A Cluster-Grid Projection Method for Solving Problems with High Dimensionality

Kenneth Judd, Lilia Maliar and Serguei Maliar

April 17, 2011

Presentation builds on the following articles:

Main

1. Kenneth L. Judd, Lilia Maliar and Serguei Maliar *"A Cluster-Grid Projection Method: Solving Problems with High Dimensionality"*, NBER working paper 15965, 2010.

Others

2. Serguei Maliar, Lilia Maliar and Kenneth L. Judd, *"Solving the Multi-Country Real Business Cycle Model Using Ergodic Set Methods"*, Journal of Economic Dynamic and Control, 2011, 35(2), 207-228.
3. Robert Kollmann, Serguei Maliar, Benjamin Malin and Paul Pichler, *"Comparison of Solutions to the Multi-Country Real Business Cycle Model"*, Journal of Economic Dynamics and Control, 2011, 35(2), 186-202.
4. Kenneth L. Judd, Lilia Maliar and Serguei Maliar *"Numerically Stable Stochastic Simulation Methods for Solving Dynamic Models"*, NBER working paper 15296, 2009.

Cluster-grid algorithm (CGA)

- **A novel accurate method for solving dynamic economic models:** works for problems with high dimensionality, intractable for earlier solution methods - 400 state variables using a laptop.
- **Related literature focuses on much lower dimensionality:** a special JEDC 2011's issue compares solution methods (including our CGA) using models with 20 state variables at most.
- **Examples of potential CGA applications:**
  - macroeconomics (many heterogeneous agents);
  - international economics (many countries);
  - industrial organization (many firms);
  - finance (many assets);
  - climate change (many sectors and countries); etc.
- **CGA is a global method:** can handle strong non-linearities and inequality constraints.

Ingredients of CGA

- **Endogenous solution domain:** our grid is constructed by clustering methods to surround the ergodic set - we avoid costs of finding a solution in the areas of state space that are never visited in equilibrium.
- **Low-cost integration:** non-product monomial and one-point quadrature integration rules.
- **Efficient solver for finding the polynomial coefficients:** fixed-point iteration.
- **Vectorized approaches for finding the control variables:** precomputation and iteration-on-allocation by Maliar, Maliar and Judd (2011).

- **Taken together, these ingredients allow us to meet challenges of high-dimensional problems.**

Background

Three broad classes of numerical methods

1. **Projection methods**; Judd (1992), Christiano and Fisher (2000).
2. **Perturbation methods;** Judd and Guu (1993), Collard and Juillard (2001).
3. **Stochastic-simulation methods;** den Haan and Marcet (1990), Smith (1991).

Illustrative example

A one-sector neoclassical growth model:

$$\max_{\{k_{t+1},c_t\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \delta^t \ln\left(c_t\right)$$

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

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

where $\epsilon_{t+1} \sim \mathcal{N}\left(0, \sigma^2\right)$; and initial condition $\left(k_0, a_0\right)$ is given;
$u\left(\cdot\right)$ = utility function; $f\left(\cdot\right)$ = production function;
c_t = consumption; k_{t+1} = capital; a_t = productivity;
δ = discount factor; d = depreciation rate of capital;
ρ = autocorrelation coefficient of the productivity level;
σ = standard deviation of the productivity shock.

Projection methods

Characteristic features

- Solve a model on a prespecified grid of points.
- Use numerical (quadrature) integration for approximating conditional expectations.
- Compute polynomial coefficients of policy functions using Newton's type solver.

A projection method for the growth model

- Choose a grid of I points in the state space $\{k_i, a_i\}_{i=1}^{I}$.
- Parameterize the capital policy function by a polynomial

$$k_i' \simeq \Psi(k_i, a_i; \beta) = \beta_0 + \beta_1 k_i + \beta_2 a_i + \beta_3 k_i^2 + \beta_4 a_i^2 + ...$$

and substitute it in the Euler equation to get

$$\min_{\beta} \left\| u_1(c_i(k_i, a_i; \beta)) - E\left\{\delta u_1\left(c_i'(k_i, a_i; \beta)\right)\left[1 - d + a_i' f'\left(\Psi(k_i, a_i; \beta)\right)\right]\right\}\right\|$$

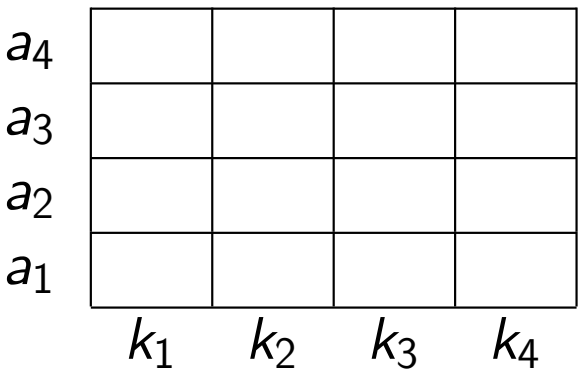
where $\beta \equiv (\beta_0, \beta_1, ...)$ is a vector of coefficients and

$$\begin{aligned} a_i' &= a_i^{\rho} \exp(\epsilon) \\ k_i'' &= \Psi\left(\Psi(k_i, a_i; \beta), a_i'; \beta\right) \\ c_i(k_i, a_i; \beta) &= (1-d) k_i + a_i f(k_i) - k_i' \\ c_i'(k_i, a_i; \beta) &= (1-d) k_i' + a_i' f(k_i') - k_i'' \end{aligned}$$

- Find a vector of coefficients β that minimizes the distance using a Newton's type of solver (this procedure involves evaluating conditional expectations, i.e., numerical integration).

Projection methods: curse of dimensionality

- Very accurate and fast with few state variables but cost grows exponentially with dimensionality!
  (a) Product hypercube domain $\Longrightarrow$ Curse of dimensionality!
  (b) Product quadrature integration $\Longrightarrow$ Curse of dimensionality!
  (c) Newton's solver (Jacobian, Hessian) $\Longrightarrow$ Curse of dimensionality!



- 2 state variables with 4 grid points $\Rightarrow 4 \times 4 = 4^2 = 16$
- 3 state variables with 4 grid points $\Rightarrow 4^3 = 64$
...
- 10 state variables with 4 grid points $\Rightarrow 4^{10} = 1,048,576$
(With 100 grid points $\Rightarrow 100^{10} = 10^{20}$).

- *Kruger and Kubler (2004):* Smolyak's sparse grid - efficient grid with relatively small number of points in a multidimensional hypercube.

Perturbation methods

Characteristic features

- Compute a solution in just one point (steady state).
- Identify polynomial coefficients of policy functions using k-order Taylor's expansion of the optimality conditions.

Perturbation methods: a log-linearization example

- *Log-linearization* - first-order Taylor's expansion, e.g.,

$$u'(c_t) \simeq u'(c) + u''(c)\, c \frac{(c_t - c)}{c} = u'(c) + u''(c)\, c \widehat{c}_t$$

  where $\widehat{c}_t = \frac{c_t - c}{c}$ = log-deviation of c_t from the steady state c.
- Substitute $\widehat{c}_t$ and $\widehat{k}_t = \frac{k_t - k}{k}$ in the optimality conditions to get a linearized system of equations.
- Postulate specific log-linear form for decision rules $c_t = C(k_t, a_t)$ and $k_t = K(k_t, a_t)$:

$$\widehat{k}_{t+1} = \zeta_{kk} \widehat{k}_t + \zeta_{ka} \widehat{a}_t, \qquad \widehat{c}_t = \zeta_{ck} \widehat{k}_t + \zeta_{ca} \widehat{a}_t$$

  where ζ_{kk}, ζ_{ka}, ζ_{ck} and ζ_{ca} = coefficients to be determined.
- Solve the obtained system of equations $\Longrightarrow$ identify the coefficients ζ_{kk}, ζ_{ka}, ζ_{ck} and ζ_{ca}.

Perturbation methods of higher orders

- **Perturbation is a Taylor's expansion performed numerically.** It is a generalization of the (first-order) log-linearization method to higher orders.
- **Perturbation methods are very fast but the range of their accuracy is uncertain.** This is a local approximation, and the accuracy might deteriorate dramatically away from the steady state.

  JEDC comparison results: 1st- and 2nd-order perturbation methods, PER1 and PER2, of Kollmann, Kim and Kim (2011) produce errors:

  - *on a stochastic simulation up to 6.3% and 1.4%, respectively;*
  - *on a 30% deviation from steady state up to 65% and 50%, respectively.*

- **These accuracy levels are not acceptable**: a method that produces errors of 6% per quarter in the US GDP is not satisfactory (in the same model, CGA produces errors of less than 0.009%).

Characteristic features

- Compute a solution on simulated series.
- Use Monte Carlo integration for approximating conditional expectations.
- Main steps:
  - *Step* 1. Guess a decision rule.
  - *Step* 2. Simulate time series.
  - *Step* 3. Use simulation results to check and to update the guess.
  - Iterate on *Steps* $2-3$ until convergence.

A stochastic simulation method for the growth model

- Parameterize the capital policy function by a polynomial

$$k_{t+1} \simeq \Psi\left(k_t, a_t; \beta\right) = \beta_0 + \beta_1 k_t + \beta_2 a_t + \beta_3 k_t^2 + \beta_4 a_t^2 + ...$$

  and substitute it into the budget constraint to get

$$c_t = \left(1 - d\right) k_t + a_t f\left(k_t\right) - \Psi\left(k_t, a_t; \beta\right).$$

- Fix $\beta \equiv (\beta_0, \beta_1, ...)$. Given shocks $\{a_t\}_{t=0}^T$, simulate $\{c_t, k_{t+1}\}_{t=0}^T$ and construct

$$y_t \equiv \delta \frac{u_1\left(c_{t+1}\right)}{u_1\left(c_t\right)} \left[1 - d + a_{t+1} f'\left(k_{t+1}\right)\right] k_{t+1}.$$

- Regress y_t on $\left(1, k_t, a_t, k_t^2, a_t^2, ...\right) \Longrightarrow$ get $\widehat{\beta}$ (Monte Carlo integration).
- Compute the next-iteration input $\beta^{(j+1)}$ as

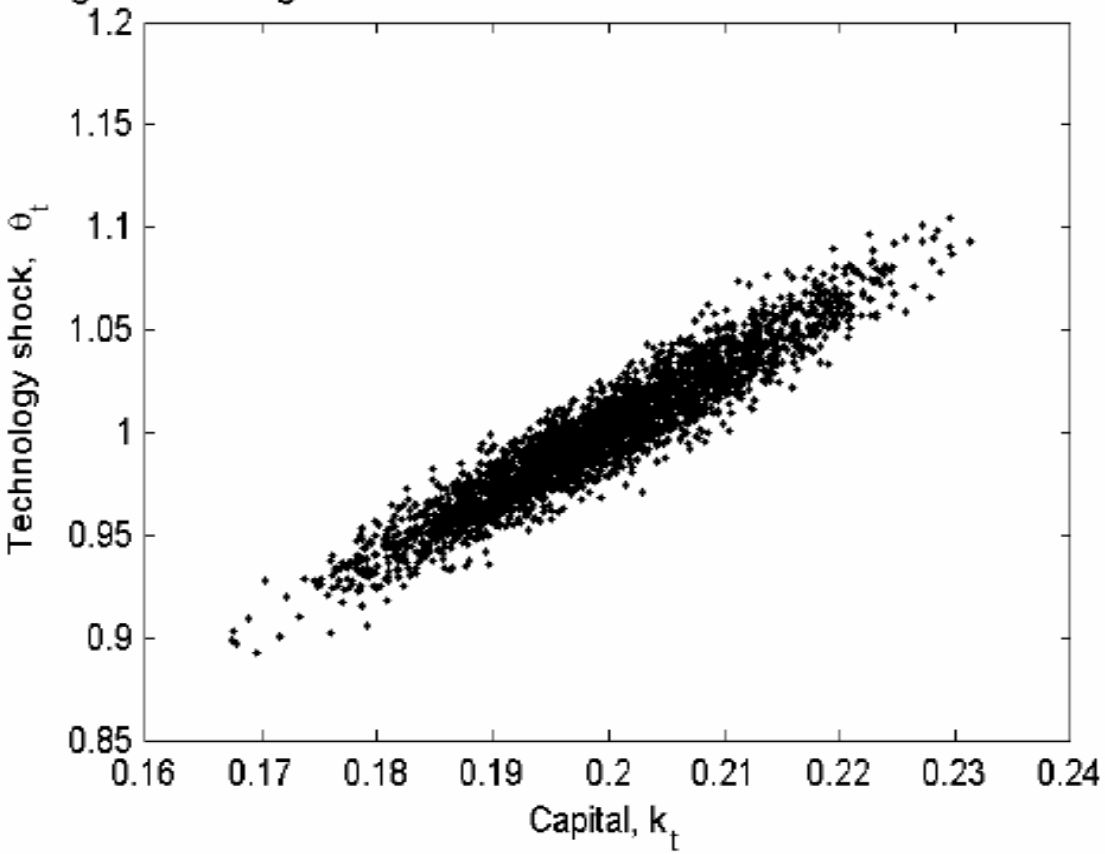
$$\beta^{(j+1)} = \left(1 - \mu\right) \beta^{(j)} + \mu \widehat{\beta},$$

  where $\mu \in (0, 1] =$ damping parameter.

Key advantage of stochastic simulation methods

Stochastic simulation methods have endogenous solution domain: the areas of the state space that are visited in simulation (the ergodic set).
Recall that for projection and perturbation methods: the domain is an exogenous rectangular grid and the steady state point, respectively.

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



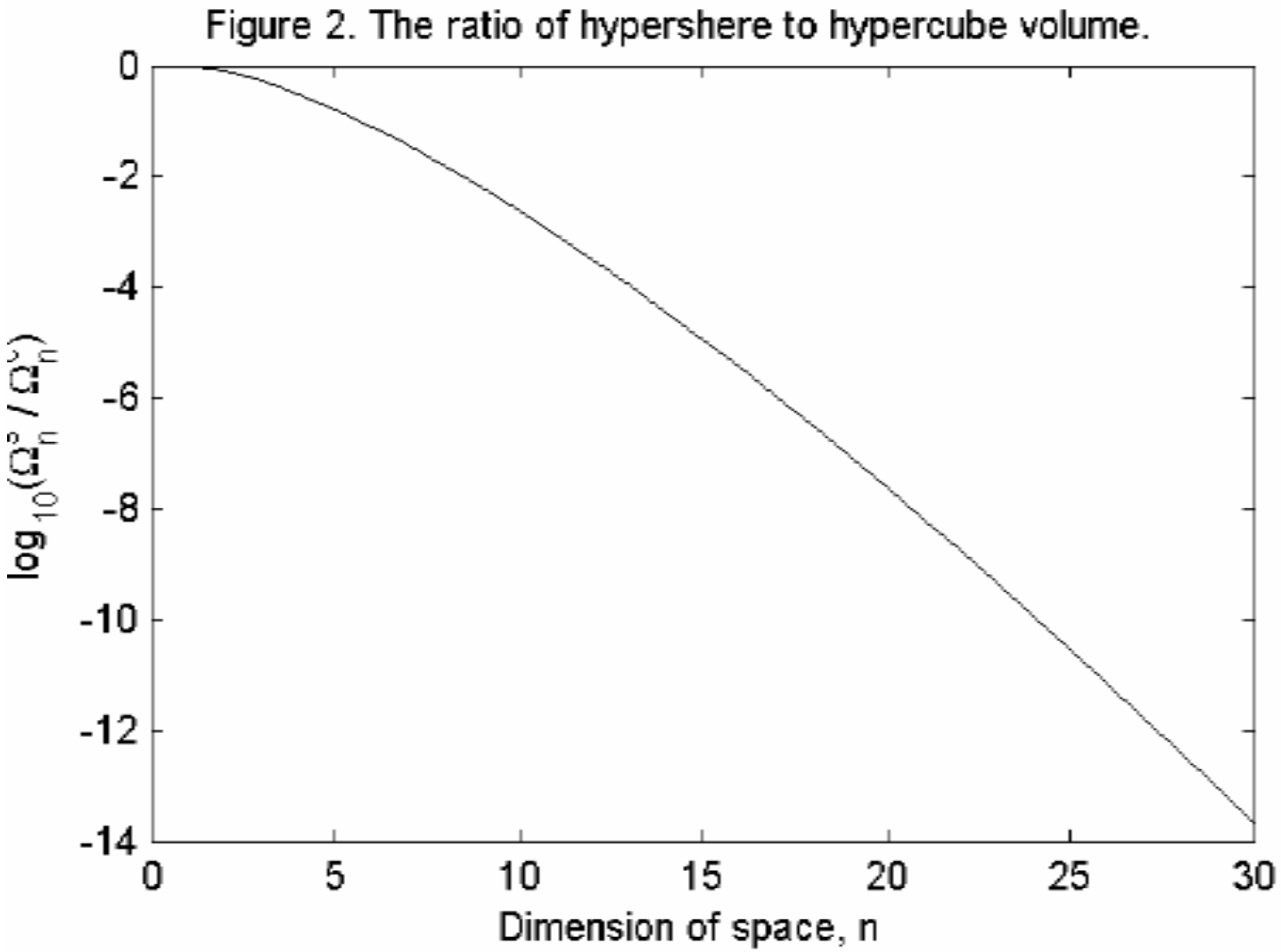
Ratio of hypersphere volume to hypercube volume

- **2-dimensional case:** a circle inscribed within a square occupies about 79% of the area of the square.
- ***p*-dimensional case:** the ratio of a hypersphere's volume Ω_p^s to a hypercube's volume Ω_p^c:

$$\frac{\Omega_p^s}{\Omega_p^c} = \begin{cases} \frac{(\pi/2)^{\frac{p-1}{2}}}{1 \cdot 3 \cdot \ldots \cdot p} \text{ for } p = 1, 3, 5\ldots \\ \\ \frac{(\pi/2)^{\frac{p}{2}}}{2 \cdot 4 \cdot \ldots \cdot p} \text{ for } p = 2, 4, 6\ldots \end{cases}.$$

- **Ratio $\frac{\Omega_p^s}{\Omega_p^c}$ declines rapidly with the dimension of the state space:**
  - when $p = 10$, the ratio $\frac{\Omega_{10}^s}{\Omega_{10}^c} = 3 \cdot 10^{-3}$;
  - when $p = 30$, the ratio $\frac{\Omega_{30}^s}{\Omega_{30}^c} = 2 \cdot 10^{-14}$.

Ergodic set versus tensor-product grid: estimated reduction in cost



Stochastic simulation in problems with high dimensionality

In problems with high dimensionality:

- The hypersphere ergodic set is just a tiny fraction of the hypercube tensor-product grid.
- Stochastic simulation methods are attractive for high-dimensional applications.

But ...

Two drawbacks of stochastic simulation methods

1. **Numerical instability:** den Haan and Marcet (1990) find that

- a simulation-based version of PEA is numerically unstable because of the multicollinearity in regression even under low (2-nd) degree polynomials.

2. **Relatively low accuracy:** In the JEDC 2011's comparison,

- stochastic simulation algorithm (SSA) produces errors of 0.15%;
- PER1 and PER2 produce errors of 6.3% and 1.4%, respectively;
- CGA produces errors of 0.009%.

Projection method on the ergodic set

In this paper,

- We will develop an accurate projection method operating on the ergodic set.
- We will construct a grid of points surrounding the ergodic set using clustering methods.

A grid of points surrounding the ergodic set

A grid of clusters' centers

1. Simulate time series solution to the model (the ergodic set), $\{k_t, a_t\}_{t=1}^{T}$.
2. Construct K clusters using methods from clustering analysis, e.g., hierarchical agglomerative or K-means clustering algorithms.
3. Compute the centers of the constructed clusters.
4. Use the clusters' centers as a grid points in multi-dimensional space.

The ergodic set and principal components

Figure 1a. Ergodic distribution.

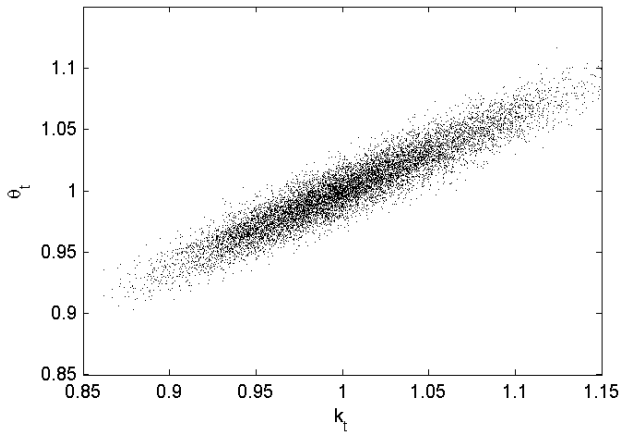


Figure 1b. Normalized ergodic distribution.

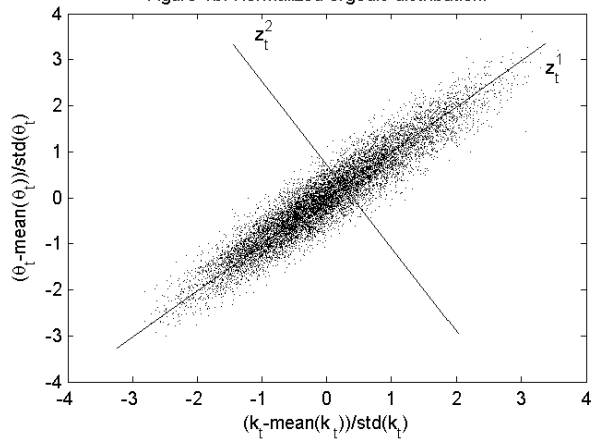


Figure 1c. PCs of ergodic distribution.

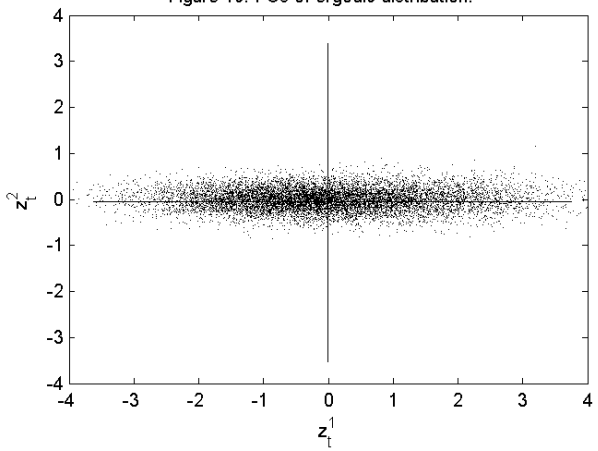
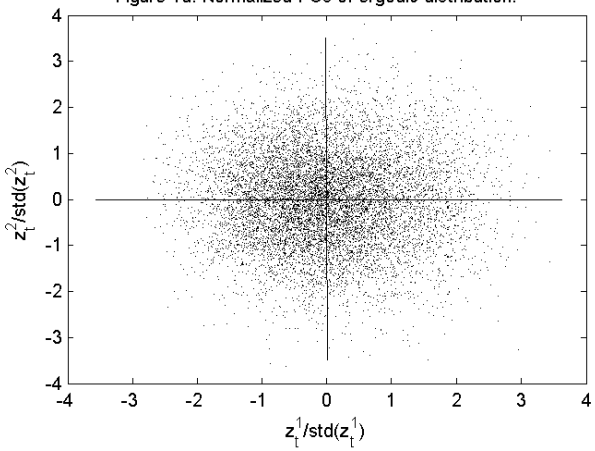


Figure 1d. Normalized PCs of ergodic distribution.



Clusters on principal components of the ergodic set

Figure 4a. Normalized PCs of ergodic distribution: 3 clusters.

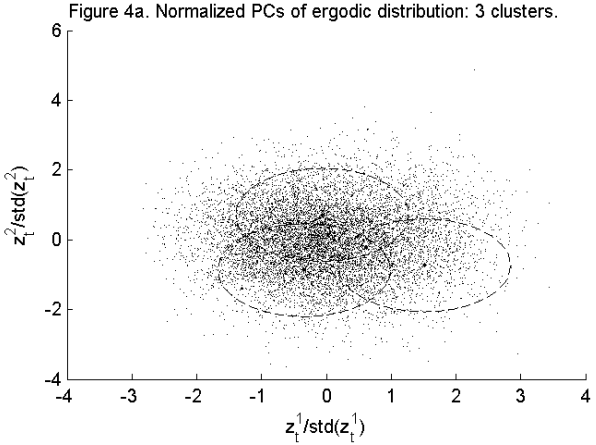


Figure 4b. Normalized PCs of ergodic distribution: 10 clusters.

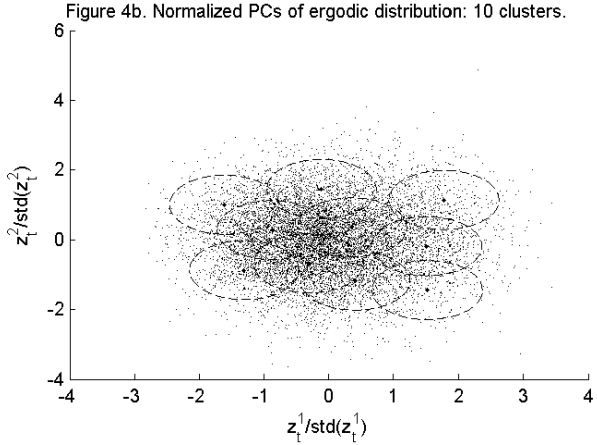


Figure 4c. Normalized PCs of ergodic distribution: 30 clusters.

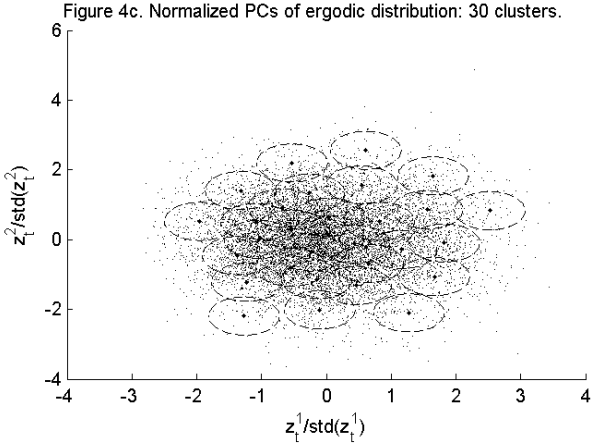
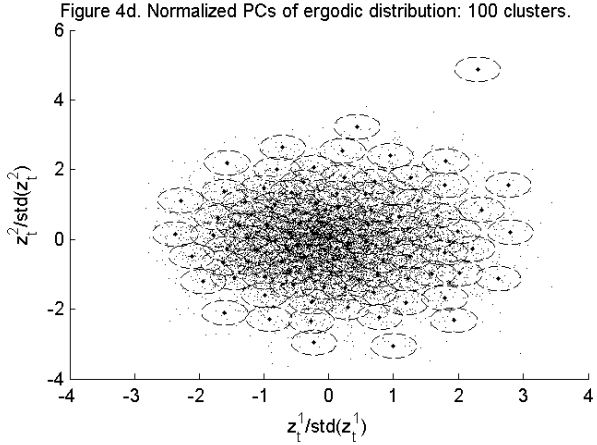


Figure 4d. Normalized PCs of ergodic distribution: 100 clusters.



Properties of the cluster grid

- **The model is solved on the ergodic set** (as is done under stochastic simulation).
- **The cluster grid is more efficient than stochastic simulation:** a large number of closely-situated simulated points is replaced with a smaller number of "representative" points.
- **The cluster grid is (mostly) fixed**, while stochastic simulation algorithms redraw the simulated points on each iteration (numerical stability).
- **The cluster grid is cheap:** constructing 300 clusters on simulated series of 10,000 observations takes:
  - 9 seconds with 2 state variables
  - just 66 seconds with 200 state variables!
- **However, the cluster-grid alone does not prevent the course of dimensionality.**

Multi-dimensional integration: monomial non-product rules

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

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

Gauss-Hermite rule: 2^N nodes

shock 1	shock 2	shock 3
1	1	1
-1	1	1
1	-1	1
-1	-1	1
1	1	-1
-1	1	-1
1	-1	-1
-1	-1	-1

Monomial rule: $2N$ nodes

shock 1	shock 2	shock 3
1	0	0
-1	0	0
0	1	0
0	-1	0
0	0	1
0	0	-1

One-node rule: just 1 node!

0	0	0

Solving for polynomial coefficients: fixed-point iteration

- The cost of Newton's type method grows quickly with dimensionality because of the growing number of terms in Jacobian and Hessian.
- A simple and efficient alternative is fixed-point iteration

$$\beta^{(j+1)} = (1-\mu)\,\beta^{(j)} + \mu\widehat{\beta},$$

  where $\mu \in (0,1)$ is damping parameter.
- Cost of fixed-point iteration grows little with dimensionality.
- Fixed-point iteration works for very high dimensions, like 400 state variables!

The CGA algorithm: putting everything together

Parameterize the RHS of the Euler equation by a polynomial $\Psi\left(k_i, a_i; \beta\right)$,

$$\begin{aligned} k_i' &= E\left\{\frac{\delta u_1\left(c_i'\right)}{u_1\left(c_i\right)}\left[1-d+a_i' f'\left(k_i'\right)\right]k_i'\right\} \\ &\simeq \Psi\left(k_i, a_i; \beta\right) = \beta_0 + \beta_1 k_i + \beta_2 a_i + ... \end{aligned}$$

Step 1. Simulate time series $\left\{k_t, a_t\right\}_{t=0}^{T}$ and construct I clusters. Use clusters' centers $\left\{k_i, a_i\right\}_{i=1}^{I}$ as a grid.
Step 2. Fix $\beta \equiv (\beta_0, \beta_1, \beta_2, ...)$. Given $\left\{k_i, a_i\right\}_{i=1}^{I}$ solve for $\left\{c_i\right\}_{i=1}^{I}$.
Step 3. Compute the expectation using numerical integration (quadrature integration or monomial rules)

$$\widehat{k}_i' \equiv E\left\{\frac{\delta u_1\left(c_i'\right)}{u_1\left(c_i\right)}\left[1-d+a_i' f'\left(k_i'\right)\right]k_i'\right\}.$$

Regress $\widehat{k}_i'$ on $\left(1, k_i, a_i, k_i^2, a_i^2, ...\right) \Longrightarrow$ get $\widehat{\beta}$.
Step 4. Solve for the coefficients using fixed-point iteration with damping,

$$\beta^{(j+1)} = (1-\mu)\,\beta^{(j)} + \mu\widehat{\beta}, \quad \mu \in (0,1).$$

One-country model: parameter choice

- 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 $\gamma \in \{0.2, 1, 5\}$.
- Process for shocks: $\ln a_{t+1} = \rho \ln a_t + \epsilon_{t+1}$, with $\rho = \{0.95, 0.99\}$ and $\sigma = \{0.01, 0.03\}$.
- Discount factor: $\delta = 0.99$.
- Depreciation rate: $d = 0.025$.
- Accuracy is measured by an Euler-equation error,

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

Table 1. The one-agent growth model

Polynomial degree	Mean error	Max error	CPU (sec)
1st degree	−4.32	−3.68	11.59
2nd degree	−6.12	−5.46	0.30
3rd degree	−7.58	−6.93	0.26
4th degree	−8.91	−7.87	0.14
5th degree	−9.99	−8.85	0.24

Mean and Max are unit-free Euler equation errors in log10 units, e.g.,

- -4 means $10^{-4} = 0.0001 \quad (0.01\%)$;
- -4.5 means $10^{-4.5} = 0.0000316 \quad (0.00316\%)$.

Benchmark parameters: $d = 0.025$, $\gamma = 1$, $\rho = 0.95$, $\sigma = 0.01$.

In the paper, many parameterizations are explored:

- low risk aversion: $\gamma = 1/5$;
- high risk aversion: $\gamma = 5$;
- highly persistent shocks: $\rho = 0.99$;
- highly volatile shocks: $\sigma = 0.03$.

For these cases, accuracy and speed are similar.

Multi-country model

The planner maximizes a weighted sum of N countries' utility functions:

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

subject to

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

where v^n is country's n welfare weight.
Productivity of country n follows the process

$$\ln a_t^n = \rho \ln a_{t-1}^n + \epsilon_t^n,$$

where $\epsilon_t^n \equiv \varepsilon_t + \zeta_t^n$ with $\varepsilon_t \sim \mathcal{N}\left(0, \sigma^2\right)$ is identical for all countries and $\zeta_t^n \sim \mathcal{N}\left(0, \sigma^2\right)$ is country-specific.

Table 2. The multi-country model

	Polyn. degree	M1			Q(1)		
		Mean	Max	CPU	Mean	Max	CPU
N=2	1st	−4.09	−3.19	44sec	−4.07	−3.19	45sec
	2nd	−5.45	−4.51	2 min	−5.06	−4.41	1 min
	3rd	−6.51	−5.29	4 min	−5.17	−4.92	2 min
N=20	1st	−4.21	−3.29	20 min	−4.17	−3.28	3 min
	2nd	−5.08	−4.17	5 hours	−4.83	−4.10	32 min
N=40	1st	−4.23	−3.31	5 hours	−4.19	−3.29	2 hours
	2nd	–	–	-	−4.86	−4.48	24 hours
N=100	1st	−4.09	−3.24	10 hours	−4.06	−3.23	36 min
N=200	1st	–	–	-	−3.97	−3.20	2 hours

M1 means monomial integration with 2N nodes; Q(1) means quadrature integration with one node in each dimension; Mean and Max are mean and maximum unit-free Euler equation errors in log10 units, respectively; CPU is running time.

Conclusion

- CGA accurately solves models that were considered to be unfeasible until now.
- A mix of techniques taken together allows us to address the challenges of high-dimensional problems:
  - cluster-grid domain - a tiny fraction of the standard hypercube domain;
  - monomial and one-node integration rules;
  - fixed-point iteration for finding policy functions;
  - iteration-on-allocation and precomputation approaches for solving for intratemporal choice.
- A proper coordination of the above techniques is crucial for accuracy and speed.
- Parallelization and supercomputer (Condor).

Hierarchical clustering algorithm: an example

Data set contains five observations for x_i^1 and x_i^2:

Observation i	Variable	
	x_i^1	x_i^2
1	1	0.5
2	2	3
3	0.5	0.5
4	3	1.6
5	3	1

The Euclidean distance between the observations pairwise:

$$d_{ij} = \left[\left(x_i^1 - x_j^1 \right)^2 + \left(x_i^2 - x_j^2 \right)^2 \right]^{1/2}$$

Hierarchical clustering algorithm: an example (cont.)

- Step 1. Compute pairwise distances between "1", "2", "3", "4", "5":

$$D_1 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & 2.7 & \boxed{0.5} & 2.3 & 2.1 \\ 2 & 2.7 & 0 & 2.9 & 1.7 & 2.2 \\ 3 & \boxed{0.5} & 2.9 & 0 & 2.7 & 2.5 \\ 4 & 2.3 & 1.7 & 2.7 & 0 & 0.6 \\ 5 & 2.1 & 2.2 & 2.5 & 0.6 & 0 \end{array} \Rightarrow$$

Merge "1" and "3" into cluster "6"

- Step 2. Compute pairwise distances between "6", "2", "4", "5":

$$D_2 = \begin{array}{c|cccc} & 6 & 2 & 4 & 5 \\ 6 & 0 & 2.7 & 2.3 & 2.1 \\ 2 & 2.7 & 0 & 1.7 & 2.2 \\ 4 & 2.3 & 1.7 & 0 & \boxed{0.6} \\ 5 & 2.1 & 2.2 & \boxed{0.6} & 0 \end{array} \Rightarrow$$

Merge "4" and "5" into cluster "7"

Hierarchical clustering algorithm: an example (cont.)

- Step 3. Compute pairwise distances between "6", "7", "2":

$$D_3 = \begin{array}{c|ccc} & 6 & 7 & 2 \\ \hline 6 & 0 & 2.1 & 2.7 \\ 7 & 2.1 & 0 & 1.7 \\ 2 & 2.7 & 1.7 & 0 \end{array} \Rightarrow$$

Merge "2" and "7" into cluster "8"

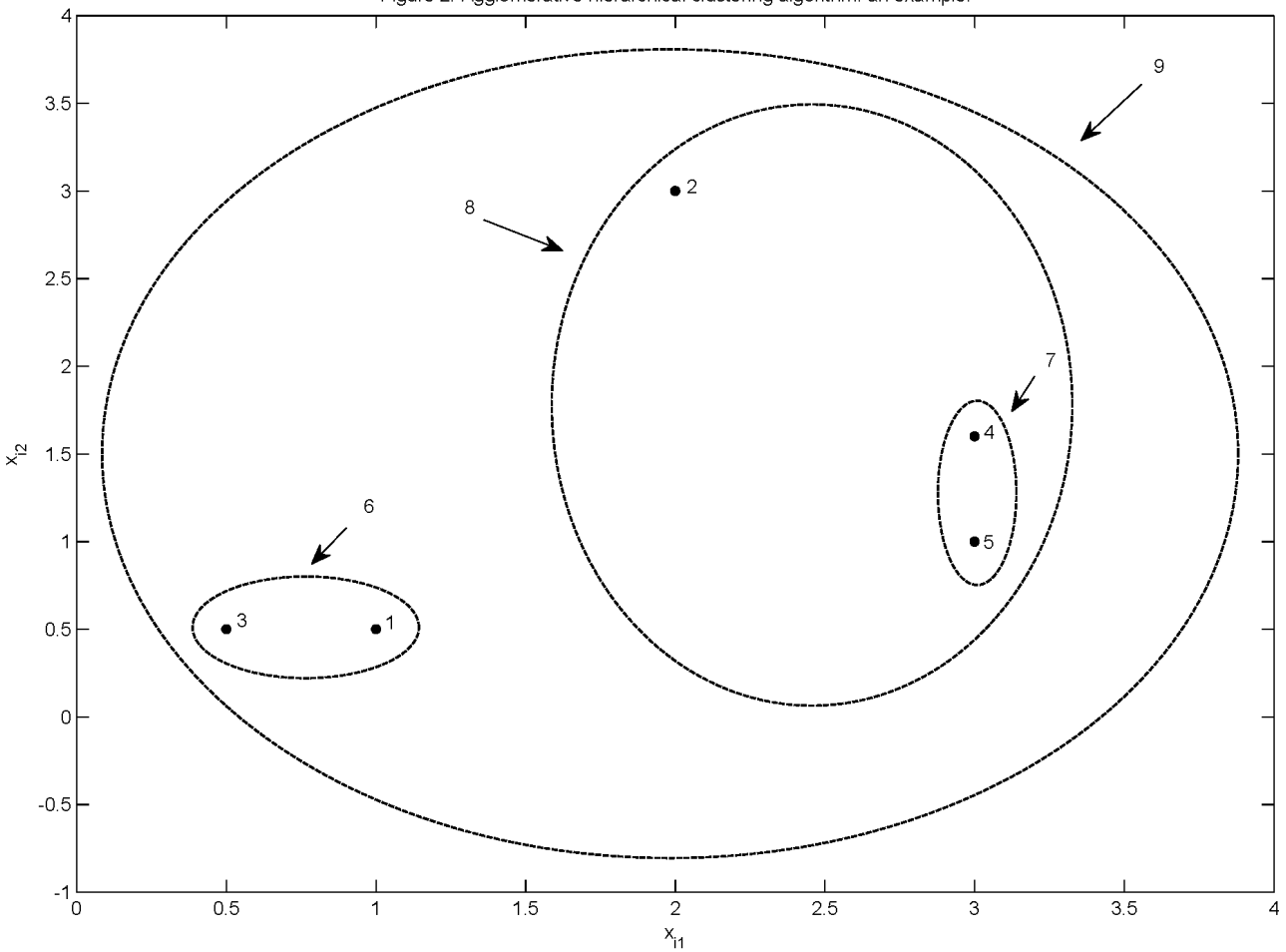
- The resulting hierarchical tree:

Cluster created	Clusters merged		Shortest distance
6	1	3	0.5
7	4	5	0.6
8	2	7	1.7
9	6	8	2.1

- For example, if we want to group the observations into three clusters, we obtain the clusters: $\{1, 3\}$; $\{4, 5\}$; $\{2\}$.

Hierarchical clustering algorithm: an example (cont.)

Figure 2. Agglomerative hierarchical clustering algorithm: an example.



Challenge of finding the intratemporal choice

- Up to this point, we solve for consumption from the budget constraint

$$c_t = (1-d)\,k_t + a_t f\,(k_t) - \Psi\,(k_t, a_t; \beta)\,.$$

- In general, one cannot so trivially solve for the intartemporal choice
- In the literature, the intratemporal choice is solved for as

$$c_t = C\,(k_t, a_t; \beta_c) \qquad \text{similar to } k_{t+1} = \Psi\,(k_t, a_t; \beta)\,.$$

- This become intractable for large-scaled models and involves substantial accuracy loss; see Maliar, Maliar and Judd (2011).

Efficient intratemporal-choice approaches

- Maliar, Maliar and Judd (2011) develop two intratemporal-choice approaches (precomputation and iteration-on-allocation) that separate:
  - the law of motion for the state variables;
  - the static problem of the intratemporal choice.
- Vectorized version of precomputation and iteration-on-allocation:
  - work with vectors and matrices and can solve for the intratemporal choice at all dates / grid points / integration nodes at once;
  - are very accurate and fast in vectorized applications.

Challenge of finding the intratemporal choice: an example

Consider a two-country model with $u\left(c_t^n\right) = \frac{1}{1-1/\gamma_n}\left(c_t^n\right)^{1-1/\gamma_n}$, $n = 1, 2$

$$(\textit{Risk sharing}) \qquad \tau^1\left(c_t^1\right)^{-1/\gamma_1} = \tau^2\left(c_t^2\right)^{-1/\gamma_2}$$

$$(BC) \qquad \sum_{n=1}^2 c_t^n = (1-d)\sum_{n=1}^2 k_t^n + \sum_{n=1}^2 a_t^n f\left(k_t^n\right) - \sum_{n=1}^2 k_{t+1}^n$$

where τ^1, τ^2 are welfare weights. Combining, we get

$$c_t^1 + \left[\frac{\tau^1}{\tau^2}\left(c_t^1\right)^{-1/\gamma^1}\right]^{-\gamma^2} = (1-\delta)\sum_{n=1}^2 k_t^n + ...$$

- No closed-form expression for the intratemporal choice, c_t^1, c_t^2.
- Computing c_t^1 as a function of state variables, $c_t^1\left(k_t^1, k_t^2, a_t^1, a_t^2\right)$ reduces accuracy by an order of magnitude.
- Computing c_t^1 by a Newton's solver at each date (or grid point or integration node) is costly.

Iteration-on-allocation: Maliar, Maliar and Judd (2011)

- A simple vectorized derivative-free solver.
- Cost does not grow much with dimensionality.

In our example, use the optimality conditions to define a mapping:

$$c_t^2 = \left[\frac{\tau^1}{\tau^2} \left(c_t^1 \right)^{-1/\gamma^1} \right]^{-\gamma^2},$$

$$\widetilde{c}_t^1 = \sum_{n=1}^{2} \left[(1-d)\, k_t^n + a_t^n \left(k_t^n \right)^{\alpha} - k_{t+1}^n \right] - c_t^2.$$

Given $a_t^1, a_t^2, k_t^1, k_t^2, k_{t+1}^1, k_{t+1}^2$, *take some value of* c_t^1 *and compute* $\widetilde{c}_t^1$. *Iterate on the consumption* c_t^1 *until convergence (using damping).*

Precomputation approach: Maliar, Maliar and Judd (2011)

- *Idea:* Compute the intratemporal choice functions outside of the main iterative cycle.
- In the main iterative cycle, use the constructed policy functions in the same way we use a closed form solution

$$c_t = (1-d)\,k_t + a_t f\,(k_t) - \Psi\,(k_t, a_t; \beta)\,.$$

- If policy functions are precomputed on the ergodic set, the cost does not grow much with dimensionality.

In our example, take a grid for values for aggregate consumption $C_t = c_t^1 + c_t^2$ *such that* $C_m \in \{C_1, C_2, ..., C_M\}$.
Define the grid function $c^1\,(C_m)$, *by solving for* c^1 for each $m = 1, ..., M$

$$c_t^1 + \left[\frac{\tau^1}{\tau^2}\left(c_t^1\right)^{-1/\gamma^1}\right]^{-\gamma^2} = C_m.$$

Within the main iterative cycle, compute c_t^1, c_t^2 *at each date* t *by interpolation of* $c^1\,(C_m)$.

The CGA algorithm: putting everything together

Parameterize the RHS of the Euler equation by a polynomial $\Psi(k_i, a_i; \beta)$,

$$\begin{aligned} k_i' &= E\left\{\frac{\delta u_1(c_i', \ell_i')}{u_1(c_i, \ell_i)}\left[1 - d + a_i' f'(k_i', \ell_i')\right] k_i'\right\} \\ &\simeq \Psi(k_i, a_i; \beta) = \beta_0 + \beta_1 k_i + \beta_2 a_i + ... \end{aligned}$$

Step 1. Simulate time series $\{k_t, a_t\}_{t=0}^{T}$ and construct I clusters. Use clusters' centers $\{k_i, a_i\}_{i=1}^{I}$ as a grid.

Step 2. Fix $\beta \equiv (\beta_0, \beta_1, \beta_2, ...)$. Given $\{k_i, a_i\}_{i=1}^{I}$ solve for $\{c_i, \ell_i\}_{i=1}^{I}$.

Step 3. Compute the expectation using numerical integration (quadrature integration or monomial rules)

$$\widehat{k}_i' \equiv E\left\{\frac{\delta u_1(c_i', \ell_i')}{u_1(c_i, \ell_i)}\left[1 - d + a_i' f'(k_i', \ell_i')\right] k_i'\right\}.$$

Regress $\widehat{k}_i'$ on $(1, k_i, a_i, k_i^2, a_i^2, ...) \Longrightarrow$ get $\widehat{\beta}$.

Step 4. Solve for the coefficients using fixed-point iteration with damping,

$$\beta^{(j+1)} = (1 - \mu)\beta^{(j)} + \mu\widehat{\beta}, \quad \mu \in (0, 1).$$