Tutorial Session

Solution and Estimation Methods for Nonlinear DSGE Models

Yasuo Hirose Takeki Sunakawa

Keio University Kobe University

Introduction

- Dynamic stochastic general equilibrium (DSGE) models have been one of the primary tools in macroeconomic analysis.
  - Microfoundation
  - Immune to the Lucas Critique
  - Suitable for welfare analysis
  - Rational expectations
- Following the development of Bayesian estimation techniques, many economists have estimated DSGE models.
  - Cross-equation restrictions
  - Avoid weak-instrument problems that would arise in GMM estimation.
  - Priors for structural parameters
    - Deal with identification issues.
    - Estimate parameters with bounded domains.
    - Make the likelihood function well-shaped.

Introduction

- While DSGE models are inherently nonlinear, the nonlinearities are often small.
- Equilibrium conditions are linearly approximated.
  - Rational expectations solution (policy function) can be obtained using a standard solution method.
    - Blanchard and Kahn (1980); Sims (2002)
  - Likelihood can be evaluated using the Kalman filter.
    - Additional assumption: Normality of shocks

Steps for Estimating Linearized DSGE Models

- DSGE model:

$$\Gamma_0(\theta)s_t = \Gamma_1(\theta)s_{t-1} + \Psi_0(\theta)\varepsilon_t + \Pi_0(\theta)\eta_t$$

- Rational expectations solution (state transition equations):

$$s_t = \Phi_1(\theta)s_{t-1} + \Phi_\varepsilon(\theta)\varepsilon_t \qquad \varepsilon_t \sim N(0, \Sigma_\varepsilon)$$

- Relation between model variables and data (observation equations):

$$y_t = \Psi_0(\theta) + \Psi_1(\theta)s_t + u_t \qquad u_t \sim N(0, \Sigma_u)$$

- State transition equations & observation equations $\Rightarrow$ Kalman filter $\Rightarrow$ Likelihood function: $L(\theta|Y)$
- Prior distribution: $p(\theta)$
- Bayes' Theorem $\Rightarrow$ Posterior distribution

$$p\left(\theta|Y\right) \propto L(\theta|Y)p\left(\theta\right)$$

Needs for Nonlinear DSGE models

- However, linearized DSGE models cannot deal with features that generate pronounced nonlinearities:
    - Occasionally binding constraints
    - Stochastic volatilities
    - Markov switching coefficients
    - Asymmetric adjustment costs
- Nonlinear solution and estimation methods are required.

Session Plan

1. Sunakawa: How to solve nonlinear DSGE models
2. Hirose: How to estimate nonlinear DSGE models

- A survey paper will be published in *Japanese Economic Review*.

1. How to solve nonlinear DSGE models

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

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

- 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

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

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 $N = N_xN_y = 9$ 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 $k' = f(k,z) - \sigma_c(k,z)$.

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: Fernï£¡ndez-Villaverde, Gordon, Guerrï£¡n-Quintana, and Rubio-Ramï£¡rez, 2015; Maliar and Maliar, 2015; Gavin, Keen, Richter, and Throckmorton, 2015; Gust, Herbst, Lï£¡pez-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):

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

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

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

- The solution has a form of

$$\begin{array}{rr} c = \sigma_c(R^*_{-1}, s), & \pi = \sigma_\pi(R^*_{-1}, s), \\ R^* = \sigma_{R^*}(R^*_{-1}, s), & y = \sigma_y(R^*_{-1}, s), \end{array}$$

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

Collman operator

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

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

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
$\vdots$	$\vdots$	$\vdots$
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).

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

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

- 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

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

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

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

  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

$$
\begin{aligned}
&c_{jm}^{-\tau} = \frac{\beta}{\bar{\gamma}} R_{jm} \int \left[\frac{\sigma_c^{(i-1)}(R_{jm}, s')^{-\tau}}{z' \sigma_\pi^{(i-1)}(R_{jm}, s')} \right] p(s'|s) ds', \\
&0 = \left((1-\nu^{-1}) + \nu^{-1} c_{jm}^{\tau} - \phi \left(\pi_{jm} - \bar{\pi} \right) \left[\pi_{jm} - \frac{1}{2\nu} \left(\pi_{jm} - \bar{\pi} \right) \right] \right) c_{jm}^{-\tau} y_{jm} \\
&+ \beta \phi \int \left[\sigma_c^{(i-1)}(R_{jm}, s')^{-\tau} \sigma_y^{(i-1)}(R_{jm}, s') \left(\sigma_\pi^{(i-1)}(R_{jm}, s') - \bar{\pi} \right) \sigma_\pi^{(i-1)}(R_{jm}, s') \right] p(s'|s) \\
&R_{jm} = \left(r\bar{\pi} \left(\frac{\pi_{jm}}{\bar{\pi}} \right)^{\psi_1} \left(\frac{y_{jm}}{y^*} \right)^{\psi_2} \right)^{1-\rho_R} R_{j,-1}^{\rho_R} e^{\epsilon_{R,m}}, \\
&c_{jm} + \frac{\phi}{2} \left(\pi_{jm} - \bar{\pi} \right)^2 y_{jm} = g_m^{-1} y_{jm}. \\
&R_{jm} = \max \left\{ R_{jm}^*, 1 \right\},
\end{aligned}
$$

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

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

  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:

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

where

$$y = \left(g^{-1} - \frac{\phi}{2} \left(\sigma_\pi(R_{-1}, s) - \bar{\pi} \right)^2 \right)^{-1} \sigma_c(R_{-1}, s),$$

$$R = \left(\bar{r} \bar{\pi} \left(\frac{\sigma_\pi(R_{-1}, s)}{\bar{\pi}} \right)^{\psi_1} \left(\frac{y}{y^*} \right)^{\psi_2} \right)^{1-\rho_R} R_{-1}^{\rho_R} e^{\epsilon_R}.$$

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 ....

2. How to estimate nonlinear DSGE models

State-Space Representation

- Linear case
  1. State transition equations:
     $$s_t = \Phi_1(\theta)s_{t-1} + \Phi_\varepsilon(\theta)\varepsilon_t, \qquad \varepsilon_t \sim N(0, \Sigma_\varepsilon)$$
  2. Observation equations:
     $$y_t = \Psi_0(\theta) + \Psi_1(\theta)s_t + u_t, \qquad u_t \sim N(0, \Sigma_u)$$
- Nonlinear case
  1. State transition equations:
     $$s_t = \Phi(s_{t-1}, \varepsilon_t; \theta), \qquad \varepsilon_t \sim F_\varepsilon(\cdot; \theta)$$
  2. Observation equations:
     $$y_t = \Psi(s_t; \theta) + u_t, \qquad u_t \sim F_u(\cdot; \theta)$$

Kalman Filter

- In a linear case, the *Kalman filter* is available to evaluate likelihood.
- Distributional assumption about the initial state s_0:

$$s_0 \sim N(\bar{s}_{0|0}, P_{0|0})$$

  - It is common to set $\bar{s}_{0|0}$ and $P_{0|0}$ equal to the unconditional first and second moments of implied by the law of motion of s_t.
- Given $\bar{s}_{0|0}$ and $P_{0|0}$, the one-period-ahead forecasts of these moments are

$$\begin{aligned} \bar{s}_{1|0} &= \Phi_1 \bar{s}_{0|0}, \\ P_{1|0} &= \Phi_1 P_{0|0} \Phi_1' + \Phi_\varepsilon \Sigma_\varepsilon \Phi_\varepsilon'. \end{aligned}$$

- Given $\bar{s}_{1|0}$ and $P_{1|0}$, the conditional mean and variances of the observables y_1 are

$$\begin{aligned} \bar{y}_{1|0} &= A + B\bar{s}_{1|0}, \\ F_{1|0} &= BP_{1|0}B'. \end{aligned}$$

- Then, the forecast error of the observables y_1 is

$$\nu_{1|0} = y_1 - \bar{y}_{1|0}.$$

- It has been known that the optimal updating leads to

$$\bar{s}_{1|1} = \bar{s}_{1|0} + P_{1|0} B' F_{1|0}^{-1} \nu_{1|0},$$
$$P_{1|1} = P_{1|0} - P_{1|0} B' F_{1|0}^{-1} B P_{1|0},$$

  where $\bar{s}_{1|1}$ and $P_{1|1}$ are the mean and variances of s_1 conditional on y_1.

- In the same way as above, given $\bar{s}_{1|1}$ and $P_{1|1}$, we can obtain $\bar{s}_{2|1}$, $P_{2|1}$, $\bar{y}_{2|1}$, $F_{2|1}$, and $\nu_{2|1}$.
  - Also, updating gives $\bar{s}_{2|2}$ and $P_{2|2}$, conditional on $\{y_1, y_2\}$.
- Repeating these steps yields the sequences of $\bar{s}_{t|t-1}$, $P_{t|t-1}$, $\bar{y}_{t|t-1}$, $F_{t|t-1}$, and $\nu_{t|t-1}$ conditional on $\{y_1, y_2, ..., y_{t-1}\}$ for $t = 1, 2, ...T$.

- Since $\varepsilon_t \sim N(0, \Sigma_\varepsilon)$,

$$y_t | Y^{t-1} \sim N\left(A + B\hat{s}_{t|t-1}, F_{t|t-1}\right),$$

where $Y^{t-1} = \{y_1, y_2, ..., y_{t-1}\}$.
- Its probability density function is

$$p\left(y_t | Y^{t-1}\right) = (2\pi)^{-\frac{n}{2}} \left|F_{t|t-1}\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\nu'_{t|t-1} F^{-1}_{t|t-1} \nu_{t|t-1}\right).$$

- Therefore, the log-likelihood is given by

$$\begin{aligned} \ln L(\theta|Y) &= \sum_{t=1}^{T} \ln p\left(y_t | Y^{t-1}\right) \\ &= -\frac{nT}{2} \ln 2\pi - \frac{1}{2} \sum_{t=1}^{T} \ln \left|F_{t|t-1}\right| - \frac{1}{2} \sum_{t=1}^{T} \nu'_{t|t-1} F^{-1}_{t|t-1} \nu_{t|t-1}. \end{aligned}$$

Particle Filter

- In a nonlinear case, the *Kalman filter* is NOT available because the distribution of $y_t|Y^{t-1}$ is non-normal.
- A particle filter can approximate the likelihood function.
- While there are many particle filters, we will focus on the *bootstrap particle filter*.
    - Gordon, Salmond, and Smith (1993)

Bootstrap Particle Filter

- Idea: Particles representing s_t are propagated according to state space representation, so that the distribution of $y_t|Y^{t-1}$ can be approximated.
- Draw the initial particles $\{s_0^j\}_{j=1}^M$ from the distribution:

$$s_0^j \sim N(\bar{s}_0, P_0).$$

  - M denotes the number of particles.
  - Set $\bar{s}$ and P equal to the unconditional first and second moments of implied by the law of motion of s_t.
  - Set particle weights $W_0^j = 1$, for $j = 1, ..., M$.

For $t = 1, ..., T$:

1. Draw shocks in period t:
$$\varepsilon_t^j \sim F_\varepsilon(\cdot;\theta),$$
and propagate particles $\{s_{t-1}^j\}$ using the state-transition equation,
$$\tilde{s}_t^j = \Phi(s_{t-1}^j, \varepsilon_t; \theta).$$

2. Define the incremental weights:
$$\tilde{w}_t^j = p(y_t|\tilde{s}_t^j, \theta).$$

- The predictive density $p(y_t|Y_{1:t-1}, \theta)$ can be approximated by
$$\hat{p}(y_t|Y_{1:t-1}, \theta) = \frac{1}{M}\sum_{j=1}^{M} \tilde{w}_t^j W_{t-1}^j.$$

- If the measurement errors $u_t \sim N(0, \Sigma_u)$, then the incremental weights are evaluated as
$$\tilde{w}_t^j = (2\pi)^{-\frac{n}{2}} |\Sigma_u|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(y_t - \Psi(\tilde{s}_t^j;\theta))'\Sigma_u^{-1}(y_t - \Psi(\tilde{s}_t^j;\theta))\right\},$$
where n denotes the number of observables.

3. Define the normalized weights:

$$\tilde{W}_t^j = \frac{\tilde{w}_t^j W_{t-1}^j}{\frac{1}{M}\sum_{j=1}^{M} \tilde{w}_t^j W_{t-1}^j}$$

   - An approximation of $E[s_t|Y_{1:t}, \theta]$ is given by

$$\hat{E}[s_t|Y_{1:t}, \theta] = \frac{1}{M}\sum_{j=1}^{M} \tilde{s}_t^j \tilde{W}_t^j .$$

4. If resampling is needed, M iid draws $\{s_t^j\}_{j=1}^M$ from a multinomial distribution characterized by support points and weights $\{\tilde{s}_t^j, \tilde{W}_t^j\}$ and set $W_t^j = 1$, for $j = 1, ..., M$.
   - If resampling is not needed, let $s_t^j = \tilde{s}_t^j$ and $W_t^j = \tilde{W}_t^j$ for $j = 1, ..., M$.
5. Repeat steps 1–4 for next t.

Resampling

- A resampling step is necessary to avoid the degeneracy of the distribution of particle weights.
    - A situation in which all but a few of the weights are near zero
- Since resampling is done with replacement, a particle with a large weight is likely to be drawn many times and particles with small weights are not likely to be drawn at all.
    - Resampling effectively deals with the degeneracy problem by eliminating the particles with very small weights.
- Resampling is done whenever the effective sample size

$$\widehat{ESS}_t = \frac{M}{\frac{1}{M}\sum_{j=1}^{M}(\tilde{W}_t^j)^2}$$

falls below a threshold, e.g., $M/2$.

Likelihood Approximation

- Repeating steps 1–4 for $t = 1, ..., T$ gives particle weights $\{\tilde{w}_t^j, W_{t-1}^j\}_{j=1}^M$ for each t.
- The approximation of the log-likelihood function is given by

$$\ln \hat{p}(Y_{1:T}|\theta) = \sum_{t=1}^{T} \ln \left(\frac{1}{M} \sum_{j=1}^{M} \tilde{w}_t^j W_{t-1}^j \right).$$

Central Difference Kalman Filter

- Likelihood approximation using a particle filter causes a huge computational cost.
- If a DSGE model is approximated by a 2nd- or 3rd-order perturbation method, the Central Difference Kalman Filter (CDFK) can approximate the likelihood more efficiently.
  - Andreasen (2013)
- Idea: Approximate the filtering equations that compute and update first and second moments of state variables by 2nd-order multivariate Stirling interpolations.
- Andreasen (2013) show that a quasi maximum likelihood estimator based on the CDFK can be consistent and asymptotically normal for DSGE models solved up to third order.

Bayesian Estimation

- Once we approximate and evaluate the likelihood function $\hat{p}(Y|\theta)$, the Bayesian likelihood approach is applicable.
  - Prior distribution: $p(\theta)$
  - Bayes' Theorem $\Rightarrow$ Posterior distribution

$$\hat{p}\left(\theta|Y\right) \propto \hat{p}(Y|\theta)p\left(\theta\right)$$

- Generate draws from the posterior distribution using Markov Chain Monte Carlo (MCMC) algorithm.
  - Random-Walk Metropolis Hasting (RWMH) algorithm is widely used.

Sequential Monte Carlo Algorithm

- Issues in the RWMH algorithm:
  - The posterior distribution is possibly multimodal.
  - The RWMH algorithm can get stuck near a local mode and fail to find the entire posterior distribution.
  - It is often very difficult to find a model.
- Herbst and Schorfheide (2014, 2015) propose the Sequential Monte Carlo (SMC) algorithm.
  - Particles representing θ are propagated, similar to a particle filter.
  - Overcome the issues by building a particle approximation to the posterior gradually through tempering the likelihood function.
  - Sequence of tempered posteriors:

$$\pi_n(\theta) = \frac{[p(Y|\theta)]^{\phi_n} p(\theta)}{\int [p(Y|\theta)]^{\phi_n} p(\theta) d\theta}, \qquad n = 0, ..., N_\phi.$$

    - Tempering schedule: $\phi_n = (n/N_\phi)^\chi$

Parallelization

- Both the particle filter and the SMC algorithm can be parallelized.
  - Nonlinear solution methods can be also parallelized.
  - Massive speed gains in estimation
- Matlab parallel toolbox is very easy to use, but not so fast.
- Explicit parallelization:
  - OpenMP
  - MPI
  - GPU programming: CUDA and OpenCL
- Hardware:
  - Workstation with multi-core and multi-processors
  - Computer cluster
  - Cloud computing

Faster Programming Languages

- Aruoba and Fernández-Villaverde (2015): “A Comparison of Programming Languages in Macroeconomics,” *Journal of Economic Dynamics and Control*, 58, 265–273.
    - Solve the stochastic neoclassical growth model using C++, Fortran, Java, Julia, Python, Matlab, Mathematica, and R.
    - Report the execution time of the codes.
- Aruoba and Fernández-Villaverde (2018) update results for new versions of each language.

Aruoba and Fernández-Villaverde (2018)

Language	Compiler	Time	Rel. Time
C++	GCC	1.60	1.00
	Intel C++	1.67	1.04
	Clang	1.64	1.03
Fortran	GCC	1.61	1.01
	Intel Fortran	1.74	1.09
Java		3.20	2.00
Julia		2.35	1.47
	fast	2.14	1.34
Matlab		4.80	3.00
Python	CPython	145.27	90.79
	CPython	166.75	104.22
R		57.06	35.66
Mathematica	base	1634.94	1021.84

Aruoba and Fernández-Villaverde (2018)

Language	Compiler	Time	Rel. Time
Matlab, Mex		2.01	1.26
Rcpp		6.60	4.13
Python	Numba	2.31	1.44
Cython	Cython	2.13	1.33
Mathematica	idiomatic	4.42	2.76

References

- Andreasen, M.M. (2013). “Non-Linear DSGE Models and the Central Difference Kalman Filter,” *Journal of Applied Econometrics*, 28, 929–955.
- Aruoba, S.B. and J. Fernández-Villaverde (2015). “A Comparison of Programming Languages in Macroeconomics,” *Journal of Economic Dynamics and Control*, 58, 265–273.
- Aruoba, S.B. and J. Fernández-Villaverde (2018). “A Comparison of Programming Languages in Macroeconomics: An Update,” mimeo.
- Blanchard, O.J. and C.M. Kahn (1980). “The Slution of Linear Difference Models under Rational Expectations,” *Econometrica*, 48, 1305-1311.
- Gordon, N., D. Salmond, and A. F. Smith (1993). “Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation,” *Radar and Signal Processing, IEE Proceedings F*, 140, 107–113.

References

- Geweke, J., (1999). “Using simulation methods for Bayesian econometric models: inference, development,and communication,” *Econometric Reviews*, 18, 1–73.
- Herbst, E. and F. Schorfheide (2014). “Sequential Monte Carlo Sampling for DSGE Models,” *Journal of Applied Econometrics*, 29, 1073–98.
- Herbst, E. and F. Schorfheide (2015). *Bayesian Estimation of DSGE Models*, Princeton University Press, Princeton.
- Sims, C. A. (2002) “Solving Linear Rational Expectations Models,” *Computational Economics*, 20, 1–20.