How to solve nonlinear DSGE Models

May 22, 2018

Plan

- Part I: What is the time iteration method?
  - Our explanation tends to be intuitive and self-contained to some extent, but not necessarily be strict.
- Part II: Applications to New Keynesian models

Part I: What is the time iteration method?

- Neoclassical growth model
  - Interpolation
  - Nonlinear optimization (or how to avoid it)
- Stochastic neoclassical growth model
  - Two-dimensional interpolation
  - Nonlinear optimization (or how to avoid it)
  - Numerical integration (or how to avoid it)

Neoclassical growth model

- We consider an example of neoclassical growth model. An individual maximizes life-time utility

$$\sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$c_t + k_{t+1} \leq f(k_t).$$

where $u(\cdot)$ and $f(\cdot)$ satisfy standard conditions.

- The first-order necessary condition is given by

$$u_c(c_t) = \beta u_c(c_{t+1}) f_k(k_{t+1})$$

where $u_c(\cdot)$ denotes the derivative of u wrt c and $f_k(\cdot)$ denotes the derivative of f wrt k.

Collman operator

- There is a mapping $\sigma = K\sigma$ that solves

$$u_c(c) = \beta u_c\left(\sigma(f(k) - c)\right) f_k(f(k) - c)$$

for $c = \sigma(k)$. σ is called policy function.

  - Note that $k' = f(k) - c$ and $c' = \sigma(k') = \sigma(f(k) - c)$.
  - Bellman operator

$$V(k) = \max_{c \in (0, f(k)]} \left\{u(c) + \beta V\left(f(k) - c\right)\right\}.$$

    is to solve the Bellman equation, whereas the Collman operator is helpful to solve the Euler equation.

Collman operator, cont'd

- Collman (1990) proves the existence of the fixed point of K in a stochastic neoclassical growth model with distortionary tax.
    - Greenwood and Huffman (1995) extend it to several cases. Also see Richter, Throckmorton and Walker (2014) and Sargent and Stachurski (2018).

Time iteration: Algorithm

- Time iteration is a method to solve the Collman operator.
- The time iteration method takes the following steps:
  1. Make an initial guess for the policy function $\sigma^{(0)}$.
  2. Given the policy function previously obtained $\sigma^{(i-1)}$ (i is an index for the number of iteration), solve

$$u_c(c) = \beta u_c\left(\sigma^{(i-1)}(f(k)-c)\right) f_k(f(k)-c)$$

     for c.
  3. Update the policy function by setting $c = \sigma^{(i)}(k)$.
  4. Repeat 2-3 until $\left\|\sigma^{(i)} - \sigma^{(i-1)}\right\|$ is small enough.

Optimization and interpolation

- We discretize the state space of k by grid points:

$$k_j \in \{k_1, k_2, \cdots, k_N\},$$

where j is an index for grid points.

- Then, given $\sigma^{(i-1)}$, we solve

$$\tilde{R}(c; k_j, \sigma^{(i-1)}) \equiv -u_c(c) + \beta u_c\left(\sigma^{(i-1)}(f(k_j) - c)\right) f_k(f(k_j) - c) = 0$$

for c at each grid point k_j. That is, the residual function is equal to zero at each grid point. This is called collocation.

- We need to know the value of $\sigma^{(i-1)}(f(k_j) - c)$, which may be off the grid points.

Optimization and interpolation, cont'd

- More generally, we want to:
- solve $f(x) = 0$ for x (optimization),
- when we know only the values of $f(x_j)$ at $x_j \in \{x_1, ..., x_N\}$ (interpolation).

Polynomial function

- The policy functions are approximated by a higher-order polynomial function

$$\hat{\sigma}(x;\boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_{N-1} x^{N-1}.$$

- We need to know the values of $\sigma(x)$ (at least) at N grid points to fit the polynomial.

- Polynomial function is a global approximation.

Chebyshev polynomial

- We define the basis functions $T(x) : [-1, 1] \to [-1, 1]$, and have an univariate polynomial

$$\hat{\sigma}(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 T_1(x) + \theta_2 T_2(x) + \cdots + \theta_{N-1} T_{N-1}(x).$$

- An example of $T(x)$:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= 2x^2 - 1, \\ &\vdots \\ T_n(x) &= 2xT_{n-1}(x) - T_{n-2}(x). \end{aligned}$$

  where $x \in [-1, 1]$. These are called Chebyshev polynomials, or Chebyshev basis functions.

Transforming grid points

- For a general function which takes a value $k_j \in [k_1, k_N]$ at each grid point, we have to transform k_j to $x_j \in [-1, 1]$ to (or x_j to k_j) by applying

$$x_j = \varphi(k_j) = \frac{2(k_j - k_1)}{k_N - k_1} - 1,$$

or

$$k_j = \varphi^{-1}(x_j) = k_1 + 0.5(1 + x_j)(k_N - k_1).$$

Chebyshev collocation points

- The polynomial is evaluated at the collocation points
- Chebyshev zeros:

$$x_j = \cos\left(\frac{(2j+1)\pi}{2(N-1)}\right) \text{ for } j = 0, 1, ..., N-1.$$

- Chebyshev extrema:

$$x_j = \cos\left(\frac{j\pi}{N-1}\right) \text{ for } j = 0, 1, ..., N-1.$$

Fitting polynomial

- Once we have the collocation points $\{x_j\}$ and the function values $\{\sigma(x_j)\}$ evaluated at x_j for $j = 1, 2, ..., N$, we can fit $\hat{\sigma}(x; \theta)$ to the data to obtain θ.

$$\begin{bmatrix} \sigma(x_1) \\ \sigma(x_2) \\ \\ \sigma(x_N) \end{bmatrix} = \begin{bmatrix} 1 & T_1(x_1) & T_2(x_1) & T_{N-1}(x_1) \\ 1 & T_1(x_2) & T_2(x_2) & T_{N-1}(x_2) \\ & & & \\ 1 & T_1(x_N) & T_2(x_1) & T_{N-1}(x_N) \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \\ \theta_{N-1} \end{bmatrix},$$

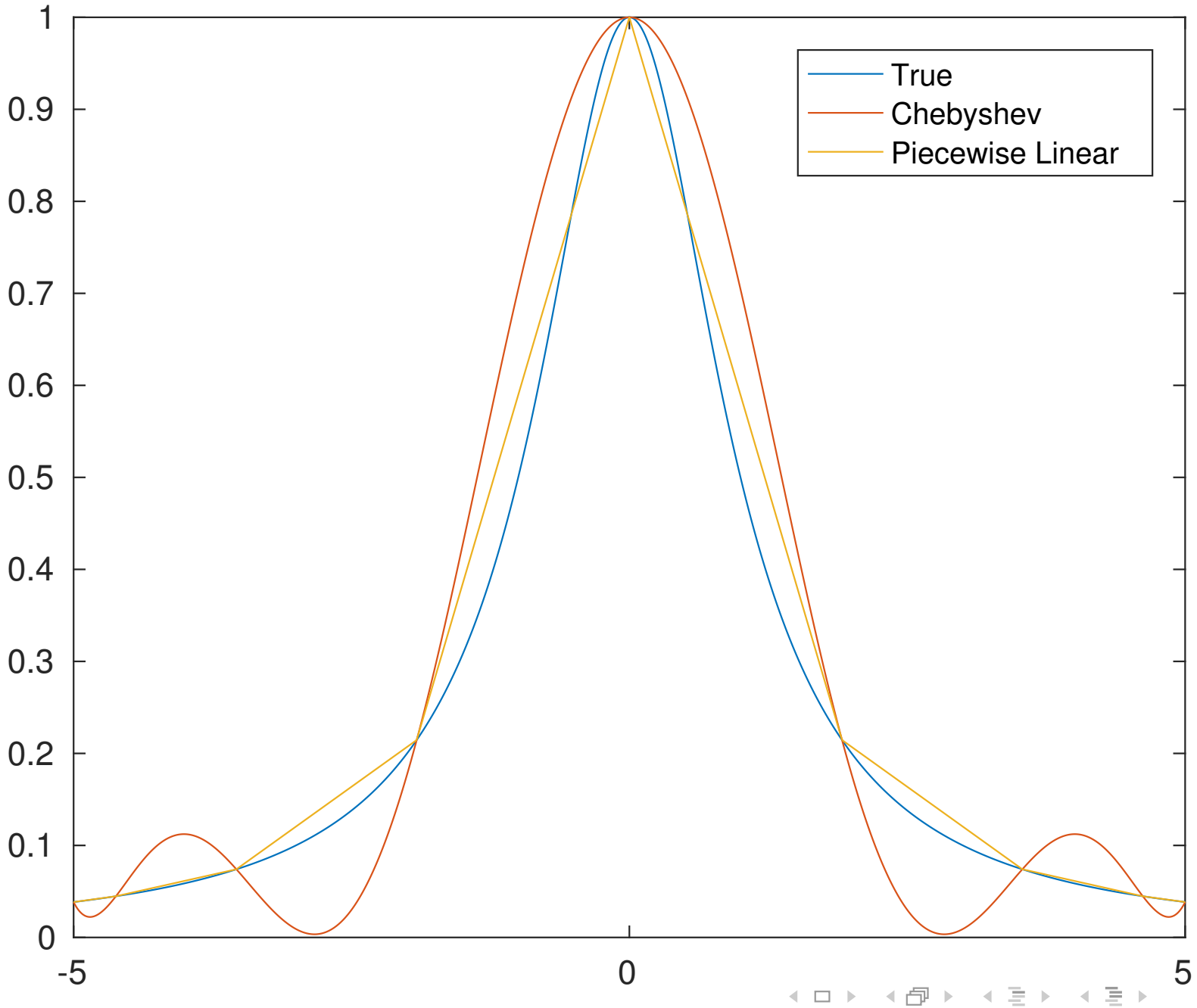
  or

$$\sigma(\boldsymbol{x}) = T(\boldsymbol{x})\boldsymbol{\theta}.$$

  Then we have $\boldsymbol{\theta} = T(\boldsymbol{x})^{-1}\sigma(\boldsymbol{x})$. $T(\boldsymbol{x})$ needs to be nonsingular.

- The basis functions with Chebyshev zeros/extrema satisfy orthogonality property. That is, each column of $T(\boldsymbol{x})$ is uncorrelated to each other.

Example: $1/(1+x^2)$ with $N = 9$



Nonlinear optimization

- Given $\hat{\sigma}^{(i-1)}(k;\boldsymbol{\theta}) = T(\varphi(k))\boldsymbol{\theta}$, we solve a nonlinear equation

$$\tilde{R}(c; k_j, \hat{\sigma}^{(i-1)}) \approx -u_c(c) + \beta u_c\left(\hat{\sigma}^{(i-1)}(f(k_j) - c;\boldsymbol{\theta})\right) f_k(f(k_j) - c) = 0$$

  for c at each grid point k_j.

- We use Newton's method to solve the nonlinear equation. For example, Matlab's command `fsolve` or Chris Sims' `csolve` does such a job.

- But, such a nonlinear optimization can be costly when the number of grid points and/or the number of nonlinear equations is large.

Parameterized expectation

- By applying a version of the parameterized expectation algorithm (PEA), we can avoid solving nonlinear equations (Maliar and Maliar, 2015; Gust et al., 2017, Hirose and Sunakawa, 2015; 2017).
  - Marcet (1988) uses a stochastic approach based on Monte Carlo simulations.
  - Christiano and Fisher (2000) propose a non-stochastic approach called PEA collocation.
  - Endogenous grid-point method (EGM) is another popular method to avoid nonlinear optimization.

PEA collocation

- There are two ways for applying PEA collocation (Christiano and Fisher, 2000):
  - One is to fit polynomials to future variables (Marcet, 1988).
  - The other is to fit polynomials to current variables (Williams and Wright, 1982a; 1982b; 1984).

Fitting future variables

- We define
$$e(k) \equiv \beta u_c\left(c'\right) f_k(k').$$
- Then, given the values of $e^{(i-1)}(k_j)$ at each grid point, we have
$$\begin{aligned} c &= u_c^{-1}(e^{(i-1)}(k_j)), \\ k' &= f(k_j) - c, \end{aligned}$$
and an intermediate policy function $c = \sigma^{(i)}(k_j)$. Note that we don't have to solve the nonlinear equation here.

Fitting future variables, cont'd

- We also update

$$e^{(i)}(k_j) = \beta u_c \left(\sigma^{(i)}(k')\right) f_k(k')$$

where k' is obtained in the previous step.

  - Note that $c' = \sigma^{(i)}(k')$, or equivalently $e^{(i-1)}(k')$, needs to be interpolated here. That is,

$$\begin{aligned} c' =& \sigma^{(i)}(k') \\ \approx& u_c^{-1}(\hat{e}^{(i-1)}(k'; \boldsymbol{\theta})) \end{aligned}$$

where $\hat{e}^{(i-1)}(k; \boldsymbol{\theta}) = \theta_0 + \theta_1 T_1(k) + \theta_2 T_2(k) + \cdots + \theta_{N-1} T_{N-1}(k)$. $\boldsymbol{\theta}$ is obtained by fitting the polynomial to the data of future variables $e^{(i-1)}(k_j)$ at each grid point k_j.

Fitting current variables

- Or, we define

$$v(k) \equiv \beta u_c\left(c\right) f_k(k),$$

- Then, given the function $v^{(i-1)}(k)$, we have

$$\begin{aligned} c &= u_c^{-1}(v^{(i-1)}(k')), \\ &\approx u_c^{-1}(\hat{v}^{(i-1)}(f(k_j) - c; \boldsymbol{\theta})) \end{aligned}$$

  - Note that $v^{(i-1)}(k)$ needs to be interpolated by using its approximation $\hat{v}^{(i-1)}(k; \boldsymbol{\theta})$. $\boldsymbol{\theta}$ is obtained by fitting the polynominal to the data of current variables $v^{(i-1)}(k_j)$ at each grid point k_j.
  - We can use a successive approximation $c = \sigma^{(i-1)}(k_j)$ to avoid nonlinear optimization.

- We also update

$$v^{(i)}(k_j) = \beta u_c\left(c\right) f_k(k_j),$$

where c is obtained in the previous step.

Part I: What is the time iteration method?

- Neoclassical growth model
  - Univariate Interpolation
  - Nonlinear optimization (or how to avoid it)
- Stochastic neoclassical growth model
  - Multivariate interpolation
  - Nonlinear optimization (or how to avoid it)
  - Numerical integration (or how to avoid it)

Stochastic neoclassical growth model

- Now we extend the earlier example with stochastic technology. An individual maximizes expected life-time utility

$$\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$c_t + k_{t+1} \leq f(k_t, z_t).$$

$\mathbb{E}_0$ is expectation operator at time 0.

- z_t follows an AR(1) process

$$z_{t+1} = \rho z_t + \epsilon_{t+1}, \quad \epsilon_{t+1} \sim N(0, \sigma_\epsilon^2)$$

Collman operator

- The first-order necessary condition is given by

$$u_c(c_t) = \beta \mathbb{E}_t \left\{ u_c(c_{t+1}) f_k(k_{t+1}, z_{t+1}) \right\}.$$

- There is a mapping $\sigma = K\sigma$ that solves

$$u_c(c) = \beta \int u_c \left(\sigma(f(k,z) - c, z') \right) f_k(f(k,z) - c, z') p(z'|z) dz'$$

for $c = \sigma(k, z)$.

Time iteration

- The time iteration method takes the following steps:
  1. Make an initial guess for the policy function $\sigma^{(0)}$.
  2. Given the policy function previously obtained $\sigma^{(i-1)}$, solve

$$u_c(c) = \beta \int u_c \left(\sigma^{(i-1)}(f(k,z) - c, z') \right) f_k(f(k,z) - c, z') p(z'|z) dz'$$

     for c.
  3. Update the policy function by setting $c = \sigma^{(i)}(k,z)$.
  4. Repeat 2-3 until $\left\| \sigma^{(i)} - \sigma^{(i-1)} \right\|$ is small enough.

Optimization, interpolation and integration

- We discretize the state space of (k, z) by grid points:

$$k_j \in \{k_1, k_2, \cdots, k_N\}, \quad z_m \in \{z_1, z_2, \cdots, z_N\},$$

  where (j, m) is an index for the set of grid points.

- Then, given $\sigma^{(i-1)}$, we solve

$$\begin{aligned}
&\tilde{R}(c; k_j, z_m, \sigma^{(i-1)}) \\
\approx& -u_c(c) \\
&+ \beta \int \left[u_c \left(\sigma^{(i-1)}(f(k_j, z_m) - c, z') \right) f_k(f(k_j, z_m) - c, z') p(z'|z_m) \right] dz' \\
=& 0
\end{aligned}$$

  for c at each grid point (k_j, z_m) (optimization).

Optimization, interpolation and integration, cont'd

- We need to know the value of $\sigma^{(i-1)}(f(k_j, z_m) - c, z')$, which may be off the grid points (interpolation).
  - $\sigma(k, z)$ is a two-dimensional object, which can be approximated by two-dimensional Chebyshev polynomial.
- Also, we compute an integral with regard to z' for the next period's expectation (integration).

2-D Chebyshev polynomial

- The policy functions are approximated by basis functions. For example, if $N_x = N_y = 3$,

$$\begin{aligned}\hat{\sigma}(x, y; \boldsymbol{\theta}) = &\theta_{0,0} + \theta_{1,0}T_1(x) + \theta_{2,0}T_2(x) + \theta_{0,1}T_1(y) + \theta_{0,2}T_2(y) \\ &+ \theta_{1,1}T_1(x)T_1(y) + \theta_{1,2}T_1(x)T_2(y) \\ &+ \theta_{2,1}T_2(x)T_1(y) + \theta_{2,2}T_2(x)T_2(y)\end{aligned}$$

  There are 9 coefficients, so we need at least $N = N_x N_y = 9$ collocation points.

- Chebyshev extrema is used as collocation points

$$\begin{aligned}(x, y) \in \{&(0,0), (-1,0), (1,0), (0,-1), (0,1), \\ &(-1,-1), (1,-1), (-1,1), (1,1)\}.\end{aligned}$$

Fitting 2-D Chebyshev polynomial

- Once we have the collocation points $\{x_i, y_j\}$ and the function values $\{\sigma(x_i, y_j)\}$ evaluated at (x_i, y_j) for $i = 1, 2, ..., N_x$ and $j = 1, 2, ..., N_y$, we can fit $\hat{\sigma}(x, y; \boldsymbol{\theta})$ to the data to obtain $\boldsymbol{\theta}$.

- For example, if $N_x = N_y = 2$,

$$\begin{bmatrix} \sigma(x_1, y_1) \\ \sigma(x_2, y_1) \\ \sigma(x_1, y_2) \\ \sigma(x_2, y_2) \end{bmatrix} = \begin{bmatrix} 1 & T_1(x_1) & T_1(y_1) & T_1(x_1)T_1(y_1) \\ 1 & T_1(x_2) & T_1(y_1) & T_1(x_2)T_1(y_1) \\ 1 & T_1(x_1) & T_1(y_2) & T_1(x_1)T_1(y_2) \\ 1 & T_1(x_2) & T_1(y_2) & T_1(x_2)T_1(y_2) \end{bmatrix} \begin{bmatrix} \theta_{0,0} \\ \theta_{1,0} \\ \theta_{1,0} \\ \theta_{1,1} \end{bmatrix}$$

  or $\sigma(\boldsymbol{x}, \boldsymbol{y}) = T(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{\theta}$. Then we have $\boldsymbol{\theta} = T(\boldsymbol{x}, \boldsymbol{y})^{-1}\sigma(\boldsymbol{x}, \boldsymbol{y})$.

- More generally, we have a tensor product for $T(\boldsymbol{x}, \boldsymbol{y}) = T(\boldsymbol{x}) \otimes T(\boldsymbol{y})$. It is costly when the dimension of the state space is large.

Approximating stochastic process

- How to compute an integral with regard to z'?
  - Tauchen's method (Rouwenhorst's method): AR(1) process is approximated by a Markov chain.
  - Gaussian Quadrature: The quadrature nodes $\{x_i\}_{i=1}^{M}$ and quadrature weights $\{w_i\}_{i=1}^{M}$ approximate

$$\int f(x)w(x)dx \approx \sum w_i f(x_i).$$

  - The total number of quadrature points is exponentially increasing in the dimension of the state space.

PEA collocation

- There are two ways for applying PEA collocation (Christiano and Fisher, 2000):
  - One is to fit polynomials to future variables (Marcet, 1988).
  - The other is to fit polynomials to current variables (Williams and Wright, 1982a; 1982b; 1984)
  - In the latter approach, we can also avoid computing numerical integration in the expectation terms by precomputation technique of Judd, Maliar, Maliar and Tsener (2017). "PEA collocation meets precomputing integrals."

Numerical examples I

- Solve the stochastic neoclassical model by the time iteration method
  - Interpolation: Chebyshev polynomial (either $N_d = 3$ or 5 for each $d \in \{k, z\}$).
  - Optimization:
    - Chris Sims' csolve is used in Time iteration with Newton's method (TI).
    - No optimization is required in PEA collocation.
  - Integration:
    - Gaussian-Hermite quadrature is used in TI and PEA collocation with fitting future variables (future PEA).
    - Precomputation technique in Judd et al. (2017) is used in PEA collocation with fitting current variables (current PEA).

Euler equation errors

- Accuracy and computation speed are compared. Look at the Euler equation errors:

$$\mathcal{E}(k,z) \equiv 1 - \beta \int \left\{ \left(\frac{\sigma_c(k',z')}{\sigma_c(k,z)} \right)^{-\tau} \left(1 - \delta + \alpha z' k'^{\alpha-1}\right) \right\} p(z'|z) dz'$$

  where $k' = f(k,z) - \sigma_c(k,z)$.

Summary

(N_d, τ)	TI			future PEA			current PEA		
	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
$(3, 1.0)$	-5.12	-4.60	4.03	-4.23	-3.69	0.04	-3.13	-2.44	0.02
$(5, 1.0)$	-7.08	-6.72	9.76	-5.92	-5.59	0.09	-3.13	-2.44	0.04
$(3, 2.0)$	-4.82	-4.35	0.86	-3.99	-3.53	0.03	-2.95	-2.26	0.01
$(5, 2.0)$	-6.76	-6.45	2.88	-5.63	-5.36	0.11	-2.96	-2.27	0.04
$(3, 5.0)$	-4.48	-3.87	0.62	-3.57	-2.88	0.05	-2.67	-1.99	0.02
$(5, 5.0)$	-6.43	-5.38	1.91	-5.10	-3.90	0.15	-2.69	-2.00	0.05

Notes: L_1 and L_∞ are, respectively, the average and maximum of absolute Euler errors (**??**) (in log 10 units) on a 10,000 period stochastic simulation. CPU is the elapsed time for computing equilibrium (in seconds).

Part II: Applications to New Keynesian models

- We solve small-scale nonlinear New Keynesian DSGE model with
  - Smolyak's method with sparse grid points
  - Simulation-based method with EDS grid
  - The techniques we mentioned earlier (PEA collocation and precomputing integrals) are also applied.

Time iteration in the New Keynesian literature

- Time iteration is a popular method to solve nonlinear New Keynesian models.
  - We have to look at the decentralized economy, as the second welfare theorem fails to hold.
  - An incomplete list includes: Fernández-Villaverde, Gordon, Guerrón-Quintana, and Rubio-Ramírez, 2015; Maliar and Maliar, 2015; Gavin, Keen, Richter, and Throckmorton, 2015; Gust, Herbst, López-Salido, and Smith, 2017; Iiboshi, Ueda, and Shintani, 2018; Nakata, 2016a, 2016b; Hills, Nakata, and Schmitt, 2016; Dennis, 2016; Ngo, 2014; Hirose and Sunakawa, 2015, 2017; Hills, Nakata, and Sunakawa, 2018.

A small scale New Keynesian DSGE model

- The example here is taken from An and Schorfheide (2007) and Herbst and Schorfheide (2016).
- The model economy consists of
  - Final-good and intermediate-good producing firms
  - Households
  - Monetary and fiscal authorities
- Prices are sticky due to Rotemberg-type (1982) adjustment cost.

Model overview

- Equilibrium conditions (after detrending):

$$
\begin{aligned}
1 &= \beta \mathbb{E}_t \left[\left(\frac{c_{t+1}}{c_t} \right)^{-\tau} \frac{R_t}{\gamma_{t+1} \pi_{t+1}} \right] \\
0 &= (1 - \nu^{-1}) + \nu^{-1} \chi_H c_t^{\tau} - \phi (\pi_t - \bar{\pi}) \left[\pi_t - \frac{1}{2\nu} (\pi_t - \bar{\pi}) \right] \\
&\quad + \beta \phi \mathbb{E}_t \left[\left(\frac{c_{t+1}}{c_t} \right)^{-\tau} \frac{y_{t+1}}{y_t} (\pi_{t+1} - \bar{\pi}) \pi_{t+1} \right], \\
R_t^* &= \left(r \bar{\pi} \left(\frac{\pi}{\bar{\pi}} \right)^{\psi_1} \left(\frac{y_t}{y_t^*} \right)^{\psi_2} \right)^{1-\rho_R} R_{t-1}^{*\rho_R} e^{\epsilon_{R,t}}, \\
c_t &+ \frac{\phi}{2} (\pi_t - \pi)^2 y_t = g_t^{-1} y_t, \\
R_t &= \max \{ R_t^*, 1 \}.
\end{aligned}
$$

There are 5 equations and 5 endogenous variables $\{c_t, \pi_t, R_t^*, R_t, y_t\}$ and 3 exogenous variables $\{\gamma_t, g_t, \epsilon_{R,t}\}$. The natural level of output is given by $y_t^* = (1 - \nu)^{1/\tau} g_t$.

Model overview, cont'd

- A_t has a deterministic trend $\bar{\gamma}$ and a shock to the trend z_t such as $\ln \gamma_t \equiv \ln(A_t/A_{t-1}) = \ln \bar{\gamma} + \ln z_t$. $\{z_t, g_t\}$ follow

$$\ln z_t = \rho_z \ln z_{t-1} + \epsilon_{z,t},$$
$$\ln g_t = (1 - \rho_g) \ln \bar{g} + \rho_g \ln g_{t-1} + \epsilon_{g,t}.$$

- The solution has a form of

$$c = \sigma_c(R^*_{-1}, s), \quad \pi = \sigma_\pi(R^*_{-1}, s),$$
$$R^* = \sigma_{R^*}(R^*_{-1}, s), \quad y = \sigma_y(R^*_{-1}, s),$$

where $s = (\gamma, g, \epsilon_R)$. Note that $R = \max\{\sigma_{R^*}(R^*_{-1}, s), 1\}$.

Collman operator

- The mapping $\sigma = K\sigma$ solves

$$
\begin{aligned}
0 &= -c^{-\tau} + \beta R \int \left[\frac{\sigma_c(R, s')^{-\tau}}{\gamma' \sigma_\pi(R, s')} \right] p(s'|s) ds', \\
0 &= \left((1 - \nu^{-1}) + \nu^{-1} c^{\tau} - \phi \left(\pi - \bar{\pi} \right) \left[\pi - \frac{1}{2\nu} \left(\pi - \bar{\pi} \right) \right] \right) c^{-\tau} y \\
&+ \beta \phi \int \left[\sigma_c(R, s')^{-\tau} \sigma_y(R, s') \left(\sigma_\pi(R, s') - \bar{\pi} \right) \sigma_\pi(R, s') \right] p(s'|s) ds', \\
R &= \left(r \bar{\pi} \left(\frac{\pi}{\bar{\pi}} \right)^{\psi_1} \left(\frac{y}{y^*} \right)^{\psi_2} \right)^{1-\rho_R} R_{-1}^{\rho_R} e^{\epsilon_R}, \\
c &+ \frac{\phi}{2} \left(\pi - \bar{\pi} \right)^2 y = g^{-1} y. \\
R &= \max \left\{ R^*, 1 \right\},
\end{aligned}
$$

  for c, π, R^*, R, y.

Time iteration

- The time iteration method takes the following steps:
  1. Make an initial guess for the policy function $\sigma^{(0)}$.
  2. Given the policy function previously obtained $\sigma^{(i-1)}$, solve the relevant equations for (c, π, R^*, y).
  3. Update the policy function by setting $c = \sigma_c^{(i)}(R^*_{-1}, s)$, $\pi = \sigma_\pi^{(i)}(R^*_{-1}, s)$, $R^* = \sigma_{R^*}^{(i)}(R^*_{-1}, s)$, and $y = \sigma_y^{(i)}(R^*_{-1}, s)$.
  4. Repeat 2-3 until $\left\| \sigma^{(i)} - \sigma^{(i-1)} \right\|$ is small enough.

Optimization, interpolation, and integration

- Here, we need to solve the system of nonlinear equations (optimization).
  - PEA collocation can also be used here to avoid costly nonlinear optimization.
- Also, we need to evaluate the function $\sigma(R^*, s')$ off the grid points (interpolation).
  - $\sigma(R^*_{-1}, s)$ is a high dimensional object (our model have 4 state variables, $(R^*_{-1}, s) = (R^*_{-1}, \gamma, g, \epsilon_R)$), which can be dealt with Smolyak's method with sparse grid points (Fernandez-Villaverde et al., 2015) or simulation based method with EDS grid (Maliar and Maliar, 2015).
- We compute an integral wrt s' for the next period's expectation (integration).
  - Precomputing integrals can also be applied here to avoid numerical integration.

Smolyak's method

- We introduce Smolyak's (1963) method with sparse grid points to handle such a high-dimensional object.
  - Krueger and Kubler (2004) first introduced Smolyak sparse grid points to solve heterogeneous OLG models with aggregate uncertainty.
  - Applications of Smolyak's method to New Keynesian models are found in Fernandez-Villaverde et al., (2015), Gust et al. (2017), Hirose and Sunakawa (2015; 2017).
- We look at simple cases with second-order polynomials with $N_d = 3$ for each dimension.
  - Cases with higher-order polynomials (for example $N_d = 5$ or 9) are a bit more complicated but manageable. See Judd, Maliar, Maliar and Valero (2014).

Smolyak's method: Simple cases

- The policy functions are approximated by basis functions. For example, if we use second-order polynomials and $N_x = N_y = 3$,

$$\hat{\sigma}(x, y; \boldsymbol{\theta}) = \theta_{0,0} + \theta_{1,0} T_1(x) + \theta_{2,0} T_2(x) + \theta_{0,1} T_1(y) + \theta_{0,2} T_2(y).$$

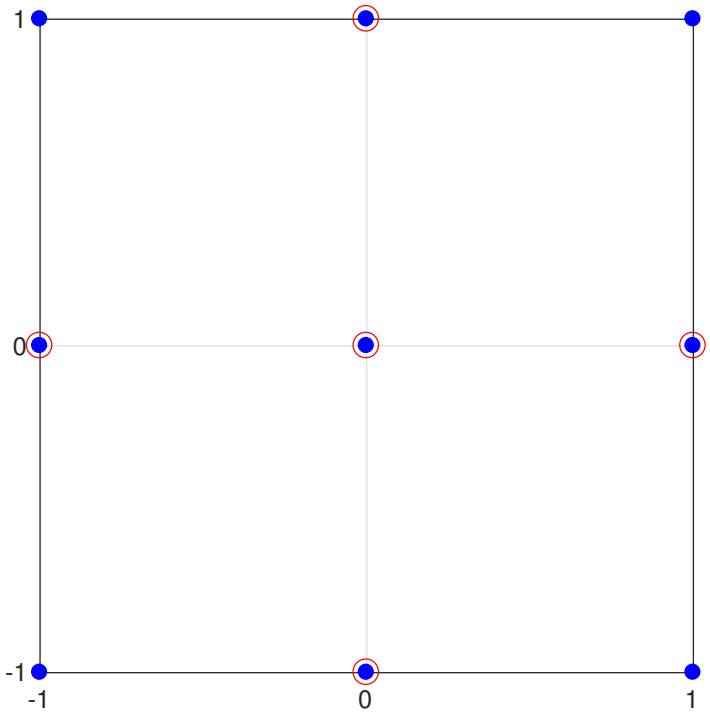
  There are 5 coefficients, so we need 5 collocation points. We have just eliminated all the cross terms!

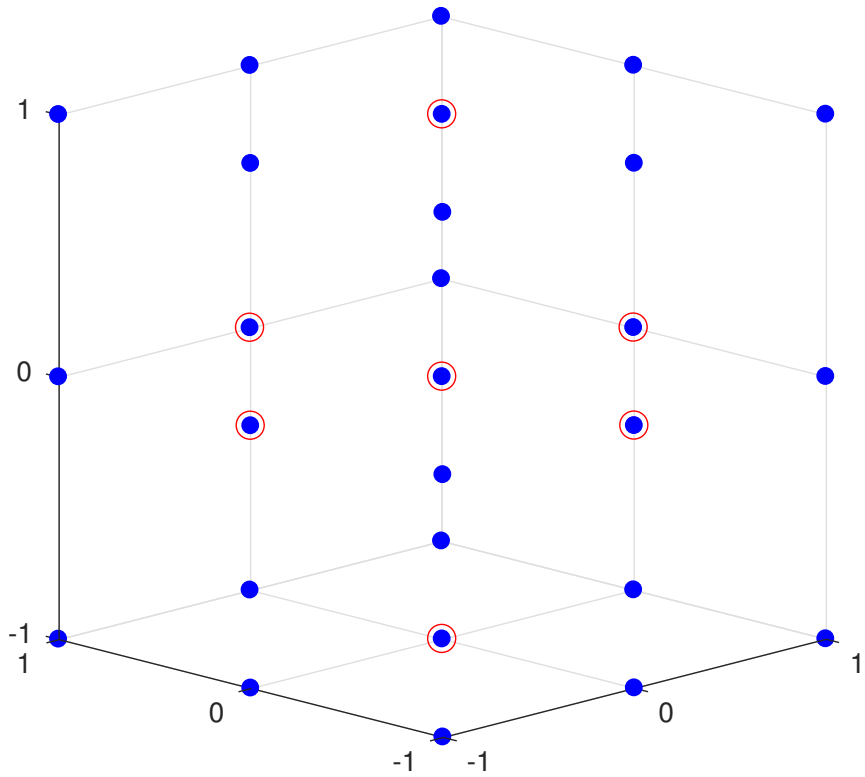
Simple cases with second-order polynomials

- Chebyshev extrema is used as collocation points:

$$N_d = 2: \quad (x, y) \in \{(0,0), (1,0), (-1,0), (0,1), (0,-1)\}$$

$$N_d = 3: \quad (x, y, z) \in \begin{array}{c} \{(0,0,0), (1,0,0), (-1,0,0), \\ (0,1,0), (0,-1,0), (0,0,1), (0,0,-1)\} \end{array}$$





Smolyak's method: Simple cases, cont'd

- The total number of the grid points is $1 + 2N_d$, whereas it is 3^{N_d} with the standard Chebyshev polynomials.

N_d	$1 + 2N_d$	3^{N_d}
2	5	9
3	7	27
4	9	81
5	11	243
⋮	⋮	⋮
10	21	59,049
20	41	3,486,784,401

Simulation-based method

- We solve for the policy functions on simulated grid points based on ergodic distribution of the state variables.
  - Judd, Maliar and Maliar (2011) and Maliar and Maliar (2015, MM hereafter) developed simulation-based method, based on the original work of Marcet's (1988) parameterized expectation algorithm.
  - Applications to New Keynesian models are found in Maliar and Maliar (2015), Lepetuyk, Maliar, and Maliar (2017), Aruoba, Cuba-borda, and Schorfheide (2018), and Hills, Nakata, and Sunakawa (2018).

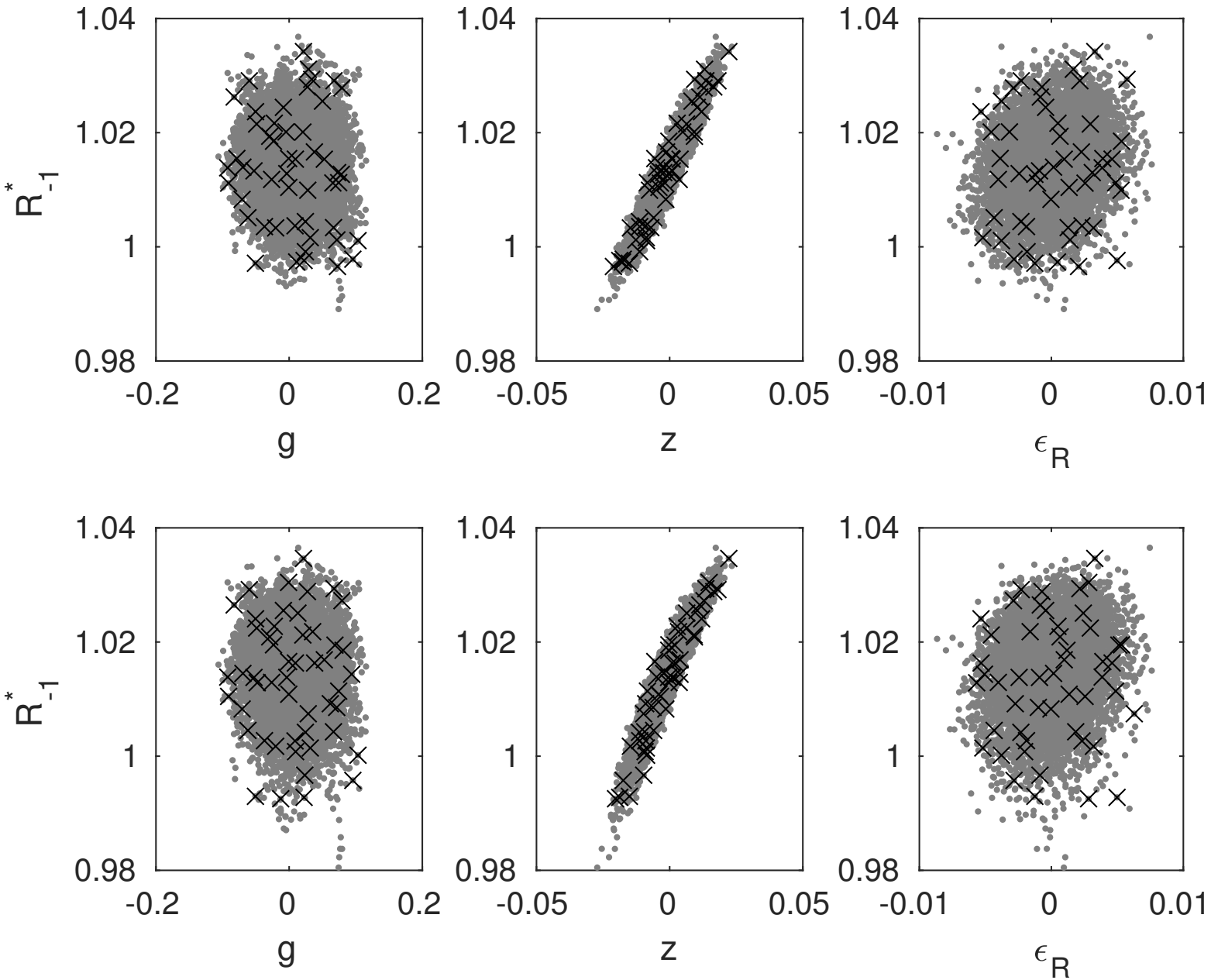
Constructing EDS grid

- In constructing an EDS grid from the ergodic set, we do the following two step procedure (See MM for more details):

1. Selecting points within an essentially ergodic set (called Algorithm $\mathcal{A}^{\eta}$ in MM)

2. Constructing a uniformly spaced set of points that covers the essentially ergodic set (called Algorithm P^{ϵ} in MM)

Constructing EDS grid



Numerical examples II

- Solve the nonlinear NK model by the time iteration method
  - Interpolation:
    - Non-stochastic method: Chebyshev polynomial with Smolyak sparse grid points (in the latter two). $(N_d, N) = (3, 81), (3, 9), (5, 41)$.
    - Simulation-based method: Second-order polynomials with cross terms. The number of grid points $N = 25, 50$ or 100 EDS grid points. As we have 4 state variables, the number of coefficients is 15.
  - Optimization:
    - Chris Sims' csolve is used in Time iteration with Newton's method (TI).
    - No optimization is required in PEA collocation.
  - Integration:
    - Gaussian-Hermite quadrature is used with $M = 3^3 = 27$ in TI and PEA collocation with fitting future variables (future PEA).
    - Precomputation technique in Judd et al. (2017) is used in PEA collocation with fitting current variables (current PEA, only with non-stochastic method).

Non-stochastic method with ZLB

(N_d, N)	TI								
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	$\text{Pr}_{\textbf{ZLB}}$	CPU
$(3, 81)$	-3.73	-2.62	-2.07	-1.42	0.76	2.05	2.53	1.53	1127.3
$(3, 9)$	-3.40	-2.38	-2.06	-1.09	0.76	2.02	2.50	1.79	12.98
$(5, 41)$	-3.97	-3.14	-2.07	-1.73	0.76	2.04	2.50	1.40	270.65

(N_d, N)	future PEA								
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	$\text{Pr}_{\textbf{ZLB}}$	CPU
$(3, 81)$	-3.73	-2.67	-2.08	-1.44	0.76	2.11	2.59	1.71	82.28
$(3, 9)$	-3.26	-2.66	-1.92	-1.48	0.76	2.19	2.68	3.42	0.96
$(5, 41)$	-4.04	-3.54	-2.13	-1.49	0.76	2.03	2.48	1.18	14.66

(N_d, N)	current PEA								
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	$\text{Pr}_{\textbf{ZLB}}$	CPU
$(3, 81)$	-4.05	-2.71	-2.12	-1.53	0.76	2.01	2.45	1.03	4.05
$(3, 9)$	-3.35	-2.42	-1.97	-1.25	0.76	2.01	2.47	1.92	0.19
$(5, 41)$	-4.17	-2.95	-2.12	-1.44	0.76	2.03	2.47	1.07	0.99

Notes: $L_{1,c}$, $L_{1,\pi}$, $L_{\infty,c}$, and $L_{\infty,\pi}$ are, respectively, the average and maximum of absolute Euler errors (??)-(??) (in log 10 units) on a 10,000 period stochastic simulation. CPU is the elapsed time for computing equilibrium (in seconds). $\sigma_{\Delta y}$, σ_π, and σ_R are the standard deviation of output growth, inflation, and the policy rate.

Simulation-based method with ZLB

N	TI								
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	$\text{Pr}_{\textbf{ZLB}}$	CPU
25	-3.39	-2.93	-2.21	-1.66	0.77	2.14	2.63	1.21	44.41
50	-3.00	-2.56	-1.83	-1.48	0.77	2.21	2.71	1.92	65.37
100	-3.28	-2.80	-2.08	-1.69	0.76	2.10	2.57	1.25	106.17

N	future PEA								
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	$\text{Pr}_{\textbf{ZLB}}$	CPU
25	-2.44	-1.24	-1.58	-0.61	0.77	2.18	2.68	1.31	3.37
50	-2.44	-1.24	-1.69	-0.63	0.77	2.15	2.63	1.46	5.08
100	-2.46	-1.26	-1.71	-0.67	0.76	2.11	2.58	1.31	7.93

Notes: $L_{1,c}$, $L_{1,\pi}$, $L_{\infty,c}$, and $L_{\infty,\pi}$ are, respectively, the average and maximum of absolute Euler errors (**??**)-(**??**) (in log 10 units) on a 10,000 period stochastic simulation. CPU is the elapsed time for computing equilibrium (in seconds). $\sigma_{\Delta y}$, σ_π, and σ_R are the standard deviation of output growth, inflation, and the policy rate.

Backups

Gaussian quadrature

- (Miranda and Fackler, 2002, p. 88) The quadrature nodes $\{x_i\}_{i=1}^n$ and quadrature weights $\{w_i\}_{i=1}^n$ are chosen to satisfy the $2n$ "moment matching" conditions

$$\int x^k w(x)dx = \sum_{i=1}^{n} w_i x_i^k,$$

  for $k = 0, ..., 2n-1$.

- The integral approximation is then computed

$$\int f(x)w(x)dx \approx \sum w_i f(x_i).$$

- By construction, an n point Gaussian quadrature is order $2n-1$ exact. That is, if f can be approximated by a polynomial, Gaussian quadrature is a good approximation.

Fitting future variables

- We define

$$e(k,z) \equiv \beta \int u_c\left(c'\right) f_k(k',z') p(z'|z) dz'.$$

- Then, given the values of $e^{(i-1)}(k_j, z_m)$ at each grid point, we have

$$\begin{aligned} c &= u_c^{-1}(e^{(i-1)}(k_j, z_m)), \\ k' &= f(k_j, z_m) - c, \end{aligned}$$

and an intermediate policy function $c = \sigma^{(i)}(k_j, z_m)$. Note that we don't have to solve the nonlinear equation here.

Fitting future variables, cont'd

- We also update

$$e^{(i)}(k_j, z_m) = \beta \int u_c \left(\sigma^{(i)}(k', z') \right) f_k(k', z') p(z'|z_m) dz'$$

where k' is obtained in the previous step.

  - Note that $c' = \sigma^{(i)}(k', z')$, or equivalently $e^{(i-1)}(k', z')$, needs to be interpolated here:

$$\begin{aligned} c' =& \sigma^{(i)}(k', z') \\ \approx& u_c^{-1}(\hat{e}^{(i-1)}(k', z'; \boldsymbol{\theta})). \end{aligned}$$

  - Also, we compute integral of a composite function

$$\int u_c \left(u_c^{-1} \left(\hat{e}^{(i-1)}(k', z'; \boldsymbol{\theta}) \right) \right) f_k(k', z') p(z'|z_m) dz'.$$

  We use Gaussian-Hermite quadrature.

Fitting current variables

- Or, we define

$$v(k,z) \equiv \beta u_c\left(c\right) f_k(k,z)$$

- Then, we have

$$\begin{aligned} c &= u_c^{-1}\left(\int v^{(i-1)}(k',z')p(z'|z_m)dz'\right), \\ &\approx u_c^{-1}\left(\int \hat{v}^{(i-1)}(k',z';\boldsymbol{\theta})p(z'|z_m)dz'\right) \end{aligned}$$

  - Note that $v^{(i-1)}(k',z')$ needs to be interpolated here by $\hat{v}^{(i-1)}(k',z';\boldsymbol{\theta})$.
  - By a successive approximation with $k' = f(k,z) - \sigma^{(i-1)}(k,z)$, we can avoid nonlinear optimization.
  - Also, we can avoid computing numerical integration of $\hat{v}^{(i-1)}(k',z';\boldsymbol{\theta})$ by computing integral analytically just once, as is explained below.

Fitting current variables

- We also update

$$v^{(i)}(k_j, z_m) \equiv \beta u_c(c) f_k(k_j, z_m)$$

where c is obtained in the previous step.

- Note that

$$e(k, z) = \int v(f(k, z) - \sigma(k, z), z')p(z'|z)dz'$$

holds.

Precomputation of integrals

- Judd et al. (2018) pointed out that we can use their precomputation technique only when we express the function to be integrated by a simple parameterized form.
  - This is exactly the case with fitting current variables with $v(k, z)$.
  - In contrast, in the case with fitting future variables with $e(k, z)$, we have to compute an integral of a composite function

$$u_c\left(u_c^{-1}\left(\hat{e}^{(i-1)}(k', z'; \boldsymbol{\theta})\right)\right) f_k(k', z').$$

Precomputation of integrals, cont'd

- For example, we approximate v by a second-order polynomial:

$$\hat{v}(k, z; \boldsymbol{\theta}) = \theta_{0,0} + \theta_{1,0}k + \theta_{2,0}k^2 + \theta_{0,1}z + \theta_{0,2}z^2.$$

- Then,

$$\begin{aligned}
&\int \hat{v}(k', z'; \boldsymbol{\theta}) p(z'|z) dz' \\
=&\theta_{0,0} + \theta_{1,0}k' + \theta_{2,0}\left(k'\right)^2 + \int \left(\theta_{0,1}z' + \theta_{0,2}(z')^2\right) p(z'|z) dz' \\
=&\theta_{0,0} + \theta_{1,0}k' + \theta_{2,0}\left(k'\right)^2 + \int \left(\theta_{0,1}(\rho z + \epsilon') + \theta_{0,2}(\rho z + \epsilon')^2\right) p(\epsilon') d\epsilon' \\
=&\theta_{0,0} + \theta_{1,0}k' + \theta_{2,0}\left(k'\right)^2 + \theta_{0,1}\rho z + \theta_{0,2}\rho^2 z^2 + \theta_{0,2}\sigma_\epsilon^2.
\end{aligned}$$

Time iteration: Initial guess

- A guess of the policy functions

$$c = \sigma_c^{(0)}(R^*_{-1}, s), \quad \pi = \sigma_\pi^{(0)}(R^*_{-1}, s),$$
$$R^* = \sigma_{R^*}^{(0)}(R^*_{-1}, s), \quad y = \sigma_y^{(0)}(R^*_{-1}, s),$$

- We know the values of the functions only at each *grid point*, e.g.,

$$\sigma_c^{(0)}(R^*_{-1}, s_m) = [c_{1m}, c_{2m}, ..., c_{Nm}]',$$
$$\sigma_\pi^{(0)}(R^*_{-1}, s_m) = [\pi_{1m}, \pi_{2m}, ..., \pi_{Nm}]',$$
$$\sigma_{R^*}^{(0)}(R^*_{-1}, s_m) = [R^*_{1m}, R^*_{2m}, ..., R^*_{Nm}]',$$
$$\sigma_y^{(0)}(R^*_{-1}, s_m) = [y_{1m}, y_{2m}, ..., y_{Nm}]',$$

  for $m = 1, ..., N_s$.

- The solution for log-linearized version of the model can be used as an initial guess.

Time iteration: Solving

- Given the policy function previously obtained $\sigma^{(i-1)}$, at each grid point (j, m), having the values of $(R_{i,-1}, s_k)$ at hand, we solve

$$c_{jm}^{-\tau} = \frac{\beta}{\bar{\gamma}} R_{jm} \int \left[\frac{\sigma_c^{(i-1)}(R_{jm}, s')^{-\tau}}{z' \sigma_\pi^{(i-1)}(R_{jm}, s')} \right] p(s'|s) ds',$$

$$0 = \left((1 - \nu^{-1}) + \nu^{-1} c_{jm}^{\tau} - \phi \left(\pi_{jm} - \bar{\pi} \right) \left[\pi_{jm} - \frac{1}{2\nu} \left(\pi_{jm} - \bar{\pi} \right) \right] \right) c_{jm}^{-\tau} y_{jm}$$

$$+ \beta\phi \int \left[\sigma_c^{(i-1)}(R_{jm}, s')^{-\tau} \sigma_y^{(i-1)}(R_{jm}, s') \left(\sigma_\pi^{(i-1)}(R_{jm}, s') - \bar{\pi} \right) \sigma_\pi^{(i-1)}(R_{jm}, s') \right] p(s'|s) ds',$$

$$R_{jm} = \left(r\bar{\pi} \left(\frac{\pi_{jm}}{\bar{\pi}} \right)^{\psi_1} \left(\frac{y_{jm}}{y^*} \right)^{\psi_2} \right)^{1-\rho_R} R_{j,-1}^{\rho_R} e^{\epsilon_{R,m}},$$

$$c_{jm} + \frac{\phi}{2} \left(\pi_{jm} - \bar{\pi} \right)^2 y_{jm} = g_m^{-1} y_{jm}.$$

$$R_{jm} = \max \left\{ R_{jm}^*, 1 \right\},$$

for $c_{jm}, \pi_{jm}, R_{jm}^*, R_{jm}, y_{jm}$.

Time iteration: Updating

- Once this is done for all the grid points, we update

$$\sigma_c^{(i)}(R^*_{-1}, s_m) = [c_{1m}, c_{2m}, ..., c_{Nm}]',$$
$$\sigma_\pi^{(i)}(R^*_{-1}, s_m) = [\pi_{1m}, \pi_{2m}, ..., \pi_{Nm}]',$$
$$\sigma_{R^*}^{(i)}(R^*_{-1}, s_m) = [R^*_{1m}, R^*_{2m}, ..., R^*_{Nm}]',$$
$$\sigma_y^{(i)}(R^*_{-1}, s_m) = [y_{1m}, y_{2m}, ..., y_{Nm}]',$$

  for $m = 1, ..., N_s$.

- We repeat the procedure until the policy functions converge, i.e., $\left\| \sigma_x^{(i)}(R^*_{-1}, s) - \sigma_x^{(i-1)}(R^*_{-1}, s) \right\| < \epsilon$ for $x \in \{c, \pi, R^*, y\}$.

Simulation-based method: Algorithm

- We merge the simulation-based sparse grid and the time iteration method by the following steps:

0. Initialization
   1. Choose initial values (R^*_{-1}, s_0) and simulation length, T.
   2. Draw a sequence of $\{s_t\}_{t=1}^T$ and fix the sequence throughout the iterations.
   3. Choose approximating policy functions $\hat{\sigma}(R^*_{-1}, s; \boldsymbol{\theta})$ and make an initial guess of $\boldsymbol{\theta}$. We use second-order polynomials with cross terms for $\hat{\sigma}$.
1. Construction of an EDS grid
   1. Given (R^*_{-1}, s_0) and $\{s_t\}_{t=1}^T$, use $\hat{\sigma}(R^*_{-1}, s; \boldsymbol{\theta})$ to simulate $\{R^*_{t-1}\}_{t=1}^T$.
   2. Construct an EDS grid $\Gamma \equiv \left\{R^*_{-1,m}, s_m\right\}_{m=1}^M$ from the simulated sequence of $\{R^*_{t-1}, s_t\}_{t=1}^T$.

Simulation-based method: Algorithm, cont'd

2. Computation of a solution on the EDS grid, $\hat{\sigma}(R^*_{-1}, s; \boldsymbol{\theta})$, using the time iteration method.

3. Repeat 2-3 until convergence of the EDS grid.

Euler equation errors

- Accuracy and computation speed are compared. Look at the Euler equation errors:

$$\begin{aligned}
\mathcal{E}_c(R_{-1}, s) =& 1 - \frac{\beta}{\bar{\gamma}} R \int \left\{ \left(\frac{\sigma_c(R, s')}{\sigma_c(R_{-1}, s)} \right)^{-\tau} \frac{1}{z' \sigma_\pi(R, s')} \right\} p(s'|s) ds', \\
\mathcal{E}_\pi(R_{-1}, s) =& (1 - \nu^{-1}) + \nu^{-1} \sigma_c(R_{-1}, s)^{-\tau} \\
& - \phi \left(\sigma_\pi(R_{-1}, s) - \bar{\pi} \right) \left[\sigma_\pi(R_{-1}, s) - \frac{1}{2\nu} \left(\sigma_\pi(R_{-1}, s) - \bar{\pi} \right) \right] \\
& + \beta \phi \int \left\{ \left(\frac{\sigma_c(R, s')}{\sigma_c(R_{-1}, s)} \right)^{-\tau} \frac{y'}{y} \left(\sigma_\pi(R, s') - \bar{\pi} \right) \sigma_\pi(R, s') \right\} p(s'|s) ds',
\end{aligned}$$

where

$$\begin{aligned}
y &= \left(g^{-1} - \frac{\phi}{2} \left(\sigma_\pi(R_{-1}, s) - \bar{\pi} \right)^2 \right)^{-1} \sigma_c(R_{-1}, s), \\
R &= \left(\bar{r} \bar{\pi} \left(\frac{\sigma_\pi(R_{-1}, s)}{\bar{\pi}} \right)^{\psi_1} \left(\frac{y}{y^*} \right)^{\psi_2} \right)^{1-\rho_R} R_{-1}^{\rho_R} e^{\epsilon_R}.
\end{aligned}$$

Parameter values

Parameter		Value
ν	Inverse of demand elasticity	1/6
$\bar{g}$	Steady state government expenditure	1.25
γ	Steady state technology growth	1.0052
β	Discount factor	0.9990
$\bar{\pi}$	Steady state inflation	1.0083
τ	CRRA parameter	2.83
ϕ	Price adjustment cost	17.85
ψ_1	Interest rate elasticity to inflation	1.80
ψ_2	Interest rate elasticity to output gap	0.63
ρ_r	Interest rate smoothing	0.77
ρ_g	Persistence of government shock	0.98
ρ_z	Persistence of technology growth shock	0.88
σ_r	Std. dev. of monetary policy shock	0.0022
σ_g	Std. dev. of government shock	0.0071
σ_z	Std. dev. of technology growth shock	0.0031

Notes: Taken from Schorfheide and Herbst. Estimation period is ....

Non-stochastic method without ZLB

(N_d, N)	TI							
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	CPU
$(3, 81)$	-4.15	-2.45	-3.47	-1.79	0.76	1.93	2.36	318.07
$(3, 9)$	-3.45	-2.35	-2.34	-1.31	0.76	1.97	2.42	3.71
$(5, 41)$	-5.09	-3.73	-3.72	-2.57	0.76	1.99	2.43	61.94

(N_d, N)	future PEA							
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	CPU
$(3, 81)$	-4.86	-2.76	-3.52	-2.04	0.76	2.01	2.46	20.90
$(3, 9)$	-3.32	-2.66	-2.21	-1.63	0.76	2.13	2.59	0.29
$(5, 41)$	-5.03	-3.69	-3.71	-2.69	0.76	2.00	2.44	4.06

(N_d, N)	current PEA							
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	CPU
$(3, 81)$	-4.32	-2.46	-3.30	-1.75	0.76	1.92	2.35	1.39
$(3, 9)$	-3.36	-2.36	-2.21	-1.34	0.76	1.98	2.43	0.04
$(5, 41)$	-4.93	-3.75	-3.56	-2.53	0.76	1.99	2.43	0.32

Notes: $L_{1,c}$, $L_{1,\pi}$, $L_{\infty,c}$, and $L_{\infty,\pi}$ are, respectively, the average and maximum of absolute Euler errors (??)-(??) (in log 10 units) on a 10,000 period stochastic simulation. CPU is the elapsed time for computing equilibrium (in seconds). $\sigma_{\Delta y}$, σ_π, and σ_R are the standard deviation of output growth, inflation, and the policy rate.

Simulation-based method without ZLB

M	TI							
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	CPU
25	-5.08	-4.09	-3.94	-2.84	0.76	1.99	2.43	7.48
50	-5.02	-4.20	-3.77	-2.75	0.76	2.00	2.44	18.73
100	-5.20	-4.25	-3.82	-2.81	0.76	2.00	2.44	28.61

M	future PEA							
	$L_{1,c}$	$L_{1,\pi}$	$L_{\infty,c}$	$L_{\infty,\pi}$	$\sigma_{\Delta y}$	σ_π	σ_R	CPU
25	-5.07	-3.74	-3.90	-2.69	0.76	1.99	2.43	1.59
50	-4.94	-3.73	-3.95	-2.62	0.76	2.00	2.44	2.32
100	-5.08	-3.76	-3.91	-2.67	0.76	2.00	2.44	3.99

Notes: $L_{1,c}$, $L_{1,\pi}$, $L_{\infty,c}$, and $L_{\infty,\pi}$ are, respectively, the average and maximum of absolute Euler errors (**??**)-(**??**) (in log 10 units) on a 10,000 period stochastic simulation. CPU is the elapsed time for computing equilibrium (in seconds). $\sigma_{\Delta y}$, σ_π, and σ_R are the standard deviation of output growth, inflation, and the policy rate.