Numerically Stable and Accurate Stochastic Simulation Approaches for Solving Dynamic Economic Models

Kenneth L. Judd, Lilia Maliar and Serguei Maliar

~~ICE-2012~~

July 2013

Three broad classes of numerical methods

1. Projection methods, Judd (1992), Christiano and Fisher (2000), etc.
    - solution domain = prespecified grid of points;
    - accurate and fast with few state variables but cost grows exponentially with the number of state variables (curse of dimensionality!).
2. Perturbation methods, Judd and Guu (1993), Gaspar and Judd (1997), Juillard (2003), etc.
    - solution domain = one point (steady state);
    - practical in large-scale models but the accuracy can deteriorate dramatically away from the steady state.
3. Stochastic simulation methods, Marcet (1988), Smith (2001), etc.
    - solution domain = simulated series;
    - simple to program but often numerically unstable, and the accuracy is lower than that of the projection methods.

Our aim is to improve the performance of stochastic simulation methods.

Stochastic simulation methods and their shortcomings

- A stochastic simulation method solves a model as follows:

  Step 1. Guess policy functions / value function.

  Step 2. Simulate time series solution.

  Step 3. Use simulation results to recompute the guess.

  Iterate on *Steps* $2-3$ until convergence.

- Step 3 requires
  - to fit an approximating function to the simulated data (regression);
  - to evaluate conditional expectations (integration).
- We show that both regression and integration have 2 problems:
  - In regression, polynomial terms are highly correlated (multicollinearity), and the standard LS technique fails ⇒ **numerical instability.**
  - Monte Carlo integration is very inaccurate ⇒ **the overall accuracy of solutions is low.**

With GSSA, we correct the above two shortcomings

- We stabilize the stochastic simulation procedure:
  - *we build the regression step on approximation methods designed for dealing with multicollinearity*
- We attain high accuracy of solutions:
  - *we generalize the stochastic simulation algorithm to include accurate Gauss Hermite quadrature and monomial integration methods*
- The generalized stochastic simulation algorithm (GSSA) is
  - *numerically stable*
  - *comparable in accuracy to most accurate methods in the literature*
  - *tractable in problems with high dimensionality (hundreds of state variables)*
  - *very simple to program*

We present the results by way of an example

One-agent stochastic growth model:

$$\max_{\{k_{t+1},c_t\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$$

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

$$\ln a_{t+1} = \rho \ln a_t + \epsilon_{t+1}, \quad \epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2)$$

where initial condition (k_0, a_0) is given;
$f(\cdot)$ = production function;
c_t = consumption; k_{t+1} = capital; a_t = productivity;
β = discount factor; δ = depreciation rate of capital;
ρ = autocorrelation coefficient of the productivity level;
σ = standard deviation of the productivity shock.

Definition of the solution

We look for the policy function $k_{t+1} = K(k_t, a_t)$ that satisfies:

- Euler equation:

$$u'(c_t) = \beta E_t \left\{ u'(c_{t+1}) \left[(1-\delta) \cancel{k_t} + a_t f'(k_t) \right] \right\}$$

- Transition equations:

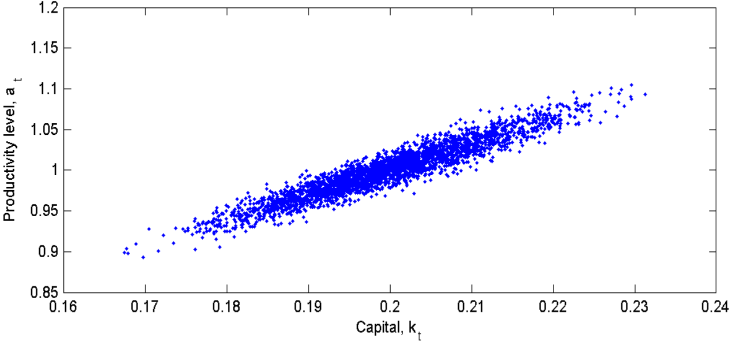
$$c_t + k_{t+1} = (1-\delta) k_t + a_t f(k_t),$$

$$\ln a_{t+1} = \rho \ln a_t + \epsilon_{t+1}.$$

Key advantage of stochastic simulation methods

- Stochastic simulation method compute a solution on **the "right"** domain - only in the areas of the state space that are visited in simulation (high-probability area or essential ergodic set).

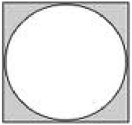
Figure 1. The ergodic set in the model with a closed-form solution.



- Projection methods use a rectangular domain which is **too large.**
- Perturbation methods use one-point domain which is **too small.**

Reduction in cost in a 2-dimensional case

- How much can we save on cost using the simulation domain comparatively to the hypercube domain?
- Suppose the (essential) ergodic set is a circle.
- In the 2-dimensional case, a circle inscribed within a square occupies about 79% of the area of the square.

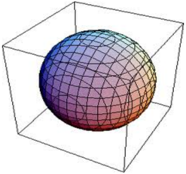


- The reduction in cost is proportional to the shaded area in the figure.

It does not seem to be a large gain but ...

Reduction in cost in a p-dimensional case

- In a 3-dimensional case, the gain is larger $\frac{\mathcal{V}^3_{sphere}}{\mathcal{V}^3_{cube}} \approx 0.52$ (a volume of a sphere of diameter 1 is 52% of the volume of a cube of width 1)



- $\frac{\mathcal{V}^p_{sphere}}{\mathcal{V}^p_{cube}}$ declines very rapidly with p, and the gains become enormous

$$\frac{\mathcal{V}^p_{sphere}}{\mathcal{V}^p_{cube}} = \begin{cases} \frac{(\pi/2)^{\frac{p-1}{2}}}{1\cdot 3\cdot \ldots \cdot p} & \text{for } p = 1, 3, 5\ldots \\ \frac{(\pi/2)^{\frac{p}{2}}}{2\cdot 4\cdot \ldots \cdot p} & \text{for } p = 2, 4, 6\ldots \end{cases}.$$

When $p = 10 \Rightarrow \frac{\mathcal{V}^{10}_{sphere}}{\mathcal{V}^{10}_{cube}} = 3 \cdot 10^{-3}$.

When $p = 30 \Rightarrow \frac{\mathcal{V}^{30}_{sphere}}{\mathcal{V}^{30}_{cube}} = 2 \cdot 10^{-14}$ – *a tiny fraction of the hypercube!*

Poor performance of stochastic simulation methods

- Stochastic simulation methods seem to be very promising, especially for problems with high dimensionality where other methods are intractable.
- But their performance in applications was truly disappointing.

We next explain why...

Starting point: simulation-based PEA of Marcet (1988)

Parameterize the marginal utility function,

$$u'(c_t) = E_t\left\{\beta u'(c_{t+1})\left[1-\delta+a_{t+1}f'(k_{t+1})\right]\right\} \approx \Psi(k_t, a_t; b),$$

where $\Psi(k_t, a_t; b) = \exp\left(b_0 + b_1 \ln k_t + b_2 \ln a_t + \ldots + b_n (\ln a_t)^L\right)$ is an exponentiated polynomial. Write the constraint as

$$k_{t+1} = (1-\delta)k_t + a_t f(k_t) - u'^{-1}\left[\Psi(k_t, a_t; b)\right].$$

- Fix $b = (b_0, \ldots, b_n)$. Given $\{a_t\}_{t=0}^{T}$, simulate $\{c_t, k_{t+1}\}_{t=0}^{T}$ and construct
$$y_t \equiv \beta u'(c_{t+1})\left[1-\delta+a_{t+1}f'(k_{t+1})\right],$$
- Run a non-linear LS (NLLS) regression $y_t = \Psi(k_t, a_t; b) + \varepsilon \Rightarrow$ get $\widehat{b}$.
- Compute the next-iteration input $b^{(j+1)}$ using fixed-point iteration
$$b^{(j+1)} = (1-\xi)\, b^{(j)} + \xi \widehat{b},$$
where $\xi \in (0, 1]$ = damping parameter.

Problems with simulation-based PEA method

- **Problem 1 (numerical instability).** Works well for 1st-degree polynomials but is numerically unstable under higher (even 2-nd) degree polynomials. For example, Den Haan and Marcet (1990) removed a cross term $\ln k_t \ln a_t$ in the 2-nd degree polynomial,

$$\exp\left(b_0 + b_1 \ln k_t + b_2 \ln a_t + b_3 \ln k_t^2 + b_4 \ln a_t^2 + \underbrace{b_5 \ln k_t \ln a_t}_{was\ \ removed}\right).$$

- **Problem 2 (low accuracy).** High degree polynomials do not produce more accurate solutions than 1-st degree polynomial (in our model, polynomials of degrees 1-5 lead to similar Euler equation errors).
- **These both problems must be solved at once (or none).**
  - *Restoring numerical stability is of no use if high-degree polynomials do not lead to more accurate solutions.*
  - *Making high-degree polynomials to be highly accurate is of no use if they are numerically unstable and cannot be computed.*

What causes instability? Ill-conditioned LS problem

- Under *the linear regression model*, $y = Xb + \varepsilon$, we have the OLS estimator
$$\widehat{b} = \left(X^\top X\right)^{-1} X^\top y,$$
where $X \equiv [1_T, x_1, ..., x_n] \in \mathbb{R}^{T\times(n+1)}$.
- The degree of ill-conditioning of $X^\top X$ is measured by the condition number
$$\mathcal{K}\left(X^\top X\right) \equiv \lambda_1 / \lambda_n$$
$\lambda_1 =$ the largest eigenvalue of $X^\top X$; $\lambda_n =$ its smallest eigenvalue.
- **Ill-conditioning:** $\mathcal{K}\left(X^\top X\right)$ is large $\Longrightarrow$ $X^\top X$ is close to being singular (not invertible).

Addressing Problem 1: Attaining numerical stability

1. We replace the exponentiated polynomial $\Psi(k, a; b) = \exp\left(b_0 + b_1 \ln k_t + b_2 \ln a_t + ... + b_n (\ln a_t)^L\right)$ used in Marcet (1988) with a simple polynomial $\Psi(k, a; b) = b_0 + b_1 \ln k_t + b_2 \ln a_t + ... + b_n (\ln a_t)^L$. This allows us to replace NLLS methods with linear methods.
2. We use approximation methods that can handle collinear data and dampen movements in b.
    - LS using SVD, Tikhonov regularization;
    - Least absolute deviations (LAD) methods (primal and dual linear programming problems);
    - Principal components (truncated SVD) method.
3. Other factors that can affect numerical stability of GSSA:
    - Data normalization.
    - The choice of a family of basis functions.
    - The choice of policy functions to parameterize.

Normalizing the variables

- *Center* - subtract the sample mean from each observation.
- *Scale* - divide each observation by the sample standard deviation.
- By construction, a centered variable has a zero mean, and a scaled variable has a unit standard deviation.
- After a regression model is estimated, the coefficients in the original (unnormalized) regression model are restored.

LS approaches to the linear regression model

Two LS approaches that are more numerically stable and more suitable for dealing with ill-conditioning than the standard OLS approach.

1. *LS using SVD (LS-SVD)*: uses a singular-value decomposition of X.
2. *Regularized LS using Tikhonov regularization (RLS-Tikhonov)*: relies on a specific (Tikhonov) regularization of the ill-conditioned LS problem that imposes penalties based on the size of the regression coefficients.

The LS-SVD approach finds a solution to the original ill-conditioned LS problem, while the RLS-Tikhonov approach modifies (regularizes) the original ill-conditioned LS problem into a less ill-conditioned problem.

LS-SVD

- SVD of the matrix $X \in \mathbb{R}^{T \times n}$

  $$X = USV^\top$$

  where $U \in \mathbb{R}^{T \times n}$ and $V \in \mathbb{R}^{n \times n}$ $=$ orthogonal matrices; $S \in \mathbb{R}^{n \times n}$ $=$ diagonal matrix with diagonal entries $s_1 \geq s_2 \geq ... \geq s_n \geq 0$, known as *singular values* of X.
- The OLS estimator $\widehat{b} = \left(X^\top X\right)^{-1} X^\top y$ in terms of the SVD:

  $$\widehat{b} = \left(VS^\top SV^\top\right)^{-1} VS^\top U^\top y = VS^{-1} U^\top y$$

- If $X^\top X$ is well-conditioned $\Longrightarrow$ the OLS formula and the LS-SVD formula give identical estimates of b.
- However, if $X^\top X$ is ill-conditioned and the standard OLS estimator cannot be computed $\Longrightarrow$ it is still possible that matrices X and S are sufficiently well-conditioned, $\mathcal{K}(S) = \sqrt{\mathcal{K}(X^\top X)}$ $\Longrightarrow$ can compute the LS-SVD estimator.

RLS-Tikhonov

- Regularization - process of re-formulating an ill-conditioned problem by imposing additional restrictions on the solution.
- Tikhonov regularization - the most commonly used regularization method in approximation theory.
- Impose an L_2 penalty on the size of the regression coefficients:

  $$\min_b \|y - Xb\|_2^2 + \eta \|b\|_2^2 = \min_b (y - Xb)^\top (y - Xb) + \eta b^\top b$$

  where $\eta \geq 0 =$ regularization parameter.
- Find the FOC with respect to b

  $$\widehat{b}(\eta) = \left(X^\top X + \eta I_n\right)^{-1} X^\top y$$

  where $I_n =$ an identity matrix of order n.
- *Note:* add a positive constant to $X^\top X$ prior to inverting this matrix.
  $\Longrightarrow$ Even if $X^\top X$ is singular, the matrix $X^\top X + \eta I_n$ is non-singular.
  $\Longrightarrow$ Can compute its inverse.

LAD approaches to the linear regression model

- Replace the ill-conditioned LS problem with a least-absolute deviations (LAD) problem

$$\min_{b} \left\| y - Xb \right\|_1 = \min_{b} 1_T^\top \left| y - Xb \right|$$

  where $\left\| \cdot \right\|_1$ denotes L_1 vector norm.
- The LAD problem does not require computing $\left(X^\top X \right)^{-1}$.
- No explicit solution. However, we can re-formulate the LAD problem to consist of a linear objective function and linear constraints $\Longrightarrow$ Solve with standard linear programming techniques.
- Substitute $\left| y - X\beta \right|$ with a vector $w \in \mathbb{R}^T$ to obtain

$$\min_{b,\ w} 1_T^\top w$$

$$\text{s.t.} \quad -w \leq y - X\beta \leq w$$

- This problem has $n + T$ unknowns. We argue that it is not the most suitable for a numerical analysis.

LAD: primal problem (LAD-PP)

- Charnes et al. (1955): express the deviation for each observation as a difference between two non-negative variables υ_t^+ and υ_t^-,

$$y_t - \sum_{i=0}^{n} b_i x_{ti} = \upsilon_t^+ - \upsilon_t^-, \tag{1}$$

- υ_t^+ and υ_t^- can be interpreted as non-negative vertical deviations above and below the fitted line, $\widehat{y}_t = X_t \widehat{b}$, respectively; $\upsilon_t^+ + \upsilon_t^- =$ absolute deviation between the fit $\widehat{y}_t$ and the observation y_t.
- *Primal problem:* minimize the total sum of absolute deviations subject to (1),

$$\begin{gathered} \min_{\upsilon^+,\upsilon^-,b} \; 1_T^\top \upsilon^+ + 1_T^\top \upsilon^- \\ \text{s.t. } \upsilon^+ - \upsilon^- + Xb = y, \\ \upsilon^+ \geq 0, \quad \upsilon^- \geq 0, \end{gathered}$$

  where $\upsilon_t^+, \upsilon_t^- \in \mathbb{R}^T$.
- This formulation is more simple to solve than the direct formulation.

LAD: dual problem (LAD-DP)

- Every primal problem can be converted into a dual problem.
- *Dual problem* corresponding to the primal problem:

$$\max_q y^\top q$$

$$\text{s.t. } X^\top q = 0$$

$$-1_T \leq q \leq 1_T$$

where $q \in \mathbb{R}^T$ is a vector of unknowns.

- If the number of observations, T, is sizable (i.e. $T \gg n$), the dual problem is less computationally cumbersome than the primal problem.

Regularized LAD (RLAD)

- Modify the original LAD problem to incorporate an L_1 penalty on b.
- *The RLAD problem:*

$$\min_b \|y - Xb\|_1 + \eta \|b\|_1 = \min_b 1_T^\top |y - Xb| + \eta 1_n^\top |b|,$$

where $\eta \geq 0 =$ regularization parameter.

- We develop a linear programming formulation of the RLAD problem parallel to the LAD-PP: replace $|b_i|$ with two variables.
- Wang, Gordon and Zhu (2006): represent $|b_i|$ as $sign(b_i)\, b_i$.

RLAD: primal problem (RLAD-PP)

- To cast the RLAD problem into a linear programming form, we represent b as $b_i = \varphi_i^+ - \varphi_i^-$, with $\varphi_i^+ \geq 0$, $\varphi_i^- \geq 0$ for $i = 1, ..., n$.
- We then impose a linear penalty on each φ_i^+ and φ_i^-.
- The resulting regularized version of the primal problem:

$$\min_{\upsilon^+,\upsilon^-,\varphi^+,\varphi^-} \quad 1_T^\top \upsilon^+ + 1_T^\top \upsilon^- + \eta 1_n^\top \varphi^+ + \eta 1_n^\top \varphi^-$$

$$\text{s.t.} \quad \upsilon^+ - \upsilon^- + X\varphi^+ - X\varphi^- = y,$$

$$\upsilon^+ \geq 0, \quad \upsilon^- \geq 0,$$

$$\varphi^+ \geq 0, \quad \varphi^- \geq 0,$$

  where φ^+, $\varphi^- \in \mathbb{R}^n$ are vectors that define $b(\eta)$.

- This problem has $2T + 2n$ unknowns, as well as T equality restrictions and $2T + 2n$ lower bounds.

RLAD: dual problem (RLAD-DP)

- The dual problem corresponding to the RLAD-PP:

$$\max_{q} y^{\top} q$$
$$\text{s.t. } X^{\top} q \leqslant \eta \cdot 1_n,$$
$$-X^{\top} q \leqslant \eta \cdot 1_n,$$
$$-1_T \leq q \leq 1_T,$$

  where $q \in \mathbb{R}^T$ = vector of unknowns.

- Here, $2n$ linear inequality restrictions and $2T$ lower and upper bounds on T unknown components of q.

Principal component method (Truncated SVD, LS-TSVD)

- $Z \equiv XV$, where $X \in \mathbb{R}^{T \times n}$, $Z \in \mathbb{R}^{T \times n}$ and $V \in \mathbb{R}^{n \times n}$.
- $z_1, ..., z_n$ are called *principal components* of X and are orthogonal, $z_i^\top z_i = s_i^2$ and $z_j^\top z_i = 0$ for any $j \neq i$, where $s_i = i$th singular value of X.
- *Idea:* reduce ill-conditioning of X to a "desired" level by excluding low-variance principal components corresponding to small singular values.
- Let $\kappa =$ largest condition number of X that we are willing to accept.
- Compute $\frac{s_1}{s_2}, ..., \frac{s_1}{s_n}$, where $s_1 =$ largest singular value.
- $\mathcal{K}(X) = \mathcal{K}(S) = \frac{s_1}{s_n} =$ actual condition number of the matrix X.

Principal component method (Truncated SVD, LS-TSVD)

- Let $Z^r \equiv (z_1, ..., z_r) \in \mathbb{R}^{T \times r}$ be the first r principal components for which $\frac{s_1}{s_i} \leq \kappa$.
- Remove the last $n - r$ principal components for which $\frac{s_1}{s_i} > \kappa$.
- By construction, $\mathcal{K}(Z^r) \leq \kappa$.
- Re-write the linear regression model in terms of Z^r,

$$y = Z^r \vartheta^r + \varepsilon,$$

  where $\vartheta^r \in \mathbb{R}^r$ = vector of coefficients.
- Estimate ϑ^r using any of the LS and LAD methods described.
- Find $\widehat{b} = V^r \widehat{\vartheta}^r \in \mathbb{R}^n$, where $V^r = (v_1, ..., v_r) \in \mathbb{R}^{n \times r}$ contains the first r right singular vectors of X.

Choosing policy functions to parameterize

- *Marcet (1988):* parameterize marginal-utility policy function

$$u'(c_t) = E_t\left\{\beta u'(c_{t+1})\left[1-\delta + a_{t+1}f'(k_{t+1})\right]\right\} \approx \Psi(k_t, a_t; b)$$

- *Our benchmark case:* parameterize capital policy function $k_{t+1} = K(k_t, a_t)$,

$$k_{t+1} = E_t\left\{\beta \frac{u'(c_{t+1})}{u'(c_t)}\left[1-\delta + a_{t+1}f'(k_{t+1})\right]k_{t+1}\right\} \approx \Psi(k_t, a_t; b)$$

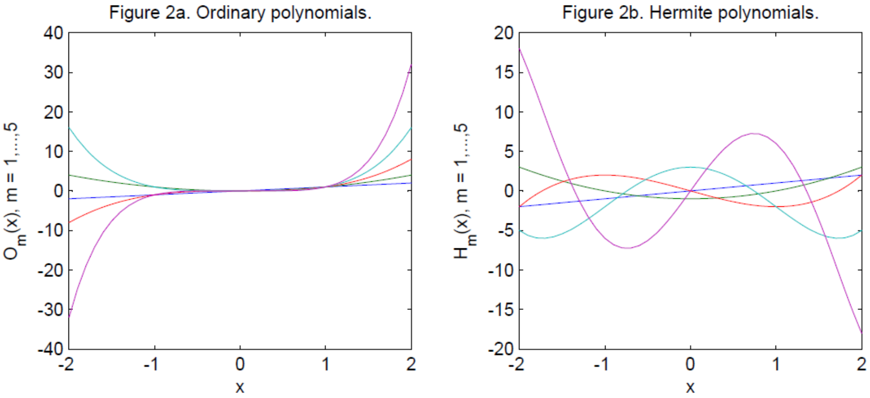
Choosing a family of basis functions

- Polynomial families of basis functions.
- Ordinary polynomial family - standard.
- A better alternative is orthogonal polynomial families.
- Ordinary polynomials $O_m(x)$ versus Hermite polynomials $H_m(x)$ up to degree 5:

$$\begin{array}{ll}
O_0(x) = 1 & H_0(x) = 1 \\
O_1(x) = x & H_1(x) = x \\
O_2(x) = x^2 & H_2(x) = x^2 - 1 \\
O_3(x) = x^3 & H_3(x) = x^3 - 3x \\
O_4(x) = x^4 & H_4(x) = x^4 - 6x^2 + 3 \\
O_5(x) = x^5 & H_5(x) = x^5 - 10x^3 + 15x.
\end{array}$$

- $O_m(x)$, $m = 1, ..., 5$ appear very similar $\Longrightarrow$ the explanatory variables for the regression are likely to be correlated.
- $H_m(x)$, $m = 1, ..., 5$ are different in the shapes $\Longrightarrow$ the multicollinearity problem manifests to a much lesser degree, if at all.

Choosing a family of basis functions



Methodology and parameterization

- Production function: $f(k_t) = k_t^{\alpha}$ with $\alpha = 0.36$.
- Utility function: $u(c_t) = \frac{c_t^{1-\gamma}-1}{1-\gamma}$ with $\gamma \in \{0.1, 1, 10\}$.
- Process for shocks: $\rho = 0.95$ and $\sigma = 0.01$.
- Discount factor: $\beta = 0.99$.
- Depreciation rate: $\delta = 1$ and $\delta = 0.02$.
- Under $\gamma = 1$ and $\delta = 1 \Longrightarrow$ *closed-form solution*.
- Accuracy is measured by an Euler-equation error,

$$\mathcal{E}(k_t, a_t) \equiv E_t \left[\beta \frac{c_{t+1}^{-\gamma}}{c_t^{-\gamma}} \left(1 - \delta + \alpha a_{t+1} k_{t+1}^{\alpha-1}\right) \right] - 1,$$

expressed in log10 units.

Results for the model with the closed-form solution

Full depreciation of capital, $\delta = 1$.

	$\mathcal{E}_{mean}$	CPU	$\mathcal{E}_{mean}$	CPU	$\mathcal{E}_{mean}$	CPU
Polyn. degree	*OLS, Ordinary Unnormalized*		*OLS, Ordinary Normalized*		*OLS, Hermite Unnormalized*	
1st	-3.52	0.8 sec	-3.52	1 sec	-3.52	1 sec
2nd	-5.46	3.1 sec	-5.46	3 sec	-5.46	4 sec
3rd	-	-	-6.84	5 sec	-6.84	6 sec
4th	-	-	-	-	-7.94	8 sec
5th	-	-	-	-	-9.09	10 sec
	Ordinary, LS-SVD Normalized		*Ordinary, LAD-PP Normalized*		*Ordinary, RLS-Tikh.* $\eta = 10^{-7}$	
1st	-3.52	1 sec	-3.52	16 sec	-3.52	1 sec
2nd	-5.46	3 sec	-5.55	1.5 min	-5.46	3 sec
3rd	-6.84	5 sec	-6.97	4.1 min	-5.85	4 sec
4th	-7.94	6 sec	-8.16	6.4 min	-6.12	7 sec
5th	-9.12	10 sec	-9.10	9.3 min	-6.22	11 sec

Results for the model without a closed-form solution

Partial depreciation of capital, $\delta = 0.02$.

	$\mathcal{E}_{mean}$	CPU
Polyn. degree	*MC(1)* $T = 10,000$	
1st	-4.26	1 sec
2nd	-4.42	11 sec
3rd	-4.32	25 sec
4th	-4.31	47 sec
5th	-4.23	80 sec

- We attain stability but now high-degree polynomials do not lead to more accurate solution. Why?
- Recall that low accuracy of Monte Carlo integration restricts the overall accuracy of solutions.

GSSA: deterministic integration methods

Our GSSA relies on accurate Gauss Hermite quadrature integration

$$\int_{\mathbb{R}^N} g(\varepsilon) w(\varepsilon) d\varepsilon \approx \sum_{j=1}^{J} \omega_j g(\varepsilon_j),$$

where $\{\varepsilon_j\}_{j=1}^{J}$ = integration nodes, $\{\omega_j\}_{j=1}^{J}$ = integration weights.

Example

a) A two-node Gauss-Hermite quadrature method, $Q(2)$, uses nodes $\epsilon_1 = -\sigma$, $\epsilon_2 = \sigma$ and weights $\omega_1 = \omega_2 = \frac{1}{2}$.
b) A three-node Gauss-Hermite quadrature method, $Q(3)$, uses nodes $\epsilon_1 = 0$, $\epsilon_2 = \sigma\sqrt{\frac{3}{2}}$, $\epsilon_3 = -\sigma\sqrt{\frac{3}{2}}$ and weights $\omega_1 = \frac{2\sqrt{\pi}}{3}$, $\omega_2 = \omega_3 = \frac{\sqrt{\pi}}{6}$.
c) A one-node Gauss-Hermite quadrature method, $Q(1)$, uses a zero node, $\epsilon_1 = 0$, and a unit weight, $\omega_1 = 1$.

Quadrature integration in the studied model

For $t = 0, ..., T-1$, we approximation the conditional expectation as

$$y_t = \sum_{j=1}^{J} \left\{ \omega_j \cdot \left(\beta u' \left(c_{t+1,j} \right) \left[1 - \delta + a_{t+1,j} \; f' \left(k_{t+1} \right) \right] \right) \right\},$$

where $c_{t+1,j}$, the value of c_{t+1} if the innovation in productivity is ϵ_j, is defined for $j = 1, ..., J$ by

$$\begin{aligned} a_{t+1,j} &\equiv a_t^{\rho} \exp \left(\epsilon_j \right), \\ c_{t+1,j} &\equiv \Psi \left(k_{t+1}, a_t^{\rho} \exp \left(\epsilon_j \right); b^{(p)} \right). \end{aligned}$$

where $\left\{ \epsilon_j \right\}_{j=1,...,J}$ and $\left\{ \omega_j \right\}_{j=1,...,J}$ are J integration nodes and weights, respectively.

Results for the model with partial depreciation of capital

	$\mathcal{E}_{mean}$	CPU	$\mathcal{E}_{mean}$	CPU	$\mathcal{E}_{mean}$	CPU
Polyn. degree	*MC(1)* $T = 10,000$		*MC(2000)* $T = 10,000$		*MC(1)* $T = 100,000$	
1st	-4.26	1 sec	-4.40	20.6 min	-4.39	4 sec
2nd	-4.42	11 sec	-6.04	28.5 min	-4.87	1.3 min
3rd	-4.32	25 sec	-6.15	36.6 min	-4.86	3.1 min
4th	-4.31	47 sec	-6.08	55.6 min	-4.72	5.7 min
5th	-4.23	80 sec	-6.07	1.27 h	-4.71	10.4 min
	Q(1) $T = 100$		*Q(2)* $T = 10,000$		*Q(10)* $T = 10,000$	
1st	-4.36	3 sec	-4.36	16 sec	-4.36	20 sec
2nd	-6.05	4 sec	-6.13	27 sec	-6.13	34 sec
3rd	-6.32	5 sec	-7.48	35 sec	-7.48	44 sec
4th	-6.24	6 sec	-8.72	44 sec	-8.72	54 sec
5th	-6.04	7 sec	-8.91	51 sec	-8.91	63 sec

RLS-TSVD with $\kappa = 10^7$

The multi-country model

The planner maximizes a weighted sum of N countries' lifetime utilities

$$\max_{\left\{\left\{c_t^h, k_{t+1}^h\right\}_{h=1}^N\right\}_{t=0}^\infty} E_0 \sum_{h=1}^N \lambda^h \left(\sum_{t=0}^\infty \beta^t u^h\left(c_t^h\right)\right)$$

subject to

$$\sum_{h=1}^N c_t^h + \sum_{h=1}^N k_{t+1}^h = \sum_{h=1}^N k_t^h (1-\delta) + \sum_{h=1}^N a_t^h f^h\left(k_t^h\right),$$

where λ^h is country h's welfare weight.
Productivity of country h follows the process

$$\ln a_{t+1}^h = \rho \ln a_t^h + \epsilon_{t+1}^h,$$

where $\epsilon_{t+1}^h \equiv \varsigma_{t+1} + \varsigma_{t+1}^h$ with $\varsigma_{t+1} \sim \mathcal{N}\left(0, \sigma^2\right)$ is identical for all countries and $\varsigma_{t+1}^h \sim \mathcal{N}\left(0, \sigma^2\right)$ is country-specific.

Results for the multi-country model

Numb. of countr.	Polyn. degree	Numb. of coeff.	$\mathcal{E}_{mean}$	CPU	$\mathcal{E}_{mean}$	CPU
			RLS-Tikh.,$\eta = 10^{-5}$		*RLS-TSVD*, $\kappa = 10^{7}$	
			MC(1), $T = 10,000$		M2, $T = 1000$	
N=2	1st	5	-4.70	4.2 min	-4.65	37 sec
	2nd	15	-4.82	19.3 min	-6.01	6.8 min
	3rd	35	-4.59	57 min	-7.09	10.4 min
	4th	70	-4.57	2.6 hours	-7.99	16.3 min
	5th	126	-4.53	6.8 hours	-8.00	34.8 min
			RLS-Tikh.,$\eta = 10^{-5}$		*RLS-Tikh.*, $\eta = 10^{-5}$	
			MC(1), $T = 10,000$		*Q(1)*, $T = 1000$	
N=20	1st	41	-4.55	6.5 min	-4.75	56 sec
	2nd	861	-3.88	2.1 hours	-5.40	18 min
N=200	1st	401	-3.97	37.2 min	-4.59	16.8 min

When N=200, for *RLS-Tikh.,Q(1)*, we use $T = 2000$

Conclusion

- Stochastic simulation methods operate on relevant domain and have potential advantages both in terms of accuracy and cost compared to methods operating on prespecified domains.
- The performance of the existing stochastic simulation algorithms was handicapped by two problems:
  - numerical instability (because of multicollinearity);
  - large integration errors (because of low accuracy of Monte Carlo integration).
- In GSSA, we fixed both of these problems:
  - approximation methods that can handle ill-conditioned problems;
  - a generalized notion of integration that relies on accurate deterministic methods.
- GSSA demonstrated a great performance in the studied examples:
  - Numerically stable;
  - Very accurate;
  - Very simple to program;
  - Tractable for problems with high dimensionality.