Comparison of Global Solution Methods to a Zero Lower Bound Model

Emily Martell[1]

College of William & Mary

April 29, 2019

[1]Thank you to Professor Throckmorton for advising and providing materials for this project. Thank you to Professor Campbell, Professor Han, Professor Rolek, and Professor Throckmorton for serving on my committee.

Introduction

- As a result of the global financial crisis of 2007-9 and the subsequent recession, central banks lowered their policy rate to its zero lower bound (ZLB)
- The ZLB now holds for a significant portion of historic data for the US, Japan, and the Euro Area
- The ZLB introduces a kink in the central bank's policy rule and calls into question linear estimation methods
- Responses in the literature to this nonlinearity include:
  1. Failing to incorporate ZLB period data
  2. Estimating linear models on the entire data set
  3. Estimating a piecewise linear version of the nonlinear model (e.g., Guerrieri and Iacoviello, 2017)
  4. Estimating fully nonlinear models that treats the ZLB as an occasionally binding constraint (e.g., Gust et al., 2017; Plante et al., 2018; Richter and Throckmorton, 2016).

Introduction

- As a result of the global financial crisis of 2007-9 and the subsequent recession, central banks lowered their policy rate to its zero lower bound (ZLB)
- The ZLB now holds for a significant portion of historic data for the US, Japan, and the Euro Area
- The ZLB introduces a kink in the central bank's policy rule and calls into question linear estimation methods
- Responses in the literature to this nonlinearity include:
  1. Failing to incorporate ZLB period data
  2. Estimating linear models on the entire data set
  3. Estimating a piecewise linear version of the nonlinear model (e.g., Guerrieri and Iacoviello, 2017)
  4. Estimating fully nonlinear models that treats the ZLB as an occasionally binding constraint (e.g., Gust et al., 2017; Plante et al., 2018; Richter and Throckmorton, 2016).

Introduction

- As a result of the global financial crisis of 2007-9 and the subsequent recession, central banks lowered their policy rate to its zero lower bound (ZLB)
- The ZLB now holds for a significant portion of historic data for the US, Japan, and the Euro Area
- The ZLB introduces a kink in the central bank's policy rule and calls into question linear estimation methods
- Responses in the literature to this nonlinearity include:
  1. Failing to incorporate ZLB period data
  2. Estimating linear models on the entire data set
  3. Estimating a piecewise linear version of the nonlinear model (e.g., Guerrieri and Iacoviello, 2017)
  4. Estimating fully nonlinear models that treats the ZLB as an occasionally binding constraint (e.g., Gust et al., 2017; Plante et al., 2018; Richter and Throckmorton, 2016).

INTRODUCTION

- As a result of the global financial crisis of 2007-9 and the subsequent recession, central banks lowered their policy rate to its zero lower bound (ZLB)
- The ZLB now holds for a significant portion of historic data for the US, Japan, and the Euro Area
- The ZLB introduces a kink in the central bank's policy rule and calls into question linear estimation methods
- Responses in the literature to this nonlinearity include:
  1. Failing to incorporate ZLB period data
  2. Estimating linear models on the entire data set
  3. Estimating a piecewise linear version of the nonlinear model (e.g., Guerrieri and Iacoviello, 2017)
  4. Estimating fully nonlinear models that treats the ZLB as an occasionally binding constraint (e.g., Gust et al., 2017; Plante et al., 2018; Richter and Throckmorton, 2016).

The Model (With Capital)

- The households choose $\{c_t, n_t, b_t, x_t, k_t\}_{t=0}^{\infty}$ to maximize expected lifetime utility given by

$$E_0 \sum_{t=0}^{\infty} \beta[\log(c_t - hc_{t-1}^a) - \chi n_t^{1+\eta}/(1+\eta)],$$

subject to their budget constraint

$$c_t + x_t + b_t/(i_t s_t) = w_t n_t + r_t^k k_{t-1} + b_{t-1}/\pi_t + d_t.$$

- The nominal bond, b, is subject to a risk premium, s, that follows

$$s_t = (1-\rho_s)\bar{s} + \rho_s s_{t-1} + \sigma_s \varepsilon_{s,t},$$

where $\bar{s}$ is the steady-state value.

The Model (With Capital)

- The households choose $\{c_t, n_t, b_t, x_t, k_t\}_{t=0}^{\infty}$ to maximize expected lifetime utility given by

$$E_0 \sum_{t=0}^{\infty} \beta[\log(c_t - hc_{t-1}^a) - \chi n_t^{1+\eta}/(1+\eta)],$$

subject to their budget constraint

$$c_t + x_t + b_t/(i_t s_t) = w_t n_t + r_t^k k_{t-1} + b_{t-1}/\pi_t + d_t.$$

- The nominal bond, b, is subject to a risk premium, s, that follows

$$s_t = (1-\rho_s)\bar{s} + \rho_s s_{t-1} + \sigma_s \varepsilon_{s,t},$$

where $\bar{s}$ is the steady-state value.

HOUSEHOLDS

- Households also face an investment adjustment cost, so the law of motion for capital is given by

$$k_t = (1-\delta)k_{t-1} + x_t(1 - \nu(x_t^g - 1)^2/2),\ 0 \leq \delta \leq 1.$$

- The FOCs to the representative household's constrained optimization problem are

$$\begin{aligned}
\lambda_t &= c_t - hc_{t-1}^a, \\
w_t &= \chi n_t^\eta \lambda_t, \\
1 &= \beta E_t[(\lambda_t/\lambda_{t+1})(s_t i_t/(\bar{\pi}\pi_{t+1}^{gap}))], \\
q_t &= \beta E_t[(\lambda_t/\lambda_{t+1})(r_{t+1}^k + (1-\delta)q_{t+1})], \\
1 &= q_t[1 - \nu(x_t^g - 1)^2/2 - \nu(x_t^g - 1)x_t^g] + \nu\beta\bar{g}E_t[q_{t+1}(\lambda_t/\lambda_{t+1})(x_{t+1}^g)^2(x_{t+1}^g - 1)].
\end{aligned}$$

HOUSEHOLDS

- Households also face an investment adjustment cost, so the law of motion for capital is given by

$$k_t = (1-\delta)k_{t-1} + x_t(1-\nu(x_t^g - 1)^2/2),\ 0 \leq \delta \leq 1.$$

- The FOCs to the representative household's constrained optimization problem are

$$\begin{gathered}
\lambda_t = c_t - hc_{t-1}^a, \\
w_t = \chi n_t^\eta \lambda_t, \\
1 = \beta E_t[(\lambda_t/\lambda_{t+1})(s_t i_t/(\bar{\pi}\pi_{t+1}^{gap}))], \\
q_t = \beta E_t[(\lambda_t/\lambda_{t+1})(r_{t+1}^k + (1-\delta)q_{t+1})], \\
1 = q_t[1-\nu(x_t^g - 1)^2/2 - \nu(x_t^g - 1)x_t^g] + \nu\beta\bar{g}E_t[q_{t+1}(\lambda_t/\lambda_{t+1})(x_{t+1}^g)^2(x_{t+1}^g - 1)].
\end{gathered}$$

Firms

- The production sector consists of a continuum of monopolistically competitive intermediate goods firms and a final goods firm
- Technology is $z_t = g_t z_{t-1}$, which is common across firms
- Deviations from the steady-state growth rate, $\bar{g}$, follow

$$g_t = \bar{g} + \sigma_g \varepsilon_{g,t}.$$

- In symmetric equilibrium, the optimality conditions reduce to

$$\begin{aligned}
y_t &= (k_{t-1})^{\alpha}(z_t n_t)^{1-\alpha}, \\
w_t &= (1-\alpha) mc_t y_t / n_t, \\
r_t^k &= \alpha mc_t y_t / k_{t-1}, \\
\varphi(\pi_t^{gap} - 1)\pi_t^{gap} &= 1 - \theta + \theta mc_t + \beta\varphi E_t[(\lambda_t/\lambda_{t+1})(\pi_{t+1}^{gap} - 1)\pi_{t+1}^{gap}(y_{t+1}/y_t)].
\end{aligned}$$

FIRMS

- The production sector consists of a continuum of monopolistically competitive intermediate goods firms and a final goods firm
- Technology is $z_t = g_t z_{t-1}$, which is common across firms
- Deviations from the steady-state growth rate, $\bar{g}$, follow

$$g_t = \bar{g} + \sigma_g \varepsilon_{g,t}.$$

- In symmetric equilibrium, the optimality conditions reduce to

$$y_t = (k_{t-1})^{\alpha}(z_t n_t)^{1-\alpha},$$
$$w_t = (1-\alpha) mc_t y_t / n_t,$$
$$r_t^k = \alpha mc_t y_t / k_{t-1},$$
$$\varphi(\pi_t^{gap} - 1)\pi_t^{gap} = 1 - \theta + \theta mc_t + \beta\varphi E_t[(\lambda_t/\lambda_{t+1})(\pi_{t+1}^{gap} - 1)\pi_{t+1}^{gap}(y_{t+1}/y_t)].$$

FIRMS

- The production sector consists of a continuum of monopolistically competitive intermediate goods firms and a final goods firm
- Technology is $z_t = g_t z_{t-1}$, which is common across firms
- Deviations from the steady-state growth rate, $\bar{g}$, follow

$$g_t = \bar{g} + \sigma_g \varepsilon_{g,t}.$$

- In symmetric equilibrium, the optimality conditions reduce to

$$\begin{aligned} y_t &= (k_{t-1})^{\alpha}(z_t n_t)^{1-\alpha}, \\ w_t &= (1-\alpha) mc_t y_t / n_t, \\ r_t^k &= \alpha mc_t y_t / k_{t-1}, \\ \varphi(\pi_t^{gap} - 1)\pi_t^{gap} &= 1 - \theta + \theta mc_t + \beta\varphi E_t[(\lambda_t/\lambda_{t+1})(\pi_{t+1}^{gap} - 1)\pi_{t+1}^{gap}(y_{t+1}/y_t)]. \end{aligned}$$

Firms

- The production sector consists of a continuum of monopolistically competitive intermediate goods firms and a final goods firm
- Technology is $z_t = g_t z_{t-1}$, which is common across firms
- Deviations from the steady-state growth rate, $\bar{g}$, follow

$$g_t = \bar{g} + \sigma_g \varepsilon_{g,t}.$$

- In symmetric equilibrium, the optimality conditions reduce to

$$\begin{gathered}
y_t = (k_{t-1})^\alpha (z_t n_t)^{1-\alpha}, \\
w_t = (1-\alpha) mc_t y_t / n_t, \\
r_t^k = \alpha mc_t y_t / k_{t-1}, \\
\varphi(\pi_t^{gap} - 1)\pi_t^{gap} = 1 - \theta + \theta mc_t + \beta\varphi E_t[(\lambda_t/\lambda_{t+1})(\pi_{t+1}^{gap} - 1)\pi_{t+1}^{gap}(y_{t+1}/y_t)].
\end{gathered}$$

Monetary Policy

- The central bank sets the gross nominal interest rate, i, according to

$$i_t = \max\{1, i_t^n\},$$
$$i_t^n = (i_{t-1}^n)^{\rho_i}(\bar{\imath}(\pi_t^{gap})^{\phi_\pi}(y_t^g)^{\phi_y})^{1-\rho_i}\exp(\sigma_i\varepsilon_{i,t}).$$

- A more negative net notional rate indicates that the central bank is more constrained

Monetary Policy

- The central bank sets the gross nominal interest rate, i, according to

$$i_t = \max\{1, i_t^n\},$$
$$i_t^n = (i_{t-1}^n)^{\rho_i} (\bar{\imath}(\pi_t^{gap})^{\phi_\pi} (y_t^g)^{\phi_y})^{1-\rho_i} \exp(\sigma_i \varepsilon_{i,t}).$$

- A more negative net notional rate indicates that the central bank is more constrained

Competitive Equilibrium

- The aggregate resource constraint and real GDP definition are given by:

$$c_t + x_t = y_t^{gdp}$$
$$y_t^{gdp} = [1 - \varphi(\pi_t^{gap} - 1)^2/2]y_t$$

- The model does not have a steady-state due to the unit root in technology, z_t. Therefore, we define the variables with a trend in terms of technology (i.e., $\tilde{x}_t \equiv x_t/z_t$)

Competitive Equilibrium

- The aggregate resource constraint and real GDP definition are given by:

$$c_t + x_t = y_t^{gdp}$$
$$y_t^{gdp} = [1 - \varphi(\pi_t^{gap} - 1)^2/2]y_t$$

- The model does not have a steady-state due to the unit root in technology, z_t. Therefore, we define the variables with a trend in terms of technology (i.e., $\tilde{x}_t \equiv x_t/z_t$)

Competitive Equilibrium

- A competitive equilibrium consists of sequences of
  1. quantities, $\{\tilde{c}_t, \tilde{y}_t, \tilde{y}_t^{gdp}, x_t^g, y_t^g, n_t, \tilde{k}_t, \tilde{x}_t\}_{t=0}^{\infty}$
  2. prices, $\{\tilde{w}_t, i_t, i_t^n, \pi_t, \tilde{\lambda}_t, q_t, r_t^k, mc_t\}_{t=0}^{\infty}$
  3. exogenous variables, $\{s_t, g_t\}_{t=0}^{\infty}$
- that satisfy the detrended equilibrium system, given
  1. the initial conditions, $\{\tilde{c}_{-1}, i_{-1}^n, \tilde{k}_{-1}, \tilde{x}_{-1}, \tilde{w}_{-1}, s_0, g_0, \varepsilon_{i,0}\}$
  2. three sequences of shocks, $\{\varepsilon_{g,t}, \varepsilon_{s,t}, \varepsilon_{i,t}\}_{t=1}^{\infty}$

Competitive Equilibrium

- A competitive equilibrium consists of sequences of
  1. quantities, $\{\tilde{c}_t, \tilde{y}_t, \tilde{y}_t^{gdp}, x_t^g, y_t^g, n_t, \tilde{k}_t, \tilde{x}_t\}_{t=0}^{\infty}$
  2. prices, $\{\tilde{w}_t, i_t, i_t^n, \pi_t, \tilde{\lambda}_t, q_t, r_t^k, mc_t\}_{t=0}^{\infty}$
  3. exogenous variables, $\{s_t, g_t\}_{t=0}^{\infty}$
- that satisfy the detrended equilibrium system, given
  1. the initial conditions, $\{\tilde{c}_{-1}, i_{-1}^n, \tilde{k}_{-1}, \tilde{x}_{-1}, \tilde{w}_{-1}, s_0, g_0, \varepsilon_{i,0}\}$
  2. three sequences of shocks, $\{\varepsilon_{g,t}, \varepsilon_{s,t}, \varepsilon_{i,t}\}_{t=1}^{\infty}$

Parameter Values

Subjective Discount Factor	β	0.9949	Rotemberg Price Adjustment Cost	φ	100
Frisch Labor Supply Elasticity	$1/\eta$	3	Inflation Gap Response	ϕ_π	2.0
Price Elasticity of Substitution	θ	6	Output Growth Gap Response	ϕ_y	0.5
Steady-State Labor Hours	$\bar{n}$	$1/3$	Habit Persistence	h	0.80
Steady-State Risk Premium	$\bar{s}$	1.0058	Risk Premium Persistence	ρ_s	0.80
Steady-State Growth Rate	$\bar{g}$	1.0034	Notional Rate Persistence	ρ_i	0.80
Steady-State Inflation Rate	$\bar{\pi}$	1.0053	Technology Growth Shock SD	σ_g	0.005
Capital Share of Income	α	0.35	Risk Premium Shock SD	σ_s	0.0085
Capital Depreciation Rate	δ	0.025	Notional Interest Rate Shock SD	σ_i	0.002
Investment Adjustment Cost	ν	4			

- Parameters are from Atkinson et al. (2019), and are chosen to be characteristic of U.S. data
- For the model without capital, $\sigma_s = 0.006$

Parameter Values

Subjective Discount Factor	β	0.9949	Rotemberg Price Adjustment Cost	φ	100
Frisch Labor Supply Elasticity	$1/\eta$	3	Inflation Gap Response	ϕ_π	2.0
Price Elasticity of Substitution	θ	6	Output Growth Gap Response	ϕ_y	0.5
Steady-State Labor Hours	$\bar{n}$	$1/3$	Habit Persistence	h	0.80
Steady-State Risk Premium	$\bar{s}$	1.0058	Risk Premium Persistence	ρ_s	0.80
Steady-State Growth Rate	$\bar{g}$	1.0034	Notional Rate Persistence	ρ_i	0.80
Steady-State Inflation Rate	$\bar{\pi}$	1.0053	Technology Growth Shock SD	σ_g	0.005
Capital Share of Income	α	0.35	Risk Premium Shock SD	σ_s	0.0085
Capital Depreciation Rate	δ	0.025	Notional Interest Rate Shock SD	σ_i	0.002
Investment Adjustment Cost	ν	4			

- Parameters are from Atkinson et al. (2019), and are chosen to be characteristic of U.S. data
- For the model without capital, $\sigma_s = 0.006$

Parameter Values

Subjective Discount Factor	β	0.9949	Rotemberg Price Adjustment Cost	φ	100
Frisch Labor Supply Elasticity	$1/\eta$	3	Inflation Gap Response	ϕ_π	2.0
Price Elasticity of Substitution	θ	6	Output Growth Gap Response	ϕ_y	0.5
Steady-State Labor Hours	$\bar{n}$	$1/3$	Habit Persistence	h	0.80
Steady-State Risk Premium	$\bar{s}$	1.0058	Risk Premium Persistence	ρ_s	0.80
Steady-State Growth Rate	$\bar{g}$	1.0034	Notional Rate Persistence	ρ_i	0.80
Steady-State Inflation Rate	$\bar{\pi}$	1.0053	Technology Growth Shock SD	σ_g	0.005
Capital Share of Income	α	0.35	Risk Premium Shock SD	σ_s	0.0085
Capital Depreciation Rate	δ	0.025	Notional Interest Rate Shock SD	σ_i	0.002
Investment Adjustment Cost	ν	4			

- Parameters are from Atkinson et al. (2019), and are chosen to be characteristic of U.S. data
- For the model without capital, $\sigma_s = 0.006$

Nonlinear Solution Methods

- As it is important to incorporate all historic data and yield accurate parameter values and predictions, the path forward seems to be in fully nonlinear models

- Solving nonlinear models with projection methods:
  1. Policy function iteration with linear interpolation on an evenly spaced grid (e.g., Plante et al., 2018; Richter and Throckmorton, 2016).
  2. Alternate approximating functions: regime-indexed policy functions (Gust et al., 2017), piecewise smooth policy functions (Aruoba et al. 2018)
  3. Alternate grid construction and policy function evaluation method: Smolyak method with Chebyshev polynomials (e.g., Gust et al., 2017; Fernańdez-Villaverde et al. 2015; Aruoba et al., 2018)

Nonlinear Solution Methods

- As it is important to incorporate all historic data and yield accurate parameter values and predictions, the path forward seems to be in fully nonlinear models
- Solving nonlinear models with projection methods:
  1. Policy function iteration with linear interpolation on an evenly spaced grid (e.g., Plante et al., 2018; Richter and Throckmorton, 2016).
  2. Alternate approximating functions: regime-indexed policy functions (Gust et al., 2017), piecewise smooth policy functions (Aruoba et al. 2018)
  3. Alternate grid construction and policy function evaluation method: Smolyak method with Chebyshev polynomials (e.g., Gust et al., 2017; Fernańdez-Villaverde et al. 2015; Aruoba et al., 2018)

Nonlinear Solution Methods

- As it is important to incorporate all historic data and yield accurate parameter values and predictions, the path forward seems to be in fully nonlinear models
- Solving nonlinear models with projection methods:
    1. Policy function iteration with linear interpolation on an evenly spaced grid (e.g., Plante et al., 2018; Richter and Throckmorton, 2016).
    2. Alternate approximating functions: regime-indexed policy functions (Gust et al., 2017), piecewise smooth policy functions (Aruoba et al. 2018)
    3. Alternate grid construction and policy function evaluation method: Smolyak method with Chebyshev polynomials (e.g., Gust et al., 2017; Fernańdez-Villaverde et al. 2015; Aruoba et al., 2018)

Two Solution Methods

	Atkinson et al. (2019)	**Gust et al. (2017)**
Discretization Method	Evenly-spaced grid points	Smolyak Method
Policy Function Evaluation	Linear Interpolation	Chebyshev Polynomials
Integration Method	Rouwenhorst	Gauss-Hermite quadrature

- Evenly-spaced grid points work well with linear interpolation
- The Smolyak method is optimal for Chebyshev polynomials
- The Rouwenhorst (1995) method improves approximation on exogenous dimensions when the driving processes are autoregressive

Two Solution Methods

	Atkinson et al. (2019)	**Gust et al. (2017)**
Discretization Method	Evenly-spaced grid points	Smolyak Method
Policy Function Evaluation	Linear Interpolation	Chebyshev Polynomials
Integration Method	Rouwenhorst	Gauss-Hermite quadrature

- Evenly-spaced grid points work well with linear interpolation
- The Smolyak method is optimal for Chebyshev polynomials
- The Rouwenhorst (1995) method improves approximation on exogenous dimensions when the driving processes are autoregressive

Two Solution Methods

	Atkinson et al. (2019)	**Gust et al. (2017)**
Discretization Method	Evenly-spaced grid points	Smolyak Method
Policy Function Evaluation	Linear Interpolation	Chebyshev Polynomials
Integration Method	Rouwenhorst	Gauss-Hermite quadrature

- Evenly-spaced grid points work well with linear interpolation
- The Smolyak method is optimal for Chebyshev polynomials
- The Rouwenhorst (1995) method improves approximation on exogenous dimensions when the driving processes are autoregressive

Two Solution Methods

	Atkinson et al. (2019)	**Gust et al. (2017)**
Discretization Method	Evenly-spaced grid points	Smolyak Method
Policy Function Evaluation	Linear Interpolation	Chebyshev Polynomials
Integration Method	Rouwenhorst	Gauss-Hermite quadrature

- Evenly-spaced grid points work well with linear interpolation
- The Smolyak method is optimal for Chebyshev polynomials
- The Rouwenhorst (1995) method improves approximation on exogenous dimensions when the driving processes are autoregressive

Atkinson et al. (2019)

- Time iteration with linear interpolation
- Directly approximates policy functions
- State space features evenly-spaced nodes
- Exogenous variables are approximated with the Markov chain from Rouwenhorst (1995)
    - Only requires interpolation along the dimensions of the endogenous state variables

Atkinson et al. (2019)

- Time iteration with linear interpolation
- Directly approximates policy functions
- State space features evenly-spaced nodes
- Exogenous variables are approximated with the Markov chain from Rouwenhorst (1995)
  - Only requires interpolation along the dimensions of the endogenous state variables

Atkinson et al. (2019)

- Time iteration with linear interpolation
- Directly approximates policy functions
- State space features evenly-spaced nodes
- Exogenous variables are approximated with the Markov chain from Rouwenhorst (1995)
  - Only requires interpolation along the dimensions of the endogenous state variables

Atkinson et al. (2019)

- Time iteration with linear interpolation
- Directly approximates policy functions
- State space features evenly-spaced nodes
- Exogenous variables are approximated with the Markov chain from Rouwenhorst (1995)
  - Only requires interpolation along the dimensions of the endogenous state variables

GUST ET AL. (2017)

- Approximates policy functions using an anisotropic Smolyak method with Chebyshev polynomials
- Approximates exogenous state variables using Gauss-Hermite quadrature
- Instead of directly computing the policy functions, they estimate functions at and away from the ZLB, building on Christiano and Fisher (2000)
  - Policy functions feature a kink or non-differentiability at the ZLB and regime-indexing the policy functions will yield smoother functions
  - Low-order Chebyshev polynomials perform better on smoother functions and are inexpensive to compute

Gust et al. (2017)

- Approximates policy functions using an anisotropic Smolyak method with Chebyshev polynomials
- Approximates exogenous state variables using Gauss-Hermite quadrature
- Instead of directly computing the policy functions, they estimate functions at and away from the ZLB, building on Christiano and Fisher (2000)
  - Policy functions feature a kink or non-differentiability at the ZLB and regime-indexing the policy functions will yield smoother functions
  - Low-order Chebyshev polynomials perform better on smoother functions and are inexpensive to compute

Gust et al. (2017)

- Approximates policy functions using an anisotropic Smolyak method with Chebyshev polynomials
- Approximates exogenous state variables using Gauss-Hermite quadrature
- Instead of directly computing the policy functions, they estimate functions at and away from the ZLB, building on Christiano and Fisher (2000)
    - Policy functions feature a kink or non-differentiability at the ZLB and regime-indexing the policy functions will yield smoother functions
    - Low-order Chebyshev polynomials perform better on smoother functions and are inexpensive to compute

This Project

- In this project, we consider how splitting up the policy functions conditional on the ZLB impacts the speed and accuracy of the solution of a nonlinear model
- We use policy function iteration on a fixed point with linear interpolation
- Atkinson et al. (2019) directly computes the policy functions; Gust et al. (2017) indirectly computes the policy functions through regime-indexed policy functions
- Solution algorithm based on Atkinson et al. (2019) denoted henceforth as ART; solution algorithm based on Gust et al. (2017) denoted henceforth as GHLS

This Project

- In this project, we consider how splitting up the policy functions conditional on the ZLB impacts the speed and accuracy of the solution of a nonlinear model
- We use policy function iteration on a fixed point with linear interpolation
- Atkinson et al. (2019) directly computes the policy functions; Gust et al. (2017) indirectly computes the policy functions through regime-indexed policy functions
- Solution algorithm based on Atkinson et al. (2019) denoted henceforth as ART; solution algorithm based on Gust et al. (2017) denoted henceforth as GHLS

This Project

- In this project, we consider how splitting up the policy functions conditional on the ZLB impacts the speed and accuracy of the solution of a nonlinear model
- We use policy function iteration on a fixed point with linear interpolation
- Atkinson et al. (2019) directly computes the policy functions; Gust et al. (2017) indirectly computes the policy functions through regime-indexed policy functions
- Solution algorithm based on Atkinson et al. (2019) denoted henceforth as ART; solution algorithm based on Gust et al. (2017) denoted henceforth as GHLS

This Project

- In this project, we consider how splitting up the policy functions conditional on the ZLB impacts the speed and accuracy of the solution of a nonlinear model
- We use policy function iteration on a fixed point with linear interpolation
- Atkinson et al. (2019) directly computes the policy functions; Gust et al. (2017) indirectly computes the policy functions through regime-indexed policy functions
- Solution algorithm based on Atkinson et al. (2019) denoted henceforth as ART; solution algorithm based on Gust et al. (2017) denoted henceforth as GHLS

Motivation for Regime-Indexed Policy Functions

- The interest rate enters directly in the consumption Euler equation

$$1 = E_t[\beta(c_t/c_{t+1})(s_t i_t/\pi_t)],$$

  where E_t is the expectation operator, $0 < \beta < 1$ is the discount factor, and c_t, s_t, i_t, and π_t are consumption, the risk premium, the interest rate, and inflation at time t.
    - $\beta(c_t/c_{t+1})$ is a stochastic discount factor used to value future real income
    - $s_t i_t/\pi_t$ is a real interest rate on a one-period bond
- At the ZLB, the presence of the interest rate creates a nonlinearity in the consumption Euler equation
- c_t depends directly on the interest rate

Motivation for Regime-Indexed Policy Functions

- The interest rate enters directly in the consumption Euler equation

$$1 = E_t[\beta(c_t/c_{t+1})(s_t i_t/\pi_t)],$$

  where E_t is the expectation operator, $0 < \beta < 1$ is the discount factor, and c_t, s_t, i_t, and π_t are consumption, the risk premium, the interest rate, and inflation at time t.
    - $\beta(c_t/c_{t+1})$ is a stochastic discount factor used to value future real income
    - $s_t i_t/\pi_t$ is a real interest rate on a one-period bond
- At the ZLB, the presence of the interest rate creates a nonlinearity in the consumption Euler equation
- c_t depends directly on the interest rate

Motivation for Regime-Indexed Policy Functions

- The interest rate enters directly in the consumption Euler equation

  $$1 = E_t[\beta(c_t/c_{t+1})(s_t i_t/\pi_t)],$$

  where E_t is the expectation operator, $0 < \beta < 1$ is the discount factor, and c_t, s_t, i_t, and π_t are consumption, the risk premium, the interest rate, and inflation at time t.
  - $\beta(c_t/c_{t+1})$ is a stochastic discount factor used to value future real income
  - $s_t i_t/\pi_t$ is a real interest rate on a one-period bond
- At the ZLB, the presence of the interest rate creates a nonlinearity in the consumption Euler equation
- c_t depends directly on the interest rate

Our Solution Algorithm

- Construct the state space with evenly-spaced nodes and approximate exogenous state variables with Rouwenhorst (1995)
- Obtain initial conjectures for a set of policy functions from the log-linear solution
- Using a fixed point iteration scheme, linearly interpolate and numerically integrate the policy functions each step
  - For ART portion, we update a single policy function, and for the GHLS portion we update regime-indexed policy functions
- The algorithm converges once the maximum distance between successive guesses of policy functions falls below a convergence criterion

Our Solution Algorithm

- Construct the state space with evenly-spaced nodes and approximate exogenous state variables with Rouwenhorst (1995)
- Obtain initial conjectures for a set of policy functions from the log-linear solution
- Using a fixed point iteration scheme, linearly interpolate and numerically integrate the policy functions each step
  - For ART portion, we update a single policy function, and for the GHLS portion we update regime-indexed policy functions
- The algorithm converges once the maximum distance between successive guesses of policy functions falls below a convergence criterion

Our Solution Algorithm

- Construct the state space with evenly-spaced nodes and approximate exogenous state variables with Rouwenhorst (1995)
- Obtain initial conjectures for a set of policy functions from the log-linear solution
- Using a fixed point iteration scheme, linearly interpolate and numerically integrate the policy functions each step
  - For ART portion, we update a single policy function, and for the GHLS portion we update regime-indexed policy functions
- The algorithm converges once the maximum distance between successive guesses of policy functions falls below a convergence criterion

Our Solution Algorithm

- Construct the state space with evenly-spaced nodes and approximate exogenous state variables with Rouwenhorst (1995)
- Obtain initial conjectures for a set of policy functions from the log-linear solution
- Using a fixed point iteration scheme, linearly interpolate and numerically integrate the policy functions each step
  - For ART portion, we update a single policy function, and for the GHLS portion we update regime-indexed policy functions
- The algorithm converges once the maximum distance between successive guesses of policy functions falls below a convergence criterion

Our Solution Algorithm

- Construct the state space with evenly-spaced nodes and approximate exogenous state variables with Rouwenhorst (1995)
- Obtain initial conjectures for a set of policy functions from the log-linear solution
- Using a fixed point iteration scheme, linearly interpolate and numerically integrate the policy functions each step
    - For ART portion, we update a single policy function, and for the GHLS portion we update regime-indexed policy functions
- The algorithm converges once the maximum distance between successive guesses of policy functions falls below a convergence criterion

Discretized State Space (No Capital)

- State variables: g_t, s_t, mp_t, i^n_{t-1}
- Number of grid points: N_g, N_s, N_{mp}, N_{i^n}
- Grid boundaries:

$$[g_{\min}, g_{\max}],\ [s_{\min}, s_{\max}],\ [mp_{\min}, mp_{\max}],\ [i^n_{\min}, i^n_{\max}]$$

- Create evenly spaced grids:

$$x_{grid} = \texttt{linspace}(x_{\min}, x_{\max}, N_x), \quad x \in \{g, s, mp, i^n\}$$

- State space contains $N = N_g \times N_s \times N_{mp} \times N_{i^n}$ nodes
- Create an array for each state variable, where every position is a unique permutation of the state space:

$$[g_{gr}, s_{gr}, mp_{gr}, i^n_{gr}] = \texttt{ndgrid}(g_{grid}, s_{grid}, mp_{grid}, i^n_{grid})$$

Discretized State Space (No Capital)

- State variables: g_t, s_t, mp_t, i^n_{t-1}
- Number of grid points: N_g, N_s, N_{mp}, N_{i^n}
- Grid boundaries:

$$[g_{\min}, g_{\max}],\ [s_{\min}, s_{\max}],\ [mp_{\min}, mp_{\max}],\ [i^n_{\min}, i^n_{\max}]$$

- Create evenly spaced grids:

$$x_{grid} = \texttt{linspace}(x_{\min}, x_{\max}, N_x), \quad x \in \{g, s, mp, i^n\}$$

- State space contains $N = N_g \times N_s \times N_{mp} \times N_{i^n}$ nodes
- Create an array for each state variable, where every position is a unique permutation of the state space:

$$[g_{gr}, s_{gr}, mp_{gr}, i^n_{gr}] = \texttt{ndgrid}(g_{grid}, s_{grid}, mp_{grid}, i^n_{grid})$$

DISCRETIZED STATE SPACE (NO CAPITAL)

- State variables: g_t, s_t, mp_t, i^n_{t-1}
- Number of grid points: N_g, N_s, N_{mp}, N_{i^n}
- Grid boundaries:

$$[g_{\min}, g_{\max}],\ [s_{\min}, s_{\max}],\ [mp_{\min}, mp_{\max}],\ [i^n_{\min}, i^n_{\max}]$$

- Create evenly spaced grids:

$$x_{grid} = \texttt{linspace}(x_{\min}, x_{\max}, N_x), \quad x \in \{g, s, mp, i^n\}$$

- State space contains $N = N_g \times N_s \times N_{mp} \times N_{i^n}$ nodes
- Create an array for each state variable, where every position is a unique permutation of the state space:

$$[g_{gr}, s_{gr}, mp_{gr}, i^n_{gr}] = \texttt{ndgrid}(g_{grid}, s_{grid}, mp_{grid}, i^n_{grid})$$

Discretized State Space (No Capital)

- State variables: g_t, s_t, mp_t, i^n_{t-1}
- Number of grid points: N_g, N_s, N_{mp}, N_{i^n}
- Grid boundaries:

$$[g_{\min}, g_{\max}],\ [s_{\min}, s_{\max}],\ [mp_{\min}, mp_{\max}],\ [i^n_{\min}, i^n_{\max}]$$

- Create evenly spaced grids:

$$x_{grid} = \texttt{linspace}(x_{\min}, x_{\max}, N_x), \quad x \in \{g, s, mp, i^n\}$$

- State space contains $N = N_g \times N_s \times N_{mp} \times N_{i^n}$ nodes
- Create an array for each state variable, where every position is a unique permutation of the state space:

$$[g_{gr}, s_{gr}, mp_{gr}, i^n_{gr}] = \texttt{ndgrid}(g_{grid}, s_{grid}, mp_{grid}, i^n_{grid})$$

Discretized State Space (No Capital)

- State variables: g_t, s_t, mp_t, i^n_{t-1}
- Number of grid points: N_g, N_s, N_{mp}, N_{i^n}
- Grid boundaries:

$$[g_{\min}, g_{\max}],\ [s_{\min}, s_{\max}],\ [mp_{\min}, mp_{\max}],\ [i^n_{\min}, i^n_{\max}]$$

- Create evenly spaced grids:

$$x_{grid} = \texttt{linspace}(x_{\min}, x_{\max}, N_x), \quad x \in \{g, s, mp, i^n\}$$

- State space contains $N = N_g \times N_s \times N_{mp} \times N_{i^n}$ nodes
- Create an array for each state variable, where every position is a unique permutation of the state space:

$$[g_{gr}, s_{gr}, mp_{gr}, i^n_{gr}] = \texttt{ndgrid}(g_{grid}, s_{grid}, mp_{grid}, i^n_{grid})$$

Discretized State Space (No Capital)

- State variables: g_t, s_t, mp_t, i^n_{t-1}
- Number of grid points: N_g, N_s, N_{mp}, N_{i^n}
- Grid boundaries:

$$[g_{\min}, g_{\max}],\ [s_{\min}, s_{\max}],\ [mp_{\min}, mp_{\max}],\ [i^n_{\min}, i^n_{\max}]$$

- Create evenly spaced grids:

$$x_{grid} = \texttt{linspace}(x_{\min}, x_{\max}, N_x), \quad x \in \{g, s, mp, i^n\}$$

- State space contains $N = N_g \times N_s \times N_{mp} \times N_{i^n}$ nodes
- Create an array for each state variable, where every position is a unique permutation of the state space:

$$[g_{gr}, s_{gr}, mp_{gr}, i^n_{gr}] = \texttt{ndgrid}(g_{grid}, s_{grid}, mp_{grid}, i^n_{grid})$$

Functional Approximation

- True RE solution only exists in special cases

- Goal: Find an approximating function that maps the state space to the optimal decision rule for consumption:

$$\underbrace{c(g, s, mp, i^n)}_{\text{True RE Solution}} \approx \underbrace{\mathcal{P}_c(g, s, mp, i^n)}_{\text{Approximating Function}}$$

- Basic elements of the algorithm:
  1. Interpolation: Linear, Chebyshev polynomials
  2. Integration: Rouwenhorst, Gauss-Hermite
  3. Iteration: Time, Fixed-point

Functional Approximation

- True RE solution only exists in special cases

- Goal: Find an approximating function that maps the state space to the optimal decision rule for consumption:

$$\underbrace{c(g, s, mp, i^n)}_{\text{True RE Solution}} \approx \underbrace{\mathcal{P}_c(g, s, mp, i^n)}_{\text{Approximating Function}}$$

- Basic elements of the algorithm:
  1. Interpolation: Linear, Chebyshev polynomials
  2. Integration: Rouwenhorst, Gauss-Hermite
  3. Iteration: Time, Fixed-point

Functional Approximation

- True RE solution only exists in special cases

- Goal: Find an approximating function that maps the state space to the optimal decision rule for consumption:

$$\underbrace{c(g, s, mp, i^n)}_{\text{True RE Solution}} \approx \underbrace{\mathcal{P}_c(g, s, mp, i^n)}_{\text{Approximating Function}}$$

- Basic elements of the algorithm:
  1. Interpolation: Linear, Chebyshev polynomials
  2. Integration: Rouwenhorst, Gauss-Hermite
  3. Iteration: Time, Fixed-point

Local Approximation

- Piecewise Linear Interpolation: 4 state variables (g, s, mp, i^n)
- Goal: Find the policy function value $\mathcal{P}_c(g', s', mp', i^{n\prime})$
- We have policy function values on nearest 24 nodes

$$[\mathcal{P}_c(g_i, s_j, mp_k, i^n_l), \mathcal{P}_c(g_i, s_j, mp_k, i^n_{l+1}), \mathcal{P}_c(g_i, s_j, mp_{k+1}, i^n_l),$$
$$\ldots, \mathcal{P}_c(g_{i+1}, s_{j+1}, mp_{k+1}, i^n_{l+1})]$$

  once we determine the grid indices, i, j, k, l

- Locate the grid point left of x', $x \in \{g, s, mp, i^n\}$

$$\text{step} = x_2 - x_1, \qquad \text{dist} = x' - x_1$$
$$\text{loc} = \min(N_x - 1, \max(1, \text{floor}(\text{dist}/\text{step}) + 1))$$

Local Approximation

- Piecewise Linear Interpolation: 4 state variables (g, s, mp, i^n)
- Goal: Find the policy function value $\mathcal{P}_c(g', s', mp', i^{n\prime})$
- We have policy function values on nearest 24 nodes

$$[\mathcal{P}_c(g_i, s_j, mp_k, i^n_l), \mathcal{P}_c(g_i, s_j, mp_k, i^n_{l+1}), \mathcal{P}_c(g_i, s_j, mp_{k+1}, i^n_l),$$
$$\ldots, \mathcal{P}_c(g_{i+1}, s_{j+1}, mp_{k+1}, i^n_{l+1})]$$

  once we determine the grid indices, i, j, k, l
- Locate the grid point left of x', $x \in \{g, s, mp, i^n\}$

$$\text{step} = x_2 - x_1, \qquad \text{dist} = x' - x_1$$
$$\text{loc} = \min(N_x - 1, \max(1, \text{floor}(\text{dist}/\text{step}) + 1))$$

Local Approximation

- Piecewise Linear Interpolation: 4 state variables (g, s, mp, i^n)
- Goal: Find the policy function value $\mathcal{P}_c(g', s', mp', i^{n\prime})$
- We have policy function values on nearest 24 nodes

$$[\mathcal{P}_c(g_i, s_j, mp_k, i^n_l), \mathcal{P}_c(g_i, s_j, mp_k, i^n_{l+1}), \mathcal{P}_c(g_i, s_j, mp_{k+1}, i^n_l), \\ \ldots, \mathcal{P}_c(g_{i+1}, s_{j+1}, mp_{k+1}, i^n_{l+1})]$$

once we determine the grid indices, i, j, k, l

- Locate the grid point left of x', $x \in \{g, s, mp, i^n\}$

$$\text{step} = x_2 - x_1, \qquad \text{dist} = x' - x_1$$
$$\text{loc} = \min(N_x - 1, \max(1, \text{floor}(\text{dist}/\text{step}) + 1))$$

Local Approximation

- Piecewise Linear Interpolation: 4 state variables (g, s, mp, i^n)
- Goal: Find the policy function value $\mathcal{P}_c(g', s', mp', i^{n\prime})$
- We have policy function values on nearest 24 nodes

$$[\mathcal{P}_c(g_i, s_j, mp_k, i^n_l), \mathcal{P}_c(g_i, s_j, mp_k, i^n_{l+1}), \mathcal{P}_c(g_i, s_j, mp_{k+1}, i^n_l),\\ \ldots, \mathcal{P}_c(g_{i+1}, s_{j+1}, mp_{k+1}, i^n_{l+1})]$$

  once we determine the grid indices, i, j, k, l

- Locate the grid point left of x', $x \in \{g, s, mp, i^n\}$

$$\text{step} = x_2 - x_1, \qquad \text{dist} = x' - x_1$$
$$\text{loc} = \min(N_x - 1, \max(1, \text{floor}(\text{dist}/\text{step}) + 1))$$

Global Approximation

- A general class of polynomials can be written as:

$$\mathcal{P}(x;\eta) = \sum_{i=0}^{n} \eta_i \varphi_i(x).$$

- Linear interpolation is a special case of this general class (i.e., $n = 1$, $\varphi_i(x) = x^i$, and weights chosen appropriately)

- Can also use bases consisting of orthogonal polynomials (e.g., Chebyshev Polynomials)

Global Approximation

- A general class of polynomials can be written as:

$$\mathcal{P}(x;\eta) = \sum_{i=0}^{n} \eta_i \varphi_i(x).$$

- Linear interpolation is a special case of this general class (i.e., $n = 1$, $\varphi_i(x) = x^i$, and weights chosen appropriately)

- Can also use bases consisting of orthogonal polynomials (e.g., Chebyshev Polynomials)

Global Approximation

- A general class of polynomials can be written as:

$$\mathcal{P}(x;\eta) = \sum_{i=0}^{n} \eta_i \varphi_i(x).$$

- Linear interpolation is a special case of this general class (i.e., $n = 1$, $\varphi_i(x) = x^i$, and weights chosen appropriately)
- Can also use bases consisting of orthogonal polynomials (e.g., Chebyshev Polynomials)

Rouwenhorst Method

- Used to approximate an exogenous $AR(p)$ process
- Kopecky and Suen (2010) show the Rouwenhorst method outperforms other approximations of an $AR(1)$ process
- The method determines the bounds of the exogenous state variables, the nodes, and the transition probabilities

Rouwenhorst Method

- Used to approximate an exogenous $AR(p)$ process
- Kopecky and Suen (2010) show the Rouwenhorst method outperforms other approximations of an $AR(1)$ process
- The method determines the bounds of the exogenous state variables, the nodes, and the transition probabilities

Rouwenhorst Method

- Used to approximate an exogenous $AR(p)$ process
- Kopecky and Suen (2010) show the Rouwenhorst method outperforms other approximations of an $AR(1)$ process
- The method determines the bounds of the exogenous state variables, the nodes, and the transition probabilities

Example: Time Iteration

On iteration q, solve for the $\mathcal{P}_c^q(g, s, mp, i^n)$ that *satisfies* equilibrium

1. Use log-linear solution on each node to obtain $\mathcal{P}_c^0$
   - Local: $\mathcal{P}_c^1 = \mathcal{P}_c^0$
   - Global: set initial basis $\hat{\eta}^0$
2. Solve for g', s', mp', $i^{n\prime}$, given ε_g', ε_s', ε_i'
3. Find $\mathcal{P}_c^q(g', s', mp', i^{n\prime})$ given the updated state
   - Local: use piecewise linear interpolation
   - Global: update the basis
4. Evaluate expectations (Rouwenhorst or Gauss Hermite)

Example: Time Iteration

On iteration q, solve for the $\mathcal{P}_c^q(g, s, mp, i^n)$ that *satisfies* equilibrium

1. Use log-linear solution on each node to obtain $\mathcal{P}_c^0$
   - Local: $\mathcal{P}_c^1 = \mathcal{P}_c^0$
   - Global: set initial basis $\hat{\eta}^0$

2. Solve for g', s', mp', $i^{n\prime}$, given ε_g', ε_s', ε_i'

3. Find $\mathcal{P}_c^q(g', s', mp', i^{n\prime})$ given the updated state
   - Local: use piecewise linear interpolation
   - Global: update the basis

4. Evaluate expectations (Rouwenhorst or Gauss Hermite)

Example: Time Iteration

On iteration q, solve for the $\mathcal{P}_c^q(g, s, mp, i^n)$ that *satisfies* equilibrium

1. Use log-linear solution on each node to obtain $\mathcal{P}_c^0$
   - Local: $\mathcal{P}_c^1 = \mathcal{P}_c^0$
   - Global: set initial basis $\hat{\eta}^0$

2. Solve for g', s', mp', $i^{n\prime}$, given ε_g', ε_s', ε_i'

3. Find $\mathcal{P}_c^q(g', s', mp', i^{n\prime})$ given the updated state
   - Local: use piecewise linear interpolation
   - Global: update the basis

4. Evaluate expectations (Rouwenhorst or Gauss Hermite)

Example: Time Iteration

On iteration q, solve for the $\mathcal{P}_c^q(g, s, mp, i^n)$ that *satisfies* equilibrium

1. Use log-linear solution on each node to obtain $\mathcal{P}_c^0$
   - Local: $\mathcal{P}_c^1 = \mathcal{P}_c^0$
   - Global: set initial basis $\hat{\eta}^0$

2. Solve for g', s', mp', $i^{n\prime}$, given ε_g', ε_s', ε_i'

3. Find $\mathcal{P}_c^q(g', s', mp', i^{n\prime})$ given the updated state
   - Local: use piecewise linear interpolation
   - Global: update the basis

4. Evaluate expectations (Rouwenhorst or Gauss Hermite)

Example: Time Iteration

5. Use nonlinear solver to find a $\mathcal{P}_c^q(g, s, mp, i^n)$ that satisfies the consumption Euler equation
6. Update policy function
   - Local: $\mathcal{P}_c^{q+1} = \mathcal{P}_c^q$
   - Global: update basis $\hat{\eta}^q$
7. Calculate distance between updates
   - Local: $\text{dist} = \mathcal{P}_c^q(g, s, mp, i^n) - \mathcal{P}_c^{q-1}(g, s, mp, i^n)$
   - Global: $\text{dist} = \hat{\eta}^q - \hat{\eta}^{q-1}$
8. If $|\text{dist}| < \text{tol}$, then stop. If not, then set $q = q + 1$ and repeat steps 2-7 using $\mathcal{P}_c^{q+1}$ as the new initial conjecture.

Advantage: Satisfies the equilibrium system on each node and nodes can be run in parallel.

Disadvantage: Nonlinear solver must execute on each node.

Example: Time Iteration

5. Use nonlinear solver to find a $\mathcal{P}_c^q(g, s, mp, i^n)$ that satisfies the consumption Euler equation
6. Update policy function
   - Local: $\mathcal{P}_c^{q+1} = \mathcal{P}_c^q$
   - Global: update basis $\hat{\eta}^q$
7. Calculate distance between updates
   - Local: $\text{dist} = \mathcal{P}_c^q(g, s, mp, i^n) - \mathcal{P}_c^{q-1}(g, s, mp, i^n)$
   - Global: $\text{dist} = \hat{\eta}^q - \hat{\eta}^{q-1}$
8. If $|\text{dist}| < \text{tol}$, then stop. If not, then set $q = q + 1$ and repeat steps 2-7 using $\mathcal{P}_c^{q+1}$ as the new initial conjecture.

Advantage: Satisfies the equilibrium system on each node and nodes can be run in parallel.

Disadvantage: Nonlinear solver must execute on each node.

Example: Time Iteration

5. Use nonlinear solver to find a $\mathcal{P}_c^q(g, s, mp, i^n)$ that satisfies the consumption Euler equation
6. Update policy function
   - Local: $\mathcal{P}_c^{q+1} = \mathcal{P}_c^q$
   - Global: update basis $\hat{\eta}^q$
7. Calculate distance between updates
   - Local: $\text{dist} = \mathcal{P}_c^q(g, s, mp, i^n) - \mathcal{P}_c^{q-1}(g, s, mp, i^n)$
   - Global: $\text{dist} = \hat{\eta}^q - \hat{\eta}^{q-1}$
8. If $|\text{dist}| < \text{tol}$, then stop. If not, then set $q = q + 1$ and repeat steps 2-7 using $\mathcal{P}_c^{q+1}$ as the new initial conjecture.

Advantage: Satisfies the equilibrium system on each node and nodes can be run in parallel.

Disadvantage: Nonlinear solver must execute on each node.

Example: Time Iteration

5. Use nonlinear solver to find a $\mathcal{P}_c^q(g, s, mp, i^n)$ that satisfies the consumption Euler equation
6. Update policy function
   - Local: $\mathcal{P}_c^{q+1} = \mathcal{P}_c^q$
   - Global: update basis $\hat{\eta}^q$
7. Calculate distance between updates
   - Local: $\text{dist} = \mathcal{P}_c^q(g, s, mp, i^n) - \mathcal{P}_c^{q-1}(g, s, mp, i^n)$
   - Global: $\text{dist} = \hat{\eta}^q - \hat{\eta}^{q-1}$
8. If $|\text{dist}| < \text{tol}$, then stop. If not, then set $q = q + 1$ and repeat steps 2-7 using $\mathcal{P}_c^{q+1}$ as the new initial conjecture.

Advantage: Satisfies the equilibrium system on each node and nodes can be run in parallel.

Disadvantage: Nonlinear solver must execute on each node.

Example: Time Iteration

5. Use nonlinear solver to find a $\mathcal{P}_c^q(g, s, mp, i^n)$ that satisfies the consumption Euler equation
6. Update policy function
   - Local: $\mathcal{P}_c^{q+1} = \mathcal{P}_c^q$
   - Global: update basis $\hat{\eta}^q$
7. Calculate distance between updates
   - Local: $\text{dist} = \mathcal{P}_c^q(g, s, mp, i^n) - \mathcal{P}_c^{q-1}(g, s, mp, i^n)$
   - Global: $\text{dist} = \hat{\eta}^q - \hat{\eta}^{q-1}$
8. If $|\text{dist}| < \text{tol}$, then stop. If not, then set $q = q + 1$ and repeat steps 2-7 using $\mathcal{P}_c^{q+1}$ as the new initial conjecture.

Advantage: Satisfies the equilibrium system on each node and nodes can be run in parallel.

Disadvantage: Nonlinear solver must execute on each node.

Example: Fixed-Point Iteration

Solve for the $\mathcal{P}_c^q(g, s, mp, i^n)$ *implied* by the equilibrium system

1. Obtain an initial conjecture
   (step 1 in the time iteration algorithm)
2. Calculate updated variables and expectations
   (steps 2-4 in the time iteration algorithm)
3. Calculate $\mathcal{P}_c^q(g, s, mp, i^n)$ on each node
4. Execute steps 6-8 in the time iteration algorithm

Advantage: Does not require a loop, since all of the nodes are evaluated simultaneously.

Disadvantage: Algorithm is less stable because it does not solve for the optimal policy function on each node.

Example: Fixed-Point Iteration

Solve for the $\mathcal{P}_c^q(g, s, mp, i^n)$ *implied* by the equilibrium system

1. Obtain an initial conjecture
   (step 1 in the time iteration algorithm)
2. Calculate updated variables and expectations
   (steps 2-4 in the time iteration algorithm)
3. Calculate $\mathcal{P}_c^q(g, s, mp, i^n)$ on each node
4. Execute steps 6-8 in the time iteration algorithm

Advantage: Does not require a loop, since all of the nodes are evaluated simultaneously.

Disadvantage: Algorithm is less stable because it does not solve for the optimal policy function on each node.

Example: Fixed-Point Iteration

Solve for the $\mathcal{P}_c^q(g, s, mp, i^n)$ *implied* by the equilibrium system

1. Obtain an initial conjecture
   (step 1 in the time iteration algorithm)
2. Calculate updated variables and expectations
   (steps 2-4 in the time iteration algorithm)
3. Calculate $\mathcal{P}_c^q(g, s, mp, i^n)$ on each node
4. Execute steps 6-8 in the time iteration algorithm

Advantage: Does not require a loop, since all of the nodes are evaluated simultaneously.

Disadvantage: Algorithm is less stable because it does not solve for the optimal policy function on each node.

Example: Fixed-Point Iteration

Solve for the $\mathcal{P}_c^q(g, s, mp, i^n)$ *implied* by the equilibrium system

1. Obtain an initial conjecture
   (step 1 in the time iteration algorithm)
2. Calculate updated variables and expectations
   (steps 2-4 in the time iteration algorithm)
3. Calculate $\mathcal{P}_c^q(g, s, mp, i^n)$ on each node
4. Execute steps 6-8 in the time iteration algorithm

Advantage: Does not require a loop, since all of the nodes are evaluated simultaneously.

Disadvantage: Algorithm is less stable because it does not solve for the optimal policy function on each node.

Example: Fixed-Point Iteration

Solve for the $\mathcal{P}_c^q(g, s, mp, i^n)$ *implied* by the equilibrium system

1. Obtain an initial conjecture
   (step 1 in the time iteration algorithm)
2. Calculate updated variables and expectations
   (steps 2-4 in the time iteration algorithm)
3. Calculate $\mathcal{P}_c^q(g, s, mp, i^n)$ on each node
4. Execute steps 6-8 in the time iteration algorithm

Advantage: Does not require a loop, since all of the nodes are evaluated simultaneously.

Disadvantage: Algorithm is less stable because it does not solve for the optimal policy function on each node.

Regime-Indexed Policy Functions

- Let the vector of policy functions at time t be denoted $\mathbf{pf}_t$ and the realization on node d be denoted $\mathbf{pf}_t(d)$

- The regime-indexed policy functions are as follows:

$$\mathbf{pf}_t(d) = \mathbf{pf}_{t,1}(d)\mathbb{I}_t(d) + \mathbf{pf}_{t,2}(d)(1 - \mathbb{I}_t(d)),$$

  where $\mathbb{I}_t(d)$ is defined by

$$\mathbb{I}_t(d) = \begin{cases} 1 & \text{if } i_t > 1 \\ 0 & \text{otherwise.} \end{cases}$$

Regime-Indexed Policy Functions

- Let the vector of policy functions at time t be denoted $\mathbf{pf}_t$ and the realization on node d be denoted $\mathbf{pf}_t(d)$
- The regime-indexed policy functions are as follows:

$$\mathbf{pf}_t(d) = \mathbf{pf}_{t,1}(d)\mathbb{I}_t(d) + \mathbf{pf}_{t,2}(d)(1 - \mathbb{I}_t(d)),$$

where $\mathbb{I}_t(d)$ is defined by

$$\mathbb{I}_t(d) = \begin{cases} 1 & \text{if } i_t > 1 \\ 0 & \text{otherwise.} \end{cases}$$

Regime-Indexed Policy Functions

The functions $\mathbf{pf}_{t,j}$ satisfy the residual functions $R_{t,l,j}$ for $j \in \{1,2\}$ and $l \in \{1,2,3,4\}$:

$$R_{t,1,1} = 1 - s_t i_t \beta E_t[(\lambda_t/\lambda_{t+1})(1/(\bar{\pi}\pi_{t+1}^{gap}))],$$
$$R_{t,1,2} = 1 - s_t \beta E_t[(\lambda_t/\lambda_{t+1})(1/(\bar{\pi}\pi_{t+1}^{gap}))],$$
$$R_{t,2,j} = q_t - \beta E_t[(\lambda_t/\lambda_{t+1})(r_{t+1}^k + (1-\delta)q_{t+1})],$$
$$R_{t,3,j} = 1 - q_t[1 - \nu(x_t^g - 1)^2/2 - \nu(x_t^g - 1)x_t^g] - \nu\beta\bar{g}E_t[q_{t+1}(\lambda_t/\lambda_{t+1})(x_{t+1}^g)^2(x_{t+1}^g - 1)],$$
$$R_{t,4,j} = \varphi(\pi_t^{gap} - 1)\pi_t^{gap} - (1-\theta) - \theta mc_t - \beta\varphi E_t[(\lambda_t/\lambda_{t+1})(\pi_{t+1}^{gap} - 1)\pi_{t+1}^{gap}(y_{t+1}/y_t)].$$

Solution Algorithm

Construct an evenly-spaced grid using the Rouwenhorst method. Denote the vector of states $\mathbf{z}_t$.

1. Solve the linear model using Sims's (2002) `gensys` algorithm.
2. Solve the nonlinear model using fixed point iteration. For each node in the state space $d \in \{1, \ldots, D\}$:
   A. Solve for the variables dated at time t given $\mathbf{pf}_t$ and $\mathbf{z}_t$.
   B. Linearly interpolate the policy functions at the updated state variables $\mathbf{z}_{t+1}$ to obtain $\mathbf{pf}_{t+1}(m)$ on every integration node $m \in \{1, \ldots, M\}$.
   C. Given the interpolated policy functions $\{\mathbf{pf}(m)\}_{m=1}^{M}$, and integration nodes and weights provided by the Rouwenhorst method, approximate the expectation operators for the model.
   D. Back out the policy functions $\mathbf{pf}_t(d)$ from the expectation operators and time $t+1$ variables.
3. Repeat step 2 until the maximum distance between successive approximations of the policy functions is below 10^{-6}.

Solution Algorithm

Construct an evenly-spaced grid using the Rouwenhorst method. Denote the vector of states $\mathbf{z}_t$.

1. Solve the linear model using Sims's (2002) `gensys` algorithm.

2. Solve the nonlinear model using fixed point iteration. For each node in the state space $d \in \{1, \ldots, D\}$:
   A. Solve for the variables dated at time t given $\mathbf{pf}_t$ and $\mathbf{z}_t$.
   B. Linearly interpolate the policy functions at the updated state variables $\mathbf{z}_{t+1}$ to obtain $\mathbf{pf}_{t+1}(m)$ on every integration node $m \in \{1, \ldots, M\}$.
   C. Given the interpolated policy functions $\{\mathbf{pf}(m)\}_{m=1}^{M}$, and integration nodes and weights provided by the Rouwenhorst method, approximate the expectation operators for the model.
   D. Back out the policy functions $\mathbf{pf}_t(d)$ from the expectation operators and time $t+1$ variables.

3. Repeat step 2 until the maximum distance between successive approximations of the policy functions is below 10^{-6}.

Solution Algorithm

Construct an evenly-spaced grid using the Rouwenhorst method. Denote the vector of states $\mathbf{z}_t$.

1. Solve the linear model using Sims's (2002) `gensys` algorithm.
2. Solve the nonlinear model using fixed point iteration. For each node in the state space $d \in \{1, \ldots, D\}$:
   a. Solve for the variables dated at time t given $\mathbf{pf}_t$ and $\mathbf{z}_t$.
   b. Linearly interpolate the policy functions at the updated state variables $\mathbf{z}_{t+1}$ to obtain $\mathbf{pf}_{t+1}(m)$ on every integration node $m \in \{1, \ldots, M\}$.
   c. Given the interpolated policy functions $\{\mathbf{pf}(m)\}_{m=1}^{M}$, and integration nodes and weights provided by the Rouwenhorst method, approximate the expectation operators for the model.
   d. Back out the policy functions $\mathbf{pf}_t(d)$ from the expectation operators and time $t+1$ variables.
3. Repeat step 2 until the maximum distance between successive approximations of the policy functions is below 10^{-6}.

Solution Algorithm

Construct an evenly-spaced grid using the Rouwenhorst method. Denote the vector of states $\mathbf{z}_t$.

1. Solve the linear model using Sims's (2002) `gensys` algorithm.
2. Solve the nonlinear model using fixed point iteration. For each node in the state space $d \in \{1, \ldots, D\}$:
   A. Solve for the variables dated at time t given $\mathbf{pf}_t$ and $\mathbf{z}_t$.
   B. Linearly interpolate the policy functions at the updated state variables $\mathbf{z}_{t+1}$ to obtain $\mathbf{pf}_{t+1}(m)$ on every integration node $m \in \{1, \ldots, M\}$.
   C. Given the interpolated policy functions $\{\mathbf{pf}(m)\}_{m=1}^{M}$, and integration nodes and weights provided by the Rouwenhorst method, approximate the expectation operators for the model.
   D. Back out the policy functions $\mathbf{pf}_t(d)$ from the expectation operators and time $t+1$ variables.
3. Repeat step 2 until the maximum distance between successive approximations of the policy functions is below 10^{-6}.

Solution Algorithm

Construct an evenly-spaced grid using the Rouwenhorst method. Denote the vector of states $\mathbf{z}_t$.

1. Solve the linear model using Sims's (2002) `gensys` algorithm.
2. Solve the nonlinear model using fixed point iteration. For each node in the state space $d \in \{1, \ldots, D\}$:
   A. Solve for the variables dated at time t given $\mathbf{pf}_t$ and $\mathbf{z}_t$.
   B. Linearly interpolate the policy functions at the updated state variables $\mathbf{z}_{t+1}$ to obtain $\mathbf{pf}_{t+1}(m)$ on every integration node $m \in \{1, \ldots, M\}$.
   C. Given the interpolated policy functions $\{\mathbf{pf}(m)\}_{m=1}^{M}$, and integration nodes and weights provided by the Rouwenhorst method, approximate the expectation operators for the model.
   D. Back out the policy functions $\mathbf{pf}_t(d)$ from the expectation operators and time $t+1$ variables.
3. Repeat step 2 until the maximum distance between successive approximations of the policy functions is below 10^{-6}.

Solution Algorithm

Construct an evenly-spaced grid using the Rouwenhorst method. Denote the vector of states $\mathbf{z}_t$.

1. Solve the linear model using Sims's (2002) `gensys` algorithm.

2. Solve the nonlinear model using fixed point iteration. For each node in the state space $d \in \{1, \ldots, D\}$:
   A. Solve for the variables dated at time t given $\mathbf{pf}_t$ and $\mathbf{z}_t$.
   B. Linearly interpolate the policy functions at the updated state variables $\mathbf{z}_{t+1}$ to obtain $\mathbf{pf}_{t+1}(m)$ on every integration node $m \in \{1, \ldots, M\}$.
   C. Given the interpolated policy functions $\{\mathbf{pf}(m)\}_{m=1}^{M}$, and integration nodes and weights provided by the Rouwenhorst method, approximate the expectation operators for the model.
   D. Back out the policy functions $\mathbf{pf}_t(d)$ from the expectation operators and time $t+1$ variables.

3. Repeat step 2 until the maximum distance between successive approximations of the policy functions is below 10^{-6}.

Solution Algorithm

Construct an evenly-spaced grid using the Rouwenhorst method. Denote the vector of states $\mathbf{z}_t$.

1. Solve the linear model using Sims's (2002) `gensys` algorithm.

2. Solve the nonlinear model using fixed point iteration. For each node in the state space $d \in \{1, \ldots, D\}$:
   a. Solve for the variables dated at time t given $\mathbf{pf}_t$ and $\mathbf{z}_t$.
   b. Linearly interpolate the policy functions at the updated state variables $\mathbf{z}_{t+1}$ to obtain $\mathbf{pf}_{t+1}(m)$ on every integration node $m \in \{1, \ldots, M\}$.
   c. Given the interpolated policy functions $\{\mathbf{pf}(m)\}_{m=1}^{M}$, and integration nodes and weights provided by the Rouwenhorst method, approximate the expectation operators for the model.
   d. Back out the policy functions $\mathbf{pf}_t(d)$ from the expectation operators and time $t+1$ variables.

3. Repeat step 2 until the maximum distance between successive approximations of the policy functions is below 10^{-6}.

Solution Algorithm

Construct an evenly-spaced grid using the Rouwenhorst method. Denote the vector of states $\mathbf{z}_t$.

1. Solve the linear model using Sims's (2002) `gensys` algorithm.
2. Solve the nonlinear model using fixed point iteration. For each node in the state space $d \in \{1, \ldots, D\}$:
   A. Solve for the variables dated at time t given $\mathbf{pf}_t$ and $\mathbf{z}_t$.
   B. Linearly interpolate the policy functions at the updated state variables $\mathbf{z}_{t+1}$ to obtain $\mathbf{pf}_{t+1}(m)$ on every integration node $m \in \{1, \ldots, M\}$.
   C. Given the interpolated policy functions $\{\mathbf{pf}(m)\}_{m=1}^{M}$, and integration nodes and weights provided by the Rouwenhorst method, approximate the expectation operators for the model.
   D. Back out the policy functions $\mathbf{pf}_t(d)$ from the expectation operators and time $t+1$ variables.
3. Repeat step 2 until the maximum distance between successive approximations of the policy functions is below 10^{-6}.

Euler Equation Errors

- Measure of solution accuracy
- To approximate errors between nodes, we use Gauss-Hermite quadrature instead of the Rouwenhorst method
- The Euler equation errors are represented in absolute value of the errors in base 10 logarithms
  - An Euler equation error of -3 means the household makes an error equivalent to one per 1,000 consumption goods
- Obtaining Euler equation errors:
  - Simulate 10,000 periods of the model using the nonlinear solution with random shocks
  - Follow one iteration of step 2 of our solution algorithm and transform the residuals to Euler equation errors

Euler Equation Errors

- Measure of solution accuracy
- To approximate errors between nodes, we use Gauss-Hermite quadrature instead of the Rouwenhorst method
- The Euler equation errors are represented in absolute value of the errors in base 10 logarithms
  - An Euler equation error of -3 means the household makes an error equivalent to one per 1,000 consumption goods
- Obtaining Euler equation errors:
  - Simulate 10,000 periods of the model using the nonlinear solution with random shocks
  - Follow one iteration of step 2 of our solution algorithm and transform the residuals to Euler equation errors

EULER EQUATION ERRORS

- Measure of solution accuracy
- To approximate errors between nodes, we use Gauss-Hermite quadrature instead of the Rouwenhorst method
- The Euler equation errors are represented in absolute value of the errors in base 10 logarithms
  - An Euler equation error of -3 means the household makes an error equivalent to one per 1,000 consumption goods
- Obtaining Euler equation errors:
  - Simulate 10,000 periods of the model using the nonlinear solution with random shocks
  - Follow one iteration of step 2 of our solution algorithm and transform the residuals to Euler equation errors

Euler Equation Errors

- Measure of solution accuracy
- To approximate errors between nodes, we use Gauss-Hermite quadrature instead of the Rouwenhorst method
- The Euler equation errors are represented in absolute value of the errors in base 10 logarithms
  - An Euler equation error of -3 means the household makes an error equivalent to one per 1,000 consumption goods
- Obtaining Euler equation errors:
  - Simulate 10,000 periods of the model using the nonlinear solution with random shocks
  - Follow one iteration of step 2 of our solution algorithm and transform the residuals to Euler equation errors

Parallel Processing in Matlab

- Any calculations that are not dependent on the results of other calculations can be performed in parallel (e.g., solving for policy values at each node in the state space)
- We replace `for` loops with `parfor` loops where applicable. This tells MATLAB to distribute each step in the loop across the specified number of processors

Parallel Processing in Matlab

- Any calculations that are not dependent on the results of other calculations can be performed in parallel (e.g., solving for policy values at each node in the state space)
- We replace `for` loops with `parfor` loops where applicable. This tells MATLAB to distribute each step in the loop across the specified number of processors

Solution Times

	Model without capital			Model with capital		
	Total nodes	Iterations	Total Time	Total Nodes	Iterations	Total time
ART	2401	76	23.01s (avg. 30 runs)	78125	56	0h 9m 16.1s
GHLS	2401	62	19.34s (avg. 30 runs)	78125	1231	3h 28m 28.8s

Solution times

- The solution times were computed with multi-core processing using the Parallel Computing Toolbox
- We implemented the interpolation steps of the algorithm in Fortran with MATLAB executable functions (MEX) provided in the companion toolbox to Richter et al. (2014)

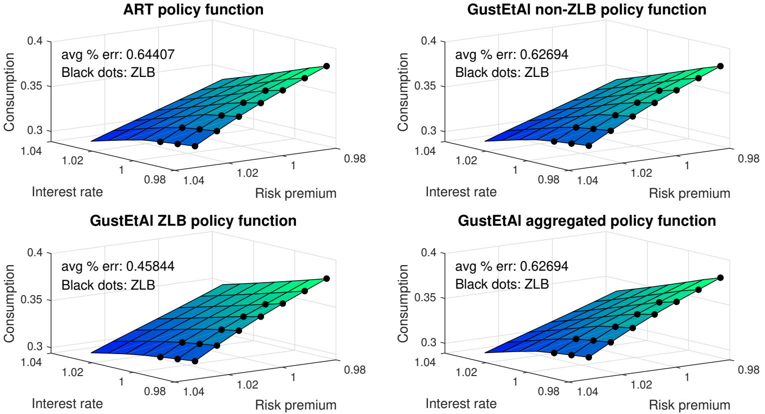
Solution Times

	Model without capital			Model with capital		
	Total nodes	Iterations	Total Time	Total Nodes	Iterations	Total time
ART	2401	76	23.01s (avg. 30 runs)	78125	56	0h 9m 16.1s
GHLS	2401	62	19.34s (avg. 30 runs)	78125	1231	3h 28m 28.8s

Solution times

- The solution times were computed with multi-core processing using the Parallel Computing Toolbox
- We implemented the interpolation steps of the algorithm in Fortran with MATLAB executable functions (MEX) provided in the companion toolbox to Richter et al. (2014)

Policy Functions: Model Without Capital



Consumption policy function for model without capital

Smoothness Measures

	Model without capital		Model with capital	
	Mean % Error	RMSE	Mean % Error	RMSE
ART policy	0.64407%	0.0027327 c units	0.271154%	0.0039806 l units
GHLS policy	0.62694%	0.0026704 c units	0.578098%	0.0080785 l units

Smoothness measures for labor policy functions (c for model with capital and n for model with capital). GHLS combined policy functions are reported.

- RMSE (root mean square error)
- Mean percent error from linear policy functions
- GHLS smoother for model without capital; ART smoother for model with capital

Smoothness Measures

	Model without capital		Model with capital	
	Mean % Error	RMSE	Mean % Error	RMSE
ART policy	0.64407%	0.0027327 c units	0.271154%	0.0039806 l units
GHLS policy	0.62694%	0.0026704 c units	0.578098%	0.0080785 l units

Smoothness measures for labor policy functions (c for model with capital and n for model with capital). GHLS combined policy functions are reported.

- RMSE (root mean square error)
- Mean percent error from linear policy functions
- GHLS smoother for model without capital; ART smoother for model with capital

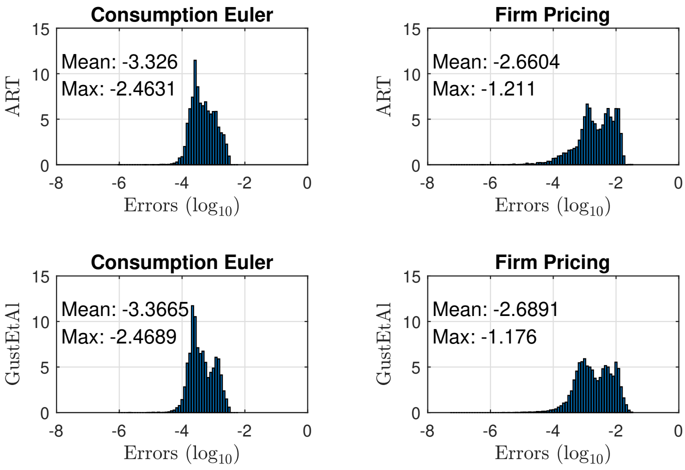
Smoothness Measures

	Model without capital		Model with capital	
	Mean % Error	RMSE	Mean % Error	RMSE
ART policy	0.64407%	0.0027327 c units	0.271154%	0.0039806 l units
GHLS policy	0.62694%	0.0026704 c units	0.578098%	0.0080785 l units

Smoothness measures for labor policy functions (c for model with capital and n for model with capital). GHLS combined policy functions are reported.

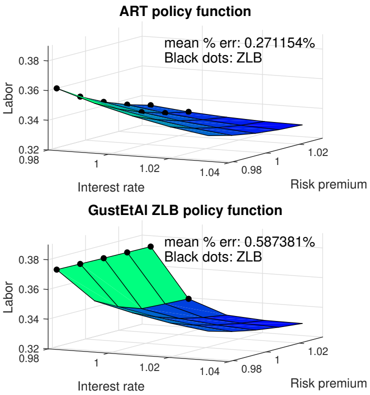
- RMSE (root mean square error)
- Mean percent error from linear policy functions
- GHLS smoother for model without capital; ART smoother for model with capital

Euler Equation Errors: Model Without Capital



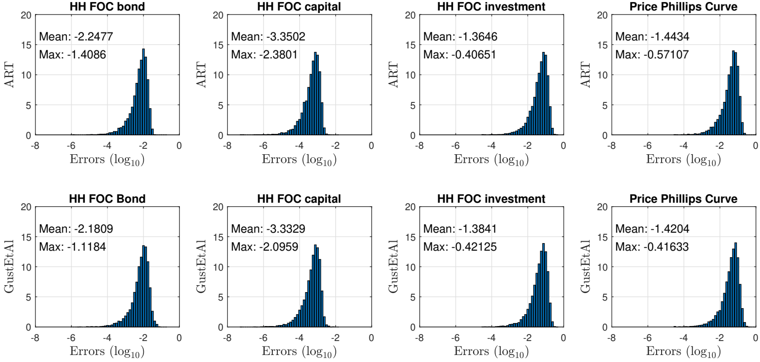
Euler equation errors for model without capital

Policy Functions: Model with Capital



Labor policy function for model with capital

Euler Equation Errors: Model With Capital



Euler equation errors for model with capital

Conclusion

- During the Great Recession, policy rates hit zero, calling into question traditional solution methods
- There is a literature in solving nonlinear models, but not much work comparing nonlinear solution methods
- This paper discusses the impact of regime-indexing the policy functions on a nonlinear solution algorithm
  1. Example of directly approximating the policy functions from Atkinson et al. (2019)
  2. Example of for regime-indexing the policy functions from Gust et al. (2017). They claim that regime-indexed policy functions are smoother and easier to approximate

Conclusion

- During the Great Recession, policy rates hit zero, calling into question traditional solution methods
- There is a literature in solving nonlinear models, but not much work comparing nonlinear solution methods
- This paper discusses the impact of regime-indexing the policy functions on a nonlinear solution algorithm
  1. Example of directly approximating the policy functions from Atkinson et al. (2019)
  2. Example of for regime-indexing the policy functions from Gust et al. (2017). They claim that regime-indexed policy functions are smoother and easier to approximate

Conclusion

- During the Great Recession, policy rates hit zero, calling into question traditional solution methods
- There is a literature in solving nonlinear models, but not much work comparing nonlinear solution methods
- This paper discusses the impact of regime-indexing the policy functions on a nonlinear solution algorithm
    1. Example of directly approximating the policy functions from Atkinson et al. (2019)
    2. Example of for regime-indexing the policy functions from Gust et al. (2017). They claim that regime-indexed policy functions are smoother and easier to approximate

CONCLUSION

- Key takeaways:
  - In the model without capital, regime-indexed policy functions were smoother and solution algorithm was faster
  - In model with capital, the regime-indexed policy functions were more nonlinear and solution algorithm was slower
- Extensions:
  - Solve the GHLS solution method using Smolyak discretization methods and Chebyshev polynomials
  - This would be expected to speed up the solution algorithm, but may lead to a less accurate solution

Conclusion

- Key takeaways:
  - In the model without capital, regime-indexed policy functions were smoother and solution algorithm was faster
  - In model with capital, the regime-indexed policy functions were more nonlinear and solution algorithm was slower
- Extensions:
  - Solve the GHLS solution method using Smolyak discretization methods and Chebyshev polynomials
  - This would be expected to speed up the solution algorithm, but may lead to a less accurate solution