:)');

% Define the last-period problem
var_policy_init dummy;
inbound_init dummy -1.0 1.0;

var_aux_init c lambda;
model_init;
  cT = yT + b*(1+r);
  cN = yN;
  c = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta);
  partial_c_partial_cT = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta-1) * omega * cT^(-eta-1);
  lambda = c^(-sigma)*partial_c_partial_cT;

  equations;
    0;
  end;
end;

% Implicit state transition functions
var_interp lambda_interp;
initial lambda_interp lambda;
lambda_interp = lambda;

% Endogenous variables, bounds, and initial values
var_policy nbNext mu cT pN;
inbound nbNext 0.0 10.0;
inbound mu 0.0 1.0;
inbound cT 0.0 10.0;
inbound pN 0.0 10.0;

var_aux c lambda bNext;
var_output bNext pN;

model;
  % Non tradable market clear
  cN = yN;

  % Transform variables
  bNext = nbNext - (kappaN*pN*yN + kappaT*yT);
  % Interp future values
  lambdaFuture' = lambda_interp'(bNext);

  % Calculate Euler residuals
  c = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta);
  partial_c_partial_cT = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta-1) * omega * cT^(-eta-1);
  lambda = c^(-sigma)*partial_c_partial_cT;
  euler_residual = 1 - beta*(1+r) * GDSGE_EXPECT{lambdaFuture'}/lambda - mu;

  % Price consistent
  price_consistency = pN - ((1-omega)/omega)*(cT/cN)^(eta+1);

  % budget constraint
  budget_residual = b*(1+r)+yT+pN*yN - (bNext+cT+pN*cN);

  equations;
    euler_residual;
    mu*nbNext;
    price_consistency;
    budget_residual;
  end;
end;

simulate;
  num_periods = 1000;
  num_samples = 100;
  initial b 0.0
  initial shock 1;
  var_simu c pN;
  b' = bNext;
end;
```

```
var_state b;

var_shock yT yN;

var_interp lambda_interp;

var_policy nbNext mu cT pN;

model;
  % Non tradable market clear
  cN = yN;

  % Transform variables
  bNext = nbNext - (kappaN*pN*yN + kappaT*yT);
  % Interp future values
  lambdaFuture' = lambda_interp'(bNext);

  % Calculate Euler residuals
  c = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta);
  partial_c_partial_cT = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta-1) * omega * cT^(-eta-1);
  lambda = c^(-sigma)*partial_c_partial_cT;
  euler_residual = 1 - beta*(1+r) * GDSGE_EXPECT{lambdaFuture'}/lambda - mu;

  % Price consistent
  price_consistency = pN - ((1-omega)/omega)*(cT/cN)^(eta+1);

  % budget constraint
  budget_residual = b*(1+r)+yT+pN*yN - (bNext+cT+pN*cN);

  equations;
    euler_residual;
    mu*nbNext;
    price_consistency;
    budget_residual;
  end;
end;
```

- The system of equations can be further simplified, by e.g., directly imposing

$$c^N = y^N$$

from the market clearing of non-tradable goods

```
var_state b;

var_shock yT yN;

var_interp lambda_interp;

var_policy nbNext mu cT pN;
inbound nbNext 0.0 10.0;
model;
  % Non tradable market clear
  cN = yN;

  % Transform variables
  bNext = nbNext - (kappaN*pN*yN + kappaT*yT);
  % Interp future values
  lambdaFuture' = lambda_interp'(bNext);

  % Calculate Euler residuals
  c = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta);
  partial_c_partial_cT = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta-1) * omega * cT^(-eta-1);
  lambda = c^(-sigma)*partial_c_partial_cT;
  euler_residual = 1 - beta*(1+r) * GDSGE_EXPECT{lambdaFuture'}/lambda - mu;

  % Price consistent
  price_consistency = pN - ((1-omega)/omega)*(cT/cN)^(eta+1);

  % budget constraint
  budget_residual = b*(1+r)+yT+pN*yN - (bNext+cT+pN*cN);

  equations;
    euler_residual;
    mu*nbNext;
    price_consistency;
    budget_residual;
  end;
end;
```

- **Trick 1**: transform the borrowing constraint $b_{t+1} \geq -(\kappa^N p_t^N y_t^N + \kappa^T y_t^T)$ into

$$\mathrm{nb}_{t+1} \equiv b_{t+1} + \kappa^N p_t^N y_t^N + \kappa^T y_t^T \geq 0,$$

  and include nb_{t+1} instead of b_{t+1} as unknown. **lb** of nb_{t+1} is fixed at 0.
- example of dealing with inequality constraint in GDSGE in equation (2).

```
var_state b;

var_shock yT yN;

var_interp lambda_interp;

var_policy nbNext mu cT pN;
inbound mu 0.0 1.0;
model;
  % Non tradable market clear
  cN = yN;

  % Transform variables
  bNext = nbNext - (kappaN*pN*yN + kappaT*yT);
  % Interp future values
  lambdaFuture' = lambda_interp'(bNext);

  % Calculate Euler residuals
  c = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta);
  partial_c_partial_cT = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta-1) * omega * cT^(-eta-1);
  lambda = c^(-sigma)*partial_c_partial_cT;
  euler_residual = 1 - beta*(1+r) * GDSGE_EXPECT{lambdaFuture'}/lambda - mu;

  % Price consistent
  price_consistency = pN - ((1-omega)/omega)*(cT/cN)^(eta+1);

  % budget constraint
  budget_residual = b*(1+r)+yT+pN*yN - (bNext+cT+pN*cN);

  equations;
    euler_residual;
    mu*nbNext;
    price_consistency;
    budget_residual;
  end;
end;
```

- **Trick 2**: transform the Euler equation into

$$1 = \beta(1+r)\mathbb{E}_t\frac{\lambda_{t+1}}{\lambda_t} + \frac{\mu_t}{\lambda_t}.$$

- The normalized multiplier $\tilde{\mu}_t \equiv \frac{\mu_t}{\lambda_t}$ thus lies in $[0,1]$.
- The resulting Euler equation is also normalized to be in $[0,1]$.

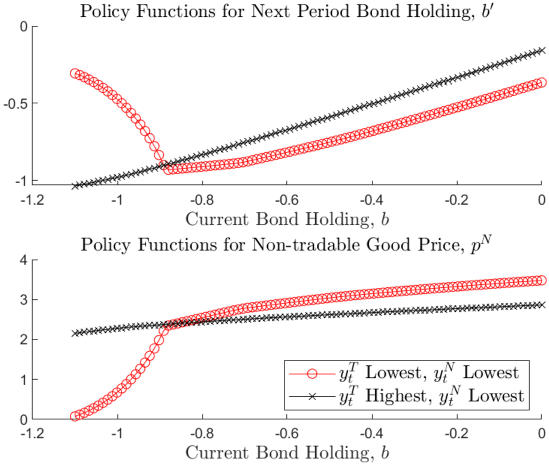
Initiate Policy Functions with model_init


```
% Define the last-period problem
var_policy_init dummy;
inbound_init dummy -1.0 1.0;

var_aux_init c lambda;
model_init;
  cT = yT + b*(1+r);
  cN = yN;
  c = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta);
  partial_c_partial_cT = (omega*cT^(-eta) + (1-omega)*cN^(-eta))^(-1/eta-1) * omega * cT^(-eta-1);
  lambda = c^(-sigma)*partial_c_partial_cT;

  equations;
    0;
  end;
end;
var_interp lambda_interp;
initial lambda_interp lambda;
lambda_interp = lambda;
```

- Crucial to initialize the **var_interp** properly for the algorithm to work
- Initializing with a last-period problem in finite-horizon economies usually works
- Define a potential different system of equation in **model_init**
- Define **var_policy_init** for unknowns and **var_aux_init** for extra returns
- **var_aux_init** and **var_aux_init** can be used following keyword **initial**

Inspecting the Policy Functions

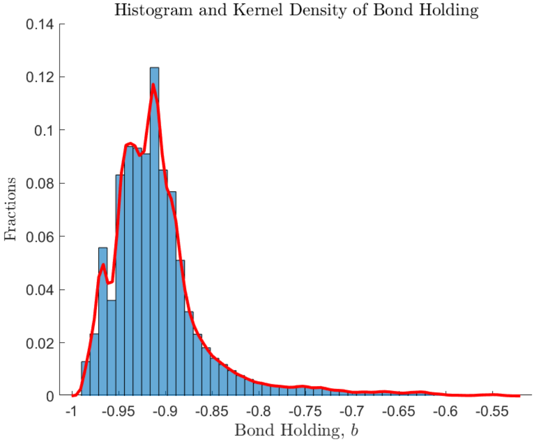
- Upload the gmod file. Run iter_bianchi2011 in MATLAB. Plot policy functions



- As shown, the policy functions are highly nonlinear, and the nonlinearity is state-dependent

Inspecting the Ergodic Distribution

- Pass the converged policy iteration results into simulate_bianchi2011 to run simulations, and inspect the ergodic distribution of bond holdings



- As shown, the nonlinearity region is in the model's ergodic set (i.e., appearing with positive probability), but is occasionally appearing

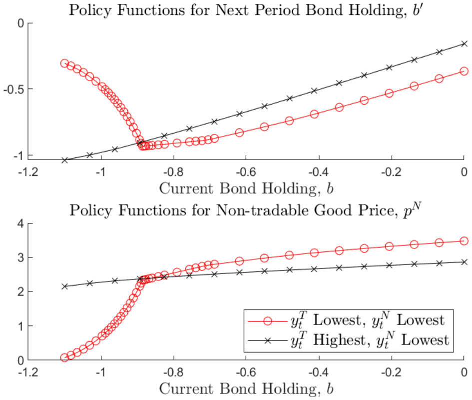
Using the Adaptive Grid Interpolation Method

- **Observation**: the model nonlinearity is state-dependent, i.e., linear functions approximate well for some regions but not for other
  Question: is there a more efficient way to specify grid points?
- **Answer**: Adaptive Grid (Ma and Zabaras, 09; Brumm and Scheidegger, 17)
- Without going into technical details, in the toolbox this can be done by adding

  ```
  USE_ASG=1; USE_SPLINE=0;
  ```

  in the gmod file (recompilation needed)
- See the Bianchi2011 example on the toolbox website for how to inspect policy functions with adaptive grids

Policy Functions with the Adaptive Grid Method



- As shown, now the toolbox automatically puts more grid points in regions with higher nonlinearity
- Importantly, the grid can be different across realizations of exogenous shocks

Further Discussions

In the Bianchi (2011) example on the toolbox website, we also guide you to

- solve the planner's problem that accounts for the effects of prices on the borrowing constraint
- interpolate policy and state transition functions for fast simulations

Other comments

- The adaptive grid method is designed based on sparse grid and is especially powerful in dealing with models with high dimensions
  - Cao, Evans and Luo (2020): a two-country IF model with incomplete markets, portfolio choice and occasionally binding constraints, up to 6 endogenous states
- We next turn to a two-agent model with two endogenous states (capital and bond) and occasionally binding collateral constraints

Kiyotaki & Moore (1997) with Risk-averse Agents

KM1997, Summary

- The interaction between capital price and output through the endogenous collateral constraint produces amplified and persistent effects of shocks to the economy.
- The original model is relatively simple with risk-neutral agents and unanticipated MIT shocks.
- As a contributed example, the model is augmented with risk-averse agents and recurrent aggregate shocks.

KM1997, Summary

- The interaction between capital price and output through the endogenous collateral constraint produces amplified and persistent effects of shocks to the economy.
- The original model is relatively simple with risk-neutral agents and unanticipated MIT shocks.
- As a contributed example, the model is augmented with risk-averse agents and recurrent aggregate shocks.
- Use the model to illustrate how to:
  - solve model with two endogenous states with occasionally binding constraints
  - deal with endogenous state variable with implicit law of motion - consistency equation
  - generate Impulse Response Function with recurrent aggregate shocks

The Model

- Two sectors: Farmers and Gatherers. Both produce using capital as input.
- A farmer maximizes

$$\mathbb{E}_0 \sum_t \beta^t \frac{(x_t)^{1-\sigma}}{1-\sigma},$$

subject to the budget constraint:

$$x_t + q_t k_{t+1} + \frac{b_{t+1}}{R_t} = y_t + q_t k_t + b_t,$$

where production $y_t = A_t (a + c) k_t$. She is also subject to:

$$x_t \geq c A_t k_t,$$
$$b_{t+1} + \theta \underline{q}_{t+1} k_{t+1} \geq 0,$$

in which $\theta \in [0, 1]$, and $\underline{q}_{t+1}$ is the lowest possible capital price in the next period.

The Model

- Similarly, a gatherer maximizes

$$\mathbb{E}_0 \sum_t \left(\beta'\right)^t \frac{\left(x_t'\right)^{1-\sigma}}{1-\sigma},$$

subject to the budget constraint,

$$x_t' + q_t k_{t+1}' + \frac{b_{t+1}'}{R_t} = y_t' + q_t k_t' + b_t,$$

in which her production function is concave, $y_t' = \underline{A}_t \left(k_t'\right)^{\alpha}$. Assume $\underline{A}_t = \delta A_t$ with $\delta < 1$, and $\beta' > \beta$.

- Optimality:

$$
\begin{aligned}
(x_t)^{-\sigma} - \lambda_t + \eta_t &= 0, && \text{(FOC of } x_t\text{)} \\
\eta_t \left(x_t - A_t c k_t\right) &= 0, && \text{(Slackness of } x_t\text{)} \\
-q_t \lambda_t + \theta \underline{q}_{t+1} \mu_t + \beta \mathbb{E}_t \left\{\xi_{t+1}\right\} &= 0, && \text{(FOC of } k_t\text{)} \\
-\frac{1}{R_t} \lambda_t + \mu_t + \beta \mathbb{E}_t \left\{\lambda_{t+1}\right\} &= 0, && \text{(FOC of } b_t\text{)} \\
\mu_t \left[\theta \underline{q}_{t+1} k_{t+1} + b_{t+1}\right] &= 0, && \text{(Slackness of CC)} \\
\left(x_t'\right)^{-\sigma} - \lambda_t' &= 0, && \text{(FOC of } x_t'\text{)} \\
q_t = \beta' \mathbb{E}_t \left\{ \left(q_{t+1} + \alpha \left(k_{t+1}'\right)^{\alpha-1} \right) \lambda_{t+1}' / \lambda_t' \right\}, & && \text{(FOC of } k_t'\text{)} \\
1 = \beta' R_t \mathbb{E}_t \left\{ \lambda_{t+1}' / \lambda_t' \right\}. & && \text{(FOC of } b_t'\text{)}
\end{aligned}
$$

with auxiliary variable $\xi_{t+1} = \left(q_{t+1} + a + c\right) \lambda_{t+1} - c\eta_{t+1}$ to simplify notation.

- Market clearing conditions:

$$
\begin{aligned}
b_{t+1} + b_{t+1}' &= 0, \\
k_{t+1} + k_{t+1}' &= \overline{K}, \\
x_t + x_t' = Y_t &= y_t + y_t'.
\end{aligned}
$$

Wealth Share as Endogenous State

- Define the farmers' and gatherers' wealth shares as

$$\omega_t = \frac{q_t k_t + b_t}{q_t \overline{K}},$$
$$\omega_t' = \frac{q_t k_t' + b_t'}{q_t \overline{K}}.$$

  In equilibrium, the market clearing conditions imply $\omega_t + \omega_t' = 1$. Thus we only need to keep track of ω_t.

- We use $\{k, \omega\}$ as endogenous states, instead of $\{k, b\}$.

Wealth Share as Endogenous State

- Define the farmers' and gatherers' wealth shares as

$$\omega_t = \frac{q_t k_t + b_t}{q_t \overline{K}},$$
$$\omega_t' = \frac{q_t k_t' + b_t'}{q_t \overline{K}}.$$

  In equilibrium, the market clearing conditions imply $\omega_t + \omega_t' = 1$. Thus we only need to keep track of ω_t.

- We use $\{k, \omega\}$ as endogenous states, instead of $\{k, b\}$.
- In general, using ω_t has 3 advantages:
  1. avoid multiple equilibria issues (as in the current model)
  2. easy to determine the feasible set of state ($\underline{\omega} = 1 - \theta$)
  3. reduce dimensionality in models with many assets
     (Heaton and Lucas, 96; Kubler and Schmedders, 03; Cao and Nie, 17)

Mapping to GDSGE and Consistency Equation

- Exogenous state, **var_shock**: $z = A_t$
- Endogenous states, **var_state**: $s = (k_t, \omega_t)$
- Policy variables (unknowns), **var_policy**: $x = (x_t, x_t', k_{t+1}, b_{t+1}, R_t, q_t, \eta_t, \mu_t)$
- Future policy functions, **var_interp**:
  $\left(\lambda_{t+1}, \lambda'_{t+1}, q_{t+1}, \xi_{t+1}\right) = \mathcal{P}^{(n-1)}\left(A_{t+1}, k_{t+1}, \omega_{t+1}\right)$

Mapping to GDSGE and Consistency Equation

- Exogenous state, **var_shock**: $z = A_t$
- Endogenous states, **var_state**: $s = (k_t, \omega_t)$
- Policy variables (unknowns), **var_policy**: $x = (x_t, x'_t, k_{t+1}, b_{t+1}, R_t, q_t, \eta_t, \mu_t)$
- Future policy functions, **var_interp**: $\left(\lambda_{t+1}, \lambda'_{t+1}, q_{t+1}, \xi_{t+1}\right) = \mathcal{P}^{(n-1)}\left(A_{t+1}, k_{t+1}, \omega_{t+1}\right)$
- Wait! Do we know endogenous state ω_{t+1}?

$$\omega_{t+1}\left(z_t, s_t, z_{t+1}\right) = \frac{q_{t+1}\left(z_{t+1}, k_{t+1}, \omega_{t+1}\right) k_{t+1} + b_{t+1}}{q_{t+1}\left(z_{t+1}, k_{t+1}, \omega_{t+1}\right) \overline{K}}.$$

Mapping to GDSGE and Consistency Equation

- Exogenous state, **var_shock**: $z = A_t$
- Endogenous states, **var_state**: $s = (k_t, \omega_t)$
- Policy variables (unknowns), **var_policy**: $x = (x_t, x_t', k_{t+1}, b_{t+1}, R_t, q_t, \eta_t, \mu_t)$
- Future policy functions, **var_interp**: $\left(\lambda_{t+1}, \lambda_{t+1}', q_{t+1}, \xi_{t+1}\right) = \mathcal{P}^{(n-1)}\left(A_{t+1}, k_{t+1}, \omega_{t+1}\right)$
- Wait! Do we know endogenous state ω_{t+1}?

$$\omega_{t+1}\left(z_t, s_t, z_{t+1}\right) = \frac{q_{t+1}\left(z_{t+1}, k_{t+1}, \omega_{t+1}\right) k_{t+1} + b_{t+1}}{q_{t+1}\left(z_{t+1}, k_{t+1}, \omega_{t+1}\right) \overline{K}}.$$

- **Solution**: we include $\{\omega_{t+1}\left(z_{t+1}\right)\}$ as unknowns, and the **consistency equation** above in **equations;**.
- Revised **var_policy**: $x = (x_t, x_t', k_{t+1}, b_{t+1}, R_t, q_t, \eta_t, \mu_t, \{\omega_{t+1}\left(z_{t+1}\right)\})$
- Need to include $(\lambda_t, \lambda_t', \xi_t)$ into **var_aux**

```
parameters a c alower sigma betaF betaG alpha Kbar theta;
a = 0.7;        % tradable productivity of farmer
c = 0.3;        % nontradable productivity of farmer
alower=0.9;     % tradable productivity of gatherer
sigma=1;        % risk aversion coefficient
betaF=0.95;     % discount factor of farmer
betaG=0.98;     % discount factor of gatherer
alpha=0.7;      % coefficient of gatherer's production
Kbar=1;         % fixed capital stock
theta=0.9;      % collateral parameter

INTERP_ORDER = 2;
EXTRAP_ORDER = 2;
SIMU_RESOLVE=0;
SIMU_INTERP=1;  % Use interpolation for fast simulate

%  ENDOGENOUS STATES
var_state kF omega;
kFPts=41;
kFMin=0.02;
kFMax=0.95;
kF=linspace(kFMin,kFMax,kFPts);

omegaPts=40;
omegaMin=0;
omegaMax=0.2;
omega=linspace(omegaMin,omegaMax,omegaPts);

%  EXOGENOUS STATES
var_shock A;

shock_num=3;
A = [0.99 1 1.01];
shock_trans = ones(shock_num,shock_num)/shock_num;

% INITIALIZATION
var_policy_init nxF xG eta kFnext R q nbFnext muF;
inbound_init nxF        0  10;
inbound_init xG         0  10;
inbound_init eta        0  1;
inbound_init kFnext     0  Kbar;
inbound_init R          0 10;
inbound_init q          0 10;
inbound_init nbFnext    0 10;
inbound_init muF        0 1;

var_aux_init loglambdaF loglambdaG logauxF;

model_init; % This corresponds to the T-1 problem
kG  =  Kbar-kF;             % market clearing for capital state
kGnext  =  Kbar-kFnext;     % market clearing for capital policy
bFnext = nbFnext;
bGnext = -bFnext;           % market clearing

% Backout xG and marginal utility
Y  =  A*(a+c)*kF + alower*A*kG^alpha;  % aggregate output
xF = nxF + c*A*kF;  % consumption of farmer

%multiplier for nontradable is eta*lambdaF
lambdaF = xF^(-sigma)/(1-eta);
lambdaG = xG^(-sigma);
loglambdaF = log(lambdaF);
loglambdaG = log(lambdaG);
auxF = (q+A*(a+c)-c*A*eta)*lambdaF;
logauxF = log(auxF);

% In the last period, people consume everything, and qT=0
xF_next =  (a+c)*kFnext + bFnext;
xG_next  = alower*kGnext^alpha + bGnext;
lambdaF_next = xF_next^(-sigma);
lambdaG_next = xG_next^(-sigma);

foc_bondG   = 1 - R*betaG*lambdaG_next / lambdaG;
foc_kG      = q   - betaG*lambdaG_next*alower*alpha*kGnext^(alpha-1)/lambdaG;

foc_bondF   = 1  -  R*betaF*lambdaF_next / lambdaF   - muF;
foc_kF      = q  -  betaF*(a+c)*lambdaF_next/lambdaF;
slack_bF    = muF*nbFnext;
slack_xF    = eta*nxF;
budgetF     = q*kFnext +bFnext/R + xF - A*(a+c)*kF - omega*q*Kbar;
MC_Y = Y - xF - xG;

equations;
foc_bondG;
foc_kG;
foc_bondF;
foc_kF;
slack_bF;
slack_xF;
budgetF;
MC_Y;
end;
end;

var_interp loglambdaF_interp loglambdaG_interp logauxF_interp q_interp;
loglambdaF_interp = loglambdaF;
loglambdaG_interp = loglambdaG;
logauxF_interp    = logauxF;
q_interp          = q;

initial loglambdaF_interp loglambdaF;
initial loglambdaG_interp loglambdaG;
initial logauxF_interp logauxF;
initial q_interp q;

var_policy nxF xG eta kFnext R q nbFnext muF omega_next[3];
inbound nxF         0  2;
inbound xG          0  2;
inbound eta         0  1;
inbound kFnext      0  Kbar;
inbound R           0  1.5 adaptive(1.5);
inbound q           0 10   adaptive(1.5);
inbound nbFnext     0 10   adaptive(1.5);
inbound muF         0 1;
inbound omega_next  0 1;

var_aux xF loglambdaF loglambdaG logauxF bFnext Y bF;
var_output xF xG Y q R eta muF kFnext omega_next bF;

model;
kG     =  Kbar-kF;                          % market clearing for capital state
Y      =  A*(a+c)*kF + alower*A*kG^alpha;  % aggregate output

bF = q*omega*Kbar - q*kF;

[loglambdaF_next', loglambdaG_next', logauxF_next', q_next']=GDSGE_INTERP_VEC'(kFnext,omega_next');
lambdaF_next' = exp(loglambdaF_next');
lambdaG_next' = exp(loglambdaG_next');
aux_next' = exp(logauxF_next');
kGnext  =  Kbar-kFnext;  % market clearing for capital policy
qbar = GDSGE_MIN(q_next');
bFnext = nbFnext - theta*qbar*kFnext; % Transformation
bGnext = -bFnext;    % market clearing

xF = nxF + c*A*kF;  % consumption of farmer
lambdaF = xF^(-sigma)/(1-eta);
lambdaG = xG^(-sigma);
auxF = (q+A*(a+c)-c*A*eta)*lambdaF;
loglambdaF = log(lambdaF);
loglambdaG = log(lambdaG);
logauxF = log(auxF);

mpk_nextplusq'= (alower*A')*alpha*kGnext^(alpha-1)+q_next';

foc_bondG   = 1 - R*(betaG*GDSGE_EXPECT(lambdaG_next')) / lambdaG;
foc_kG      = q   - betaG*GDSGE_EXPECT(lambdaG_next'*mpk_nextplusq')/lambdaG;
foc_bondF   = 1  -  R*(betaF*GDSGE_EXPECT(lambdaF_next')) / lambdaF   - muF;
foc_kF      = q  -  betaF*GDSGE_EXPECT(aux_next') /lambdaF   - theta*qbar*muF/R;
slack_bF    = muF*nbFnext;
slack_xF    = eta*nxF;
budgetF     = q*kFnext +bFnext/R + xF - A*(a+c)*kF - omega*q*Kbar;
MC_Y = Y - xG - xF;

consis_omega_next' = (q_next'*kFnext + bFnext) - q_next'*omega_next'*Kbar;

equations;
foc_bondG;
foc_kG;
foc_bondF;
foc_kF;
slack_bF;
slack_xF;
budgetF;
MC_Y;
consis_omega_next';
end;
end;

simulate;
num_periods = 1000;
num_samples = 100;
initial kF 0.05;
initial omega 0.01;
initial shock 2;

var_simu xF xG Y q R eta muF bF;
kF'=kFnext;
omega' = omega_next';
end;
```

```
var_state kF omega;
var_shock A;
var_interp loglambdaF_interp loglambdaG_interp logauxF_interp q_interp;
var_policy nxF xG eta kFnext R q nbFnext muF omega_next[3];
model;
kG     =  Kbar-kF;                         % market clearing for capital state
Y      =  A*(a+c)*kF + alower*A*kG^alpha;   % aggregate output

bF = q*omega*Kbar - q*kF;

[loglambdaF_next', loglambdaG_next', logauxF_next', q_next']=GDSGE_INTERP_VEC'(kFnext,omega_next');
lambdaF_next' = exp(loglambdaF_next');
lambdaG_next' = exp(loglambdaG_next');
auxF_next' = exp(logauxF_next');
kGnext  =  Kbar-kFnext;  % market clearing for capital policy
qbar = GDSGE_MIN{q_next'};
bFnext = nbFnext - theta*qbar*kFnext; % Transformaion
consis_omega_next' = (q_next'*kFnext + bFnext) - q_next'*omega_next'*Kbar;
equations;
consis_omega_next';
end;
end;
```

- Notice we set $\{\omega_{t+1}(z_{t+1})\}$ as unknown, and derive $\widetilde{\omega}_{t+1}(z_{t+1}) = \frac{q_{t+1}(z_{t+1})k_{t+1}+b_{t+1}}{q_{t+1}(z_{t+1})\overline{K}}\ \forall z_{t+1}$.
  Consistency equation requires $\omega_{t+1}(z_{t+1}) = \widetilde{\omega}_{t+1}(z_{t+1})\ \forall z_{t+1}$.
- We can derive current debt level by $b_t = q_t\left(\omega_t \overline{K} - k_t\right)$

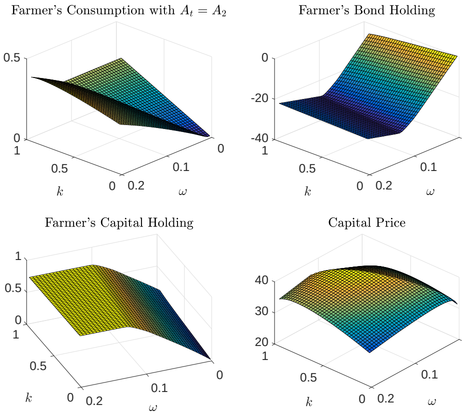
```
var_state kF omega;
var_shock A;
var_interp loglambdaF_interp loglambdaG_interp logauxF_interp q_interp;
var_policy nxF xG eta kFnext R q nbFnext muF omega_next[3];
model;
kG      =  Kbar-kF;                      % market clearing for capital state
Y       =  A*(a+c)*kF + alower*A*kG^alpha;  % aggregate output

bF = q*omega*Kbar - q*kF;

[loglambdaF_next', loglambdaG_next', logauxF_next', q_next']=GDSGE_INTERP_VEC'(kFnext,omega_next');
lambdaF_next' = exp(loglambdaF_next');
lambdaG_next' = exp(loglambdaG_next');
auxF_next' = exp(logauxF_next');
kGnext  =  Kbar-kFnext;  % market clearing for capital policy
qbar = GDSGE_MIN{q_next'};
bFnext = nbFnext - theta*qbar*kFnext; % Transformaion
consis_omega_next' = (q_next'*kFnext + bFnext) - q_next'*omega_next'*Kbar;
equations;
consis_omega_next';
end;
end;
```

- **Trick 1**: Use log of $\left\{\lambda_{t+1}, \lambda'_{t+1}, \xi_{t+1}\right\}$ for interpolation to reduce nonlinearity.
- GDSGE_INTERP_VEC evaluates future variables in **var_interp** once for all.
- As mentioned, GDSGE can accommodate more general dependence on future policy than expectation.

```
var_policy nxF xG eta kFnext R q nbFnext muF omega_next[3];
inbound nxF          0  2;
inbound xG           0  2;
inbound eta          0  1;
inbound kFnext       0  Kbar;
inbound R            0  1.5 adaptive(1.5);
inbound q            0 10    adaptive(1.5);
inbound nbFnext      0 10    adaptive(1.5);
inbound muF          0 1;
inbound omega_next   0 1;
model;
qbar = GDSGE_MIN{q_next'};
bFnext = nbFnext - theta*qbar*kFnext; % Transformaion
bGnext = -bFnext;     % market clearing

xF = nxF + c*A*kF;  % consumption of farmer
slack_bF      = muF*nbFnext;
slack_xF      = eta*nxF;
equations;
slack_bF;
slack_xF;
budgetF;
MC_Y;
consis_omega_next';
end;
end;
```

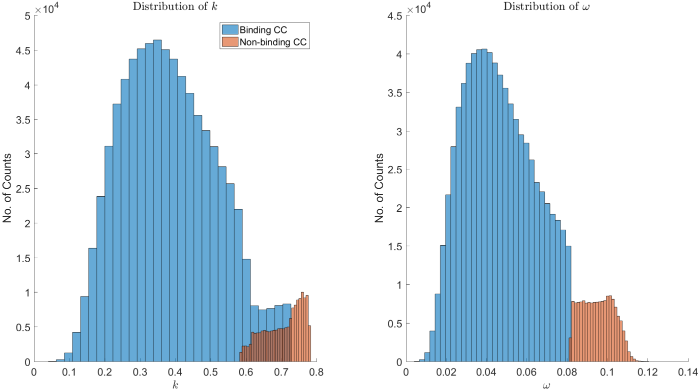
- **Trick 2**: Transform collateral and consumption constraints into $\mathrm{nb}_{t+1} = b_{t+1} + \theta \underline{q}_{t+1} k_{t+1} \geq 0$, and $\mathrm{nx}_t = x_t + cA_t k_t \geq 0$, and include nb_{t+1} and nx_t as unknowns, as in Bianchi2011 and equation (2).
- Also initialize by solving the corresponding last-period problem (**model_init**)

Inspecting the Policy Functions



- **highly nonlinear results across regions**: the collateral constraint binds with low k_t and low ω_t; the consumption constraint binds with high k_t and low ω_t.

Inspecting the Ergodic Distribution



- The ergodic distributions of k and ω confirm our choice of state space.
- The collateral constraint binds with prob. 0.83; and consumption constraint binds with prob. 0.82.

Generalized Impulse Response Function

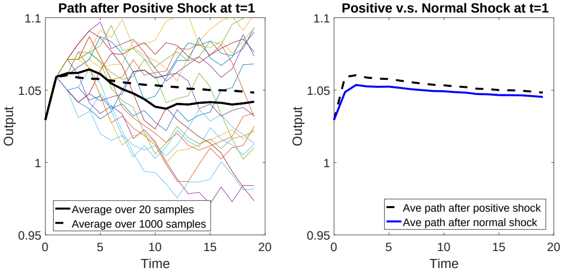
- How to generate IRF with recurrent shock and without steady state?

Generalized Impulse Response Function

- How to generate IRF with recurrent shock and without steady state?
- Assume $A_t \in \left\{\underline{A} < A^* < \overline{A}\right\}$. Pick an initial position (k_0, ω_0, A_0).

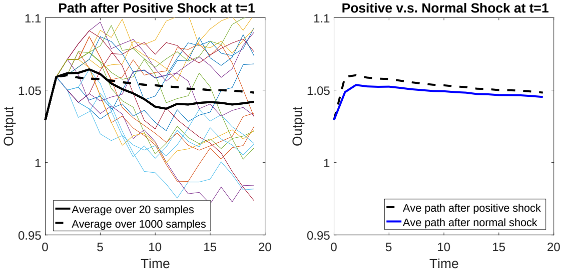
Generalized Impulse Response Function

- How to generate IRF with recurrent shock and without steady state?
- Assume $A_t \in \{\underline{A} < A^* < \overline{A}\}$. Pick an initial position (k_0, ω_0, A_0).
- Step 1: set $A_1 = \overline{A}$ at $t = 1$, simulate forward and compute the average (left figure):



Generalized Impulse Response Function

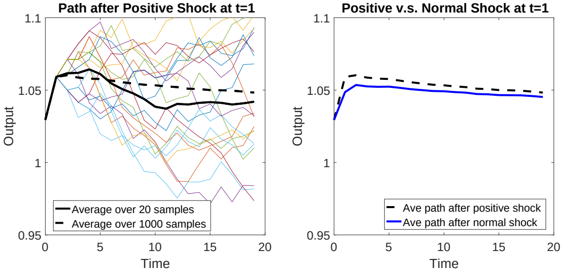
- How to generate IRF with recurrent shock and without steady state?
- Assume $A_t \in \{\underline{A} < A^* < \overline{A}\}$. Pick an initial position (k_0, ω_0, A_0).
- Step 1: set $A_1 = \overline{A}$ at $t = 1$, simulate forward and compute the average (left figure):



- Step 2: set $A_1 = A^*$ at $t = 1$, and compute the average of the simulation (right figure).

Generalized Impulse Response Function

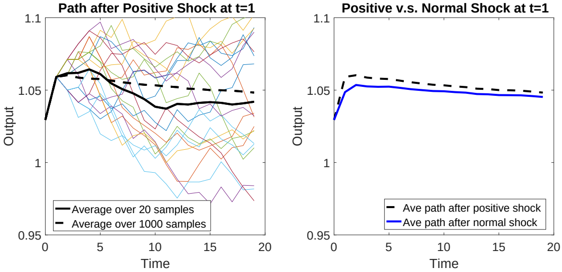
- How to generate IRF with recurrent shock and without steady state?
- Assume $A_t \in \{\underline{A} < A^* < \overline{A}\}$. Pick an initial position (k_0, ω_0, A_0).
- Step 1: set $A_1 = \overline{A}$ at $t = 1$, simulate forward and compute the average (left figure):



- Step 2: set $A_1 = A^*$ at $t = 1$, and compute the average of the simulation (right figure).
- Step 3: Take their difference starting from $t = 1$ as conditional IRF.

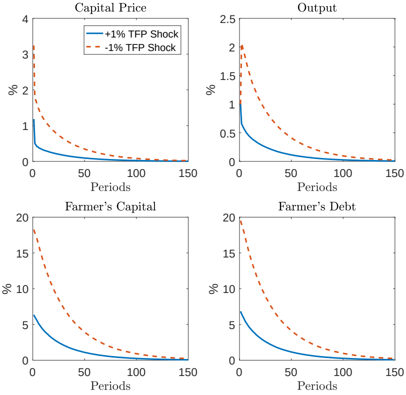
Generalized Impulse Response Function

- How to generate IRF with recurrent shock and without steady state?
- Assume $A_t \in \{\underline{A} < A^* < \overline{A}\}$. Pick an initial position (k_0, ω_0, A_0).
- Step 1: set $A_1 = \overline{A}$ at $t = 1$, simulate forward and compute the average (left figure):



- Step 2: set $A_1 = A^*$ at $t = 1$, and compute the average of the simulation (right figure).
- Step 3: Take their difference starting from $t = 1$ as conditional IRF.
- Step 4: Average the conditional IRF over the ergodic distribution for unconditional IRF.

Generalized Impulse Response Function



- The IRFs are asymmetric and persistent, although the TFP shocks are symmetric and temporary, thanks to collateral constraint and market incompleteness.

General Framework: State with Implicit Law of Motion

$$\boldsymbol{F}\left(s, x, z, \left\{s'(z'), \mathcal{P}^{(n)}\left(z', s'(z')\right)\right\}_{z' \in \mathcal{Z}}\right) = 0.$$

- Question: How to evaluate the transition to future endogenous states $s'(z')$?
- Some admit explicit transition, as in the RBC and Bianchi example
  - s' is an explicit function of **var_shock**, **var_state** and **var_policy**
  - Consistency equation is trivial here since s' does not depend on z'

General Framework: State with Implicit Law of Motion

$$\boldsymbol{F}\left(s, x, z, \left\{s'(z'), \mathcal{P}^{(n)}\left(z', s'(z')\right)\right\}_{z' \in \mathcal{Z}}\right) = 0.$$

- Question: How to evaluate the transition to future endogenous states $s'(z')$?
- Some admit explicit transition, as in the RBC and Bianchi example
  - s' is an explicit function of **var_shock**, **var_state** and **var_policy**
  - Consistency equation is trivial here since s' does not depend on z'
- It becomes involved with endogneous state (e.g., ω_t here)
  - the transition of some endogenous states $\bar{\bar{s}}$ satisfies

$$0 = \bar{\bar{g}}\left(s, x, z, \bar{\bar{s}}'(z'), x'(z'), z'\right),$$

for some non-trivial function $\bar{\bar{g}}$.

General Framework: State with Implicit Law of Motion

$$\boldsymbol{F}\left(s, x, z, \left\{s'(z'), \mathcal{P}^{(n)}\left(z', s'(z')\right)\right\}_{z' \in \mathcal{Z}}\right) = 0.$$

- Question: How to evaluate the transition to future endogenous states $s'(z')$?
- Some admit explicit transition, as in the RBC and Bianchi example
  - s' is an explicit function of **var_shock**, **var_state** and **var_policy**
  - Consistency equation is trivial here since s' does not depend on z'
- It becomes involved with endogneous state (e.g., ω_t here)
  - the transition of some endogenous states $\bar{\bar{s}}$ satisfies

    $$0 = \bar{\bar{g}}\left(s, x, z, \bar{\bar{s}}'(z'), x'(z'), z'\right),$$

    for some non-trivial function $\bar{\bar{g}}$.
  - **Our solution**: include $\bar{\bar{s}}'(z'), \forall z'$ as unknowns and $\bar{\bar{g}}$ in the equation system
  - Kubler and Schmedders(03), and Elenev et al.(16) handle this differently. See an example of the method in Elenev et al.(16) here.
- Consistency equation: the key innovation of the algorithm that enables design of the toolbox

Advice on Using GDSGE and Conclusion

Other Examples on www.gdsge.com

- GDSGE offers great flexibility. Check other examples on our website.
  1. RBC with Irreversible Investment: how to introduce a continous exogenous shock process (e.g. **AR(1)**)
  2. Heaton and Lucas (1996):
     (i) Evaluate the accuracy of solutions
     (ii) Using consumption share (instead of wealth share) as endogenous state
  3. Guvenen (2009): use one solved equilibrium as initial guess for another one
  4. Bianchi (2011): use adaptive sparse grid method
  5. Barro et al. (2017): deal with model with extremly high curvature (risk aversion coefficient=100)
  6. Cao and Nie (2017): different system of equations at different collocation points
  7. Cao (2018): beliefs heterogeneity
  8. Heterogenous-agent model: Huggett(97) with transitional dynamics, and Krusell and Smith(98) with aggregate shocks

Some Advice on Using GDSGE

1. Start by modifying the existing examples first. For example:
    1.1 **two-agent models:** KM (1997), Heaton and Lucas (1996), Cao and Nie (2017), Cao (2018)
    1.2 **open economy models:** Bianchi (2011), Mendoza (2010)
    1.3 **portfolio choice and asset pricing:** Heaton and Lucas (1996), Guvenen (2009)
    1.4 **rare disasters:** Barro et.al (2017)
    1.5 **Heterogenous-agent models:** Huggett (1997), Krusell and Smith (1998)
    1.6 **bubble in backward iteration:** Brumm et al (2015)

Some Advice on Using GDSGE

1. Start by modifying the existing examples first. For example:
   1.1 **two-agent models:** KM (1997), Heaton and Lucas (1996), Cao and Nie (2017), Cao (2018)
   1.2 **open economy models:** Bianchi (2011), Mendoza (2010)
   1.3 **portfolio choice and asset pricing:** Heaton and Lucas (1996), Guvenen (2009)
   1.4 **rare disasters:** Barro et.al (2017)
   1.5 **Heterogenous-agent models:** Huggett (1997), Krusell and Smith (1998)
   1.6 **bubble in backward iteration:** Brumm et al (2015)

2. Crucial to initialize the **var_interp** properly for the algorithm to work. Initializing with a last-period problem in finite-horizon economies usually works robustly.

Some Advice on Using GDSGE

1. Start by modifying the existing examples first. For example:
   1.1 **two-agent models:** KM (1997), Heaton and Lucas (1996), Cao and Nie (2017), Cao (2018)
   1.2 **open economy models:** Bianchi (2011), Mendoza (2010)
   1.3 **portfolio choice and asset pricing:** Heaton and Lucas (1996), Guvenen (2009)
   1.4 **rare disasters:** Barro et.al (2017)
   1.5 **Heterogenous-agent models:** Huggett (1997), Krusell and Smith (1998)
   1.6 **bubble in backward iteration:** Brumm et al (2015)

2. Crucial to initialize the **var_interp** properly for the algorithm to work. Initializing with a last-period problem in finite-horizon economies usually works robustly.

3. input unit-free Euler equations: $\beta\mathbb{E}_t\left(R_t c_{t+1}^{-\sigma}/c_t^{-\sigma}\right) - 1 = 0$, instead of $c_t^{-\sigma} - \beta\mathbb{E}_t\left(R_t c_{t+1}^{-\sigma}\right) = 0$. Also normalize Lagrangian multipliers to bound their values.

Some Advice on Using GDSGE

1. Start by modifying the existing examples first. For example:
   1.1 **two-agent models:** KM (1997), Heaton and Lucas (1996), Cao and Nie (2017), Cao (2018)
   1.2 **open economy models:** Bianchi (2011), Mendoza (2010)
   1.3 **portfolio choice and asset pricing:** Heaton and Lucas (1996), Guvenen (2009)
   1.4 **rare disasters:** Barro et.al (2017)
   1.5 **Heterogenous-agent models:** Huggett (1997), Krusell and Smith (1998)
   1.6 **bubble in backward iteration:** Brumm et al (2015)

2. Crucial to initialize the **var_interp** properly for the algorithm to work. Initializing with a last-period problem in finite-horizon economies usually works robustly.

3. input unit-free Euler equations: $\beta \mathbb{E}_t \left(R_t c_{t+1}^{-\sigma} / c_t^{-\sigma} \right) - 1 = 0$, instead of $c_t^{-\sigma} - \beta \mathbb{E}_t \left(R_t c_{t+1}^{-\sigma} \right) = 0$. Also normalize Lagrangian multipliers to bound their values.

4. debug: use **mex_modname** function in **iter_modname.m** to debug.

Interface of the mex File

- The compiled mex file contains the libraries for the actual calculations
- The mex file is called by by the iter_ and simulate_ file, e.g. in RBC:

```
[GDSGE_SOL,GDSGE_F,GDSGE_AUX,GDSGE_EQVAL,GDSGE_OPT_INFO] = ...
mex_modname(GDSGE_SOL,GDSGE_LB,GDSGE_UB,GDSGE_DATA,...
GDSGE_SKIP,GDSGE_F,GDSGE_AUX,GDSGE_EQVAL);
```

- **Input**: vectors with information for all problems across collocation points
  - GDSGE_SOL: the (vector of) initial points of **var_policy** for solving equations
  - GDSGE_LB / GDSGE_UB: lower and upper bounds of **var_policy** to search
  - GDSGE_DATA: parameters and states that characterize problems at each collocation point

```
[GDSGE_SOL,GDSGE_F,GDSGE_AUX,GDSGE_EQVAL,GDSGE_OPT_INFO] = ...
mex_modname(GDSGE_SOL,GDSGE_LB,GDSGE_UB,GDSGE_DATA,...
GDSGE_SKIP,GDSGE_F,GDSGE_AUX,GDSGE_EQVAL);
```

- **Output**: vectors of output from equation solving across collocation points
  - GDSGE_SOL: **var_policy** returned
  - GDSGE_F: max absolute residual
  - GDSGE_AUX: **var_aux** evaluated at returned var_policy
  - GDSGE_EQVAL: residual of each equation at returned var_policy
  - GDSGE_OPT_INFO: information returned from equation solving procedures

Conclusion

- A framework and toolbox that solves GDSGE with global methods robustly and efficiently.
- Any models with short-run equilibrium conditions represented by equations fit in the framework, covering classical and state-of-art models in macro, IF, macro finance and asset pricing
- Key innovation: consistency equations to deal with endogenous states with implicit laws of motion
- Can solve models with discrete choice (e.g., sovereign default) by smoothing out discrete choices
- Comments and contributions welcome! gdsge.cln2020@gmail.com