Applying the Explicit Aggregation Algorithm to Heterogeneous Agent Models in Continuous Time*

Masakazu Emoto† Takeki Sunakawa‡

First draft: September 2020

This version: April 2021

Abstract

This paper applies the explicit aggregation (XPA) algorithm to the standard heterogeneous agent model with aggregate uncertainty in continuous time. We find that the XPA algorithm is faster in solving the model than the Krusell–Smith algorithm, because the XPA algorithm does not rely on simulations within the algorithm to solve the model. The XPA algorithm is more accurate than the perturbation method when aggregate uncertainty is large.

Keywords: Continuous Time, Heterogeneous Agent Models, Explicit Aggregation Algorithm.

JEL codes: C63; D52

*We thank Eric Young (the editor), anonymous associate editor and referee, Takashi Kamihigashi, Soyoung Lee, Tamotsu Nakamura, Galo Nuño, and Haruki Shibuya for comments and suggestions. The programming code used in the paper is available at `https://github.com/Masakazu-Emoto/XPA-in-Continuous-Time`.

†Graduate School of Economics, Kobe University, 2-1 Rokko-dai, Nada, Kobe, 657-8501; Email: masakazu.emoto@gmail.com

‡Graduate School of Economics, Hitotsubashi University; Email: takeki.sunakawa@gmail.com.

1 Introduction

There is more interest in heterogeneous agent macro models than ever before. Recent studies such as Achdou et al. (2017) and Ahn et al. (2018) apply newly developed numerical methods to solve heterogeneous agent models in continuous time. In particular, Ahn et al. (2018) study heterogeneous agent models with aggregate uncertainty in continuous time and are able to solve the model quickly. However, there are some challenges to their approach. First, their method solves the model using a linear approximation and thus does not capture nonlinear effects of aggregate uncertainty. Second, because of the linearization, the accuracy of solving the model is significantly compromised when aggregate uncertainty is large and/or the model itself is highly nonlinear.

We present an alternative numerical method to address the issues mentioned above. We introduce the explicit aggregation (XPA) algorithm of den Haan and Rendahl (2010) into the standard heterogeneous agent model with aggregate shocks of Krusell and Smith (1998) in continuous time. The XPA algorithm obtains the forecasting rule for aggregate capital from the individual saving function without relying on simulations within the algorithm.[1] Then we compare the XPA algorithm in terms of accuracy and efficiency with the Krusell–Smith (KS) algorithm using simulations and the Reiter–Ahn (REITER) algorithm using perturbations around the stationary distribution without aggregate shocks.[2]

We find that, compared with the KS algorithm, the XPA algorithm is faster than the KS algorithm in solving the standard Krusell–Smith model (5.7 seconds vs. 93.0 seconds in our example) at the same level of accuracy. Compared with the REITER algorithm, the XPA algorithm is more accurate than the REITER algorithm. Although the REITER algorithm can solve the model in a fraction of second, its accuracy is deteriorated especially when aggregate uncertainty is large.

Our study is closely related to at least two areas of research. One is the literature on the XPA algorithm. den Haan and Rendahl (2010) is the first paper to apply this method to the standard heterogeneous agent model in Krusell and Smith (1998). Sunakawa (2020) shows that it is possible to apply their approach to some other heterogeneous agent models such as Khan and

[1] Simulations are a part of the KS algorithm, whereas they are not in the XPA algorithm. See also Section 3.4 for a discussion.

[2] Although we focus on these algorithms, some other useful algorithms are also found in den Haan et al. (2010).

Thomas (2003, 2008) and Krueger et al. (2016) in discrete time. Therefore, it is straightforward to apply the XPA algorithm to more empirically plausible models even in continuous time. To the best of our knowledge, the present paper is the first to apply the XPA algorithm to the standard heterogeneous agent model with aggregate uncertainty in continuous time. Specifically, we use a forecasting rule of $\dot{K}_t$ in continuous time, whereas den Haan and Rendahl (2010); Sunakawa (2020) use that of K_{t+1} in discrete time. One of our contributions is that this alternative forecasting rule in continuous time also works with the XPA algorithm.

The other is the research on methods to solve heterogeneous agent models with aggregate shocks in continuous time. The pioneering research in this area is Ahn et al. (2018), which is also the first to solve the Krusell and Smith (1998) model in continuous time. They adapt the perturbation method originally developed by Reiter (2009) to heterogeneous agent models with aggregate shocks in continuous time.[3] Fernández-Villaverde et al. (2019a) propose a neural-network algorithm to solve heterogeneous agent models with aggregate shocks in continuous time. Our algorithm, unlike Fernández-Villaverde et al. (2019a) and the standard KS algorithm, solves the model without using simulations. Furthermore, whereas the REITER algorithm solves the model using a linear approximation, our algorithm solves the model nonlinearly so as to capture nonlinear effects of aggregate uncertainty.[4] This is especially important when we look at, for example, the effect of aggregate uncertainty on the stochastic steady state (Fernández-Villaverde et al., 2019a).[5] As we will discuss later, the accuracy of solving the model is much higher than what is reported in Ahn et al. (2018). Our results also hold for different degrees of persistence of the aggregate shock.

The paper consists of the following sections. In Section 2, we apply the XPA algorithm, as well as the KS and REITER algorithms, to the Krusell and Smith (1998) model in continuous time. In Section 3, we compare the results of the three algorithms, XPA, KS, and REITER, in terms of

[3] Reiter (2010a); Winberry (2018) further develop a method to reduce the dimension of the state space by projecting the distribution onto principal components. Bayer and Luetticke (2020) and Childers (2018) also suggest novel approaches using linearization.

[4] Reiter's (2010b) backward induction method can also be applied to solve heterogeneous agent models nonlinearly. The method is applied to the stochastic overlapping generations model with aggregate uncertainty of Khan (2017); Kim (2018). Okahata (2018) demonstrates that the method can also be merged with the continuous-time methods.

[5] Perfect foresight methods (used in e.g., McKay et al. (2016); Kaplan et al. (2018)) are also unable to capture this effect. Recently, Schaab (2021) has done such an analysis of continuous time models with aggregate uncertainty by using a sparse grid method developed by Schaab and Zhang (2021).

accuracy and efficiency. Finally, Section 4 concludes.

2 Algorithms

We apply the XPA algorithm first developed by den Haan and Rendahl (2010) to the Krusell and Smith (1998) model in continuous time as studied by Ahn et al. (2018). We choose the Krusell–Smith model as it is known as one of the most popular heterogeneous agent models with aggregate uncertainty. As the model is well known, we defer the details of the model to Appendix A.

KS and XPA algorithms Both the XPA and KS algorithms require two types of calculations, *the inner loop and the outer loop*. The inner loop calculation is common between the XPA and KS algorithms. In continuous time models, a finite difference method is used to solve the Hamilton–Jacobi–Bellman (HJB) equation as in Achdou et al. (2017) for the policy function of household savings. See Appendix A for details.[6]

In the outer loop, the policy function $s(a, z, K_t, Z_t)$ obtained in the inner loop is used to obtain the forecasting rule (perceived law of motion) for the next period's aggregate capital

$$\dot{K}_t = \Gamma(K_t, Z_t). \tag{1}$$

The XPA and KS algorithms are different in how they represent Equation (1). Let $g_t(a, z)$ be the joint distribution of wealth and productivity. In the XPA algorithm, the forecasting rule is obtained by aggregating the policy function with the distribution *explicitly*:

$$\begin{aligned}\Gamma_{\text{XPA}}(K_t, Z_t) &= \sum_z \int_a s(a, z; K_t, Z_t) g_t(a, z) da, \\ &= \sum_z \{s(K_t(z), z, K_t, Z_t) + \xi(z)\}\phi(z),\end{aligned} \tag{2}$$

where $K_t(z)$ is capital conditioned on labor productivity, $\phi(z) = \int g_t(a, z) da$ is the proportion of

[6] Fernández-Villaverde et al. (2019b) solve the standard Krusell–Smith model in continuous time using the KS algorithm. We extend their original programming code especially for the XPA algorithm. See Appendix C and D for the details of computation.

households with z, and $\xi(z)$ is for correcting the biases due to Jensen's inequality. These objects can be calculated immediately when we compute the steady state. We just need to evaluate the policy function at $a = K_t(z)$. In Appendix B, we explain the details of the XPA algorithm following den Haan and Rendahl (2010) and Sunakawa (2020).

In the KS algorithm, taking the policy function as given, we *simulate* the model to obtain the total factor productivity (TFP) sequence Z_t and the mean of the wealth distribution in the next period $\dot{K}_t = \sum_z \int_a s(a, z; K_t, Z_t) g_t(a, z) da$ at each point in time. Then the forecasting rule is obtained by estimating the following forecasting rule using the simulated sequence of $\{K_t, Z_t\}$.

$$\Gamma_{\mathrm{KS}}(K_t, Z_t) = \beta_0 + \beta_1 \ln K_t + \beta_2 Z_t, \tag{3}$$

where $(\beta_0, \beta_1, \beta_2)$ are coefficients estimated by ordinary least squares.

REITER algorithm The REITER algorithm linearizes the model around the stationary distribution obtained from a model without aggregate shocks and uses the system of linearized equations to solve for the dynamics of the economy in the event of aggregate shocks. As a result, the whole distribution is used for the perceived law of motion (forecasting rule). See Ahn et al. (2018) for more details.

The law of motion in the REITER algorithm is obtained as follows. Let g_t be a vector of the wealth–productivity density $g_t(a, z)$ discretized by grid points of (a, z) and $\hat{g}_t = g_t - g$ be its deviation from the stationary distribution. Then we have

$$\frac{\mathrm{d}\hat{g}_t}{\mathrm{d}t} = \tilde{B}_g \hat{g}_t + \tilde{B}_Z Z_t \tag{4}$$

where $\tilde{B}_g$ and $\tilde{B}_Z$ are the matrices obtained by solving the system of linearized equations.[7] Given the sequence of $\{g_t\}$, the sequence of aggregate capital is obtained by $K_t = \int a g_t(a, z) da dz$.

[7] This equation corresponds to Equation (17) in Ahn et al. (2018).

3 Numerical results

We compare the numerical results from three algorithms; XPA algorithm, KS algorithm, and REITER algorithm. The parameter values and detailed settings are found in Appendix F.[8]

3.1 Forecasting rules and simulation paths

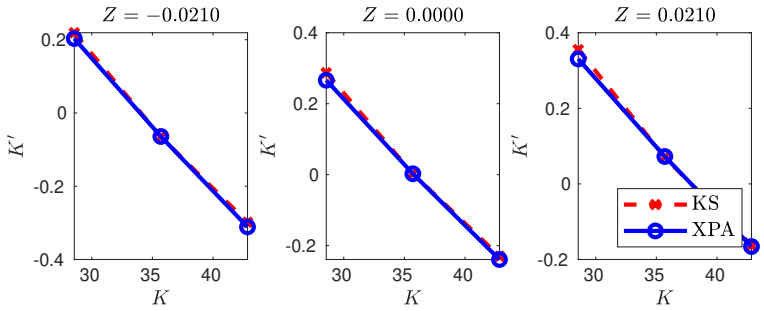
In Figure 1, we show the forecasting rule $\dot{K}_t(K_t, Z_t)$ in KS and XPA.[9] We consider two cases. One is with the standard deviation of the disturbance to TFP being equal to $\sigma = 0.007$, which is the standard value. The other is with $\sigma = 0.05$, which is very large compared to the benchmark case and is intended to show how large differences can be in the extreme case. As is clear from the figure, when $\sigma = 0.007$, there is no significant difference in the forecasting rules obtained by each algorithm. Each forecasting rule is characterized by a decreasing function with respect to capital K_t and an increasing function with respect to TFP Z_t. Even when $\sigma = 0.05$, the forecasting rules in XPA and KS are very similar to each other, although they are slightly different when the values of (K_t, Z_t) are far away from the ones in the stationary distribution.

[8] We use the code developed by Ahn et al. (2018) for REITER. All the codes are modified to use the same set of parameters simply for the purpose of comparison.

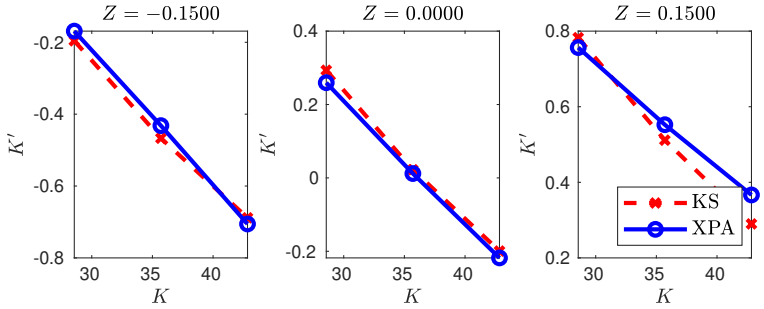
[9] As in Fernández-Villaverde et al. (2019b), we use $\dot{K}(K, Z) = 0$ for all the grid points of (K, Z) as an initial guess for the forecasting rule in both KS and XPA.

Figure 1: Forecasting rules

a. $\sigma = 0.007$



b. $\sigma = 0.05$



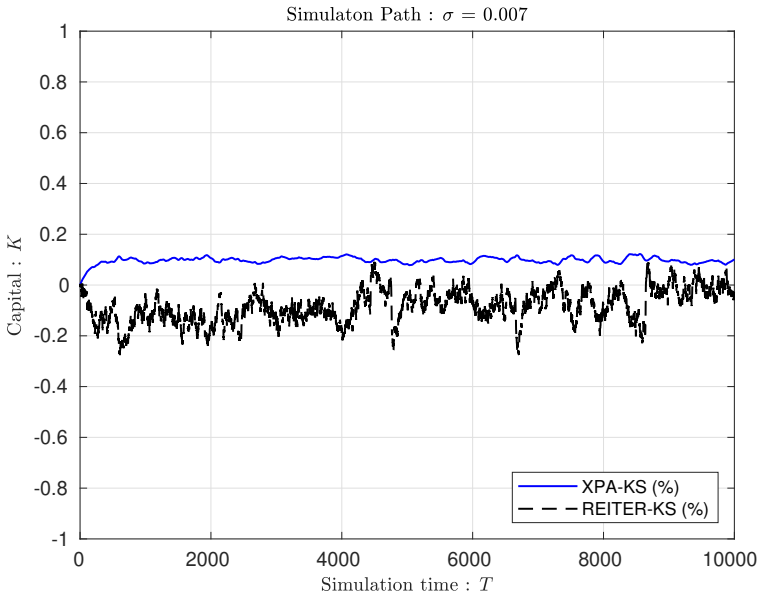
In Figure 2, we show the results of the simulation path derived from the nonlinear model for 10,000 periods in XPA (compared with that in KS), and the simulation path derived from the linearized model (i.e., Equation (4)) in REITER (compared with that in KS).[10] It is clear from the left panel of Figure 2 that the capital path obtained by each algorithm is very similar to each other when $\sigma = 0.007$. In contrast, in the right panel of Figure 2, the capital path obtained by REITER is very different from those by other algorithms when $\sigma = 0.05$.

The capital path in XPA tend to have an upward bias compared with that in KS. The more σ increases, the larger the bias is (about 1% when $\sigma = 0.05$). However, each algorithm is accurate in terms of the Den Haan errors as shown later in Figure 3.

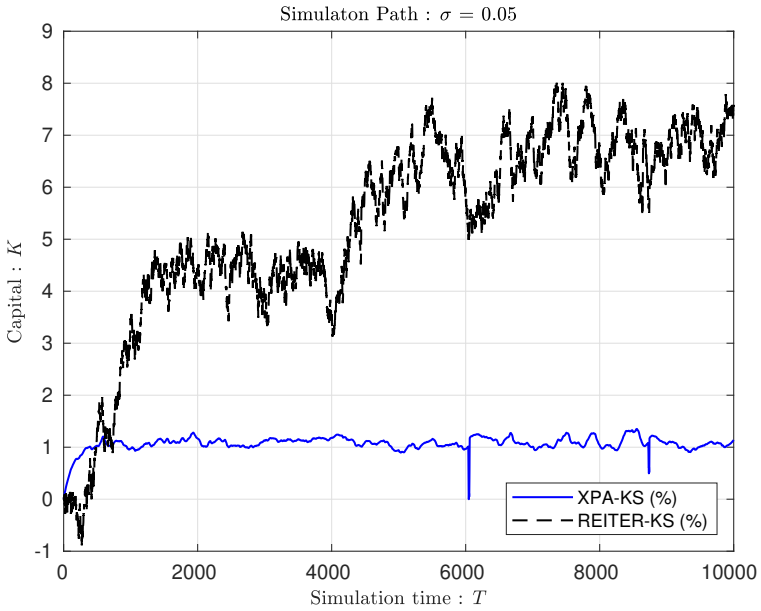
[10] In all the algorithms, the time interval $dt = 0.25$ used in the simulation is the same as the time interval of Ahn et al. (2018). In KS and XPA, we use a bilinear interpolation for aggregate capital and TFP to calculate the wealth–productivity density at each point of time. See Appendix D.

Figure 2: Simulation paths

a. $\sigma = 0.007$



b. $\sigma = 0.05$



3.2 Efficiency

Comparing the results of XPA and KS in Table 1, we can see that XPA solves the model much faster than KS because it does not use simulations. Comparing the computation times of XPA and REITER, we can see that REITER is also faster than XPA. However, the difference between the two is not that large (XPA: 5.7 seconds vs. REITER: 0.3 seconds).[11]

Table 1: Computation time

Algorithm	Computation time
XPA (Our method)	5.714 sec
Krusell-Smith	92.964 sec
REITER (Ahn et al., 2018)	0.296 sec

Notes: Computations are done on a workstation with Intel Xeon E5–2699 v4 (2.20Ghz) and 32GB RAM using MATLAB R2020b. Computation time is the average of 10 runs and excludes the time spent for simulations.

3.3 Accuracy

We check the accuracy of solving the model using the Den haan errors proposed by den Haan (2010). In the present paper, we simulate two sequences of 10,000 time periods with each algorithm. One is $\{\tilde{K}_t\}_{t\in[0,T]}$ obtained only from the forecasting rule for each algorithm (Equations (2)–(4)), and the other is $\{K_t^*\}_{t\in[0,T]}$ obtained from the fully nonlinear model including the the household HJB equation and the Kolmogorov Forward equation.[12] Then we use the sequences of $\{K_t^*\}_{t\in[0,T]}$ and $\{\tilde{K}_t\}_{t\in[0,T]}$ to measure the Den haan errors

$$\varepsilon_{DH}^{MAX} \equiv 100 \cdot \max_{t\in[0,T]} |\ln \tilde{K}_t - \ln K_t^*|.$$

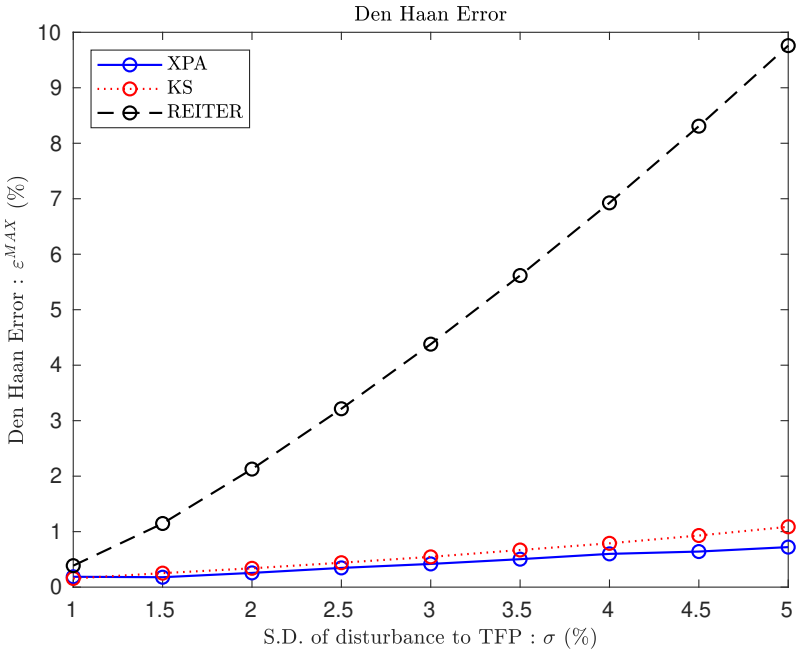
Figure 3 summarize the Den haan errors for each algorithm. It is clear that when σ is small, there is not much difference in the Den haan errors. However, when σ is large, the Den haan errors

[11] Note that we use MATLAB and no parallelization, so the gap might be smaller when we use a faster language and/or parallelization.

[12] As in Ahn et al. (2018); Bayer and Luetticke (2020), to obtain the sequence of $\{K_t^*\}$ in REITER, we solve for the optimal savings plan $s_t^*(a_j, z_i)$ at every pair of the wealth and productivity values (a_j, z_i) in the histogram under the equilibrium factor prices (w_t, r_t) in every period t. See Appendix E.

for REITER are considerably larger than those for XPA and KS. Furthermore, the Den Haan errors of XPA are smaller than those of KS.[13][14]

Figure 3: Comparison of maximum of Den Haan errors



3.4 Discussion

Time for simulation Simulations are a part of the KS algorithm, whereas they are not in the XPA algorithm. Specifically, simulations are done in KS to update the forecasting rules in the outer loop. The outer loop in XPA is much faster than in KS, as simulations are unnecessary in XPA.

The computation time in Table 1 is solely for solving the model and includes time for simulations for the KS algorithm only. Having said that, to get the simulated paths in Figure 2, simulations in XPA are as slow as in KS. Indeed, taking the converged forecasting rule by each algorithm as given, simulation time in XPA is almost identical to that in KS (about 10 seconds to compute a sequence of 10,000 time periods in Figure 2).

[13] However, there is a possibility of using non-parametric forecasting rules or forecasting rules with higher-order terms for KS to achieve higher accuracy.

[14] We also confirm that our results hold when the persistence of TFP, $1-\mu$, is lowered from $\mu = 0.25$ to $\mu = 0.5$ or $\mu = 0.75$. See Appendix F.

Market clearing It is nontrivial to handle market clearing conditions in XPA or KS. Sunakawa (2020) shows a detailed explanation how to apply XPA in discrete time with market clearing in lumpy investment models developed in Khan and Thomas (2003, 2008).[15] XPA can solve the model faster with a similar degree of accuracy to KS. XPA (or KS) in continuous time should work in a similar way, although it takes more time to solve the model than REITER does. In REITER, however, market clearing conditions hold only at the stationary distribution.

4 Conclusion

In this paper, we apply the explicit aggregation (XPA) algorithm proposed by den Haan and Rendahl (2010) to the standard heterogeneous agent model with aggregate uncertainty (Krusell and Smith, 1998) in continuous time. Using XPA, we can get an accuracy close to that of KS with a similar speed to that of REITER. Future research will show that XPA can also be useful in models such as ones with uncertainty shocks (Bloom et al., 2018) where linearization-based methods fail.

References

Achdou, Y., J. Han, J.-M. L. Lasry, P.-L. L. Lions, and B. Moll (2017): "Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach," NBER Working Papers 23732, National Bureau of Economic Research, Inc.

Ahn, S., G. Kaplan, B. Moll, , T. Winberry, and C. Wolf (2018): "When Inequality Matters for Macro and Macro Matters for Inequality," *NBER Macroeconomics Annual*, 1–75.

Bayer, C. and R. Luetticke (2020): "Solving heterogeneous agent models in discrete time with many idiosyncratic states by perturbation methods," *Quantitative Economics*, 11, 1253–1288.

Bloom, N., M. Floetotto, N. Jaimovich, I. Saporta-Eksten, and S. J. Terry (2018): "Really Uncertain Business Cycles," *Econometrica*, 86, 1031–1065.

[15] In these models, taking the shadow price of consumption as given, we obtain the aggregate consumption and implied new shadow price. We solve for the fixed point of the shadow price at each grid point in the outer loop in XPA.

Childers, D. (2018): "Solution of Rational Expectations Models with Function Valued States," Working Paper.

den Haan, W. J. (2010): "Comparison of Solutions to the Incomplete Markets Model with Aggregate Uncertainty," *Journal of Economic Dynamics and Control*, 34, 4–27.

den Haan, W. J., K. L. Judd, and M. Juillard (2010): "Computational suite of models with heterogeneous agents: Incomplete markets and aggregate uncertainty," *Journal of Economic Dynamics and Control*, 34, 1–3.

den Haan, W. J. and P. Rendahl (2010): "Solving the Incomplete Markets Model with Aggregate Uncertainty Using Explicit Aggregation," *Journal of Economic Dynamics and Control*, 34, 69–78.

Fernández-Villaverde, J., S. Hurtado, and G. Nuño (2019a): "Financial Frictions and the Wealth Distribution," NBER Working Papers 26302, National Bureau of Economic Research, Inc.

——— (2019b): "Solving the Krusell-Smith (1998) model," Manuscript.

Kaplan, G., B. Moll, and G. L. Violante (2018): "Monetary Policy According to HANK," *American Economic Review*, 108, 697–743.

Khan, A. (2017): "Large Recessions in an Overlapping Generations with Unemployment," 2017 Meeting Papers 1559, Society for Economic Dynamics.

Khan, A. and J. K. Thomas (2003): "Nonconvex Factor Adjustments in Equilibrium Business Cycle Models: Do Nonlinearities Matter?" *Journal of Monetary Economics*, 50, 331–360.

——— (2008): "Idiosyncratic Shocks and the Role of Nonconvexities in Plant and Aggregate Investment Dynamics," *Econometrica*, 76, 395–436.

Kim, H. (2018): "Inequality, Portfolio Choice, and the Business Cycle," Working Paper.

Krueger, D., K. Mitman, and F. Perri (2016): "Macroeconomics and Household Heterogeneity," in *Handbook of Macroeconomics*, ed. by J. B. Taylor and H. Uhlig, Elsevier, vol. 2, 843–921.

Krusell, P. and A. A. J. Smith (1998): "Income and Wealth Heterogeneity in the Macroeconomy," *Journal of Political Economy*, 106, 867–896.

McKay, A., E. Nakamura, and J. Steinsson (2016): "The Power of Forward Guidance Revisited," *American Economic Review*, 106, 3133–3158.

Okahata, N. (2018): "An Alternative Solution Method for Continuous-Time Heterogeneous Agent Models with Aggregate Shocks," Mimeo.

Reiter, M. (2009): "Solving heterogeneous-agent models by projection and perturbation," *Journal of Economic Dynamics and Control*, 33, 649–665.

——— (2010a): "Approximate and Almost-Exact Aggregation in Dynamic Stochastic Heterogeneous-Agent Models," Economics Series 258, Institute for Advanced Studies.

——— (2010b): "Solving the incomplete markets model with aggregate uncertainty by backward induction," *Journal of Economic Dynamics and Control*, 34, 28–35.

Schaab, A. (2021): "Micro and Macro Uncertainty," Mimeo.

Schaab, A. and A. Zhang (2021): "Adaptive Sparse Grid Methods for Differential Equations in Economics," Mimeo.

Sunakawa, T. (2020): "Applying the Explicit Aggregation Algorithm to Heterogeneous Macro Models," *Computational Economics*, 55, 845–874.

Winberry, T. (2018): "A Method for Solving and Estimating Heterogeneous Agent Macro Models," *Quantitative Economics*, 9, 1123–1151.

Young, E. R. (2005): "Approximate Aggregation," Mimeo, University of Virginia.

Appendix (not for publication)

A The Krusell–Smith model in continuous time

A.1 Environments

Households

There is a continuum of households with a normalized fixed mass indexed by $j \in [0, 1]$. Households face idiosyncratic uncertainty regarding labor productivity and the borrowing constraint $a_{jt} \geq 0$, where a_{jt} is the value of asset holdings for household j in period t. There are two states of labor productivity z_{jt} for each household, z_e and z_u, which follow the Poisson process with arrival rates λ_e and λ_u. z_e shows that the household is employed and z_u indicates that the household is unemployed. If the household is employed, she/he receives labor income after taxation $(1 - \tau)w_t$. When the household is unemployed, she/he receives unemployment insurance bw_t financed by the labor income tax.

Each household $j \in [0, 1]$ chooses their consumption and savings (c_{jt}, a_{jt}) in each period $t \geq 0$ to maximize their expected life-time utility by taking the wage rate w_t and the interest rate r_t as given.

$$
\begin{aligned}
& v_{j0} = \max \mathbb{E}_0 \left[\int_0^\infty e^{-\rho t} \frac{c_{jt}^{1-\gamma}}{1-\gamma} dt \right], \\
\text{s.t.} \quad & \mathrm{d}a_{jt} = (r_t a_{jt} + (1-\tau) z_{jt} w_t + (1 - z_{jt}) b w_t - c_{jt}) \mathrm{d}t, \ a_{jt} \geq 0, \\
& z_{jt} \in \{z_e, z_u\}, \ z_e = 1, \ z_u = 0.
\end{aligned}
$$

The instantaneous utility function is of constant-relative-risk-aversion form, ρ is the rate of time preference, and γ is the degree of relative risk aversion. The population of households with a pair of wealth and productivity levels is time-variant and given by $g_t(a, z)$. For notational convenience, we drop time subscripts t from variables at the individual level hereafter.

Firm

The representative firm produces the final good Y_t using capital K_t and labor L_t. The production function is Cobb–Douglas form

$$Y_t = e^{Z_t} K_t^{\alpha} L_t^{1-\alpha},$$

where α is the capital share. Z_t is the logarithm of total factor productivity (TFP) following the Ornstein–Uhlenbeck process

$$dZ_t = \mu(\bar{Z} - Z_t)dt + \sigma dW_t, \quad \bar{Z} = 0,$$

where dW_t follows a Wiener process. $1 - \mu$ is the persistence of TFP and σ is the volatility of the TFP. This process is similar to an AR(1) process in discrete time.

The wage rate and the interest rate are obtained from the first-order conditions for the profit maximization problem as follows:

$$w_t = (1-\alpha)e^{Z_t} K_t^{\alpha} L_t^{-\alpha}, \;\; r_t = \alpha e^{Z_t} K_t^{\alpha-1} L_t^{1-\alpha} - \delta,$$

where δ is the depreciated rate of capital.

Government

The government imposes a tax on labor income to finance unemployment compensation. The government's budget is balanced as below

$$\tau w_t \phi(z_e) = b w_t \phi(z_u).$$

That is, the government's tax revenue from labor income is equal to the government's expenditure to finance unemployment insurance. $\phi(z_e) = \int g_t(a, z_e)da$ is the share of employment and $\phi(z_u) =$

$\int g_t(a, z_u)da$ is the share of unemployment in the economy.

Equilibrium

An equilibrium in this economy is consist of a set of prices $\{w_t, r_t\}$, quantities $\{K_t, L_t\}$ and a density $\{g_t(a, z)\}$ such that

1. Given w_t, r_t, the policy functions for consumption and savings, $c_t^*(a, z)$ and $s_t^*(a, z)$, are the solution of the the Hamilton–Jacobi–Bellman (HJB) equation

$$\begin{aligned}\rho v_t(a, z) = \max_c u(c) + \partial_a v_t(a, z)(r_t a + (1 - \tau)z w_t + (1 - z)b w_t - c) \\ + \lambda_z(v_t(a, z') - v_t(a, z)) + \frac{1}{dt}\mathbb{E}_t[dv_t(a, z)],\end{aligned} \tag{5}$$

   where $u(\cdot)$ is the utility function and $v_t(\cdot)$ is the value function in period t, which depends on a particular realization of the aggregate state $(g_t(a, z), Z_t)$.

2. The sequence of $\{g_t(a, z)\}$ is the solution of the Kolmogorov Forward (KF) equation

$$\frac{\mathrm{d}g_t(a, z)}{\mathrm{d}t} = -\partial_a\left[s_t(a, z)g_t(a, z)\right] - \lambda_z g_t(a, z) + \lambda_{z'} g_t(a, z') \tag{6}$$

   where $s_t(a, z) = s_t^*(a, z)$ is the optimal saving policy function corresponding to the household optimization problem.

3. The wage rate and the interest rate are given by

$$w_t = (1 - \alpha)e^{Z_t}K_t^{\alpha}L_t^{-\alpha}, \;\; r_t = \alpha e^{Z_t}K_t^{\alpha-1}L_t^{1-\alpha} - \delta.$$

4. The government's budget constraint is satisfied as

$$\tau w_t \phi(z_e) = b w_t \phi(z_u),$$

   where $\phi(z_e) = \int_a g_t(a, z_e)da$, $\phi(z_u) = \int_a g_t(a, z_u)da$ are time-invariant distributions of pro-

ductivity.

5. The capital and the labor markets clear

$$K_t = \sum_z \int_a a g_t(a,z) da, \;\; L_t = \sum_z \int_a z g_t(a,z) da.$$

HJB equation in the XPA and KS algorithms

In the XPA and KS algorithms, we assume *approximate aggregation* (Young, 2005) so that the wealth–productivity density $\{g_t(a,z)\}$ as a state variable is approximated by the mean K_t so that the aggregate state is (K_t, Z_t). In this case, the value function is denoted by $v(a,z,K_t,Z_t)$ and the HJB equation can be written as

$$\begin{aligned}\rho v(a,z,K_t,Z_t) = \max_c \frac{c^{1-\gamma}-1}{1-\gamma} &+ v_a(a,z,K_t,Z_t)\dot{a} + \lambda_z(v(a,z',K_t,Z_t) - v(a,z,K_t,Z_t)) \\ &+ v_K(a,z,K_t,Z_t)\dot{K}_t \\ &+ v_Z(a,z,K_t,Z_t)(-\mu Z_t) + \frac{\sigma^2}{2} v_{ZZ}(a,z,K_t,Z_t) \end{aligned} \tag{7}$$

subject to the budget constraint $\dot{a} = r_t a + (1-\tau)z w_t + (1-z)b w_t - c$ and the forecasting rule $\dot{K}_t = \Gamma(K_t, Z_t)$ for the policy function of household savings, $s(a,z,K_t,Z_t)$. v_a, v_K, and v_Z are the first order derivatives of the value function in terms of a, K, and Z and v_{ZZ} is the second order derivative of the value function in terms of Z.

B Details of the XPA algorithm

We provide details of the XPA algorithm following den Haan and Rendahl (2010) and Sunakawa (2020). In contrast with the KS algorithm, the XPA algorithm calculates the forecasting rules without simulations within the algorithm. First, we rewrite the wealth–productivity density $g_t(a,z)$

using the conditional probability as

$$\begin{aligned} g_t(a|z) &= \frac{g_t(a,z)}{\int g_t(a,z)da} = \frac{g_t(a,z)}{\phi(z)} \\ \Leftrightarrow g_t(a,z) &= g_t(a|z)\phi(z) \end{aligned}$$

where $\phi(z) = \int g_t(a,z)da$ is equal to the proportion of households with labor productivity z in the economy.[16] Then we can rewrite the forecasting rule using the conditional distribution of wealth as

$$\begin{aligned} \dot{K}(K_t, Z_t) &= \sum \int s(a,z;K_t,Z_t)g_t(a,z)da \\ &\approx \sum s\left(\int a g_t(a|z)da, z; K_t, Z_t\right)\phi(z) \\ &= \sum s(K_t(z), z; K_t, Z_t)\phi(z) \end{aligned}$$

where $K_t(z) = \int a g_t(a|z)da$ is capital conditioned on labor productivity z. We *assume* that the household's policy function $s(a,z;K_t,Z_t)$ is linear at $a = K_t(z)$ so that $\int s(a,z;K_t,Z_t)g(a|z)da \approx s\left(K_t(z),z;K_t,Z_t\right)$ holds. We compute $K_t(z)$ by the following equations:

$$\begin{aligned} K_t(z) &= \psi(z)K_{ss}, \ \ \psi(z) \equiv \frac{K_{ss}(z)}{K_{ss}} = \frac{K_{ss}(z)}{\sum_z \int a g_{ss}(a,z)da}, \\ K_{ss}(z) &= \int a g_{ss}(a|z)da = \frac{\int a g_{ss}(a,z)da}{\phi(z)}, \end{aligned}$$

where K_{ss} and g_{ss} are capital and wealth distribution at the steady state without aggregate uncertainty. We assume that $\psi(z)$ is constant over aggregate fluctuations. Note that the ratio of the capital conditioned on z to aggregate capital, $\psi(z) = K_{ss}(z)/K_{ss}$, can be easily obtained in the steady-state calculation.

Moreover, following den Haan and Rendahl (2010), we conduct bias correction. As we assume that the policy function is linear at $a = K_t(z)$, there may be bias in the forecasting rule from

[16] We assume the share of employment $\phi(z_e)$ $(= 1 - \phi(z_u))$ is time-invariant, although it is straightforward to make the employment measure be time-variant and depend on aggregate uncertainty as in Krusell and Smith (1998). See also Sunakawa (2020).

Jensen's inequality. We compute the steady-state counterparts to correct the bias:

$$\begin{aligned}\xi(z) &= \dot{K}_{ss}(z) - s_{ss}(K_{ss}(z), z), \\ \dot{K}_{ss}(z) &= \int s_{ss}(a,z) g_{ss}(a|z) da = \frac{\int s_{ss}(a,z) g_{ss}(a,z) da}{\phi(z)}.\end{aligned}$$

Again, $\xi(z)$ can be computed at negligible cost in the steady state. Finally, we can write the forecasting rule as follows

$$\dot{K}(K_t, Z_t) = \sum_z \{s(K_t(z), z, K_t, Z_t) + \xi(z)\}\phi(z). \tag{8}$$

That is, to obtain the forecasting rule, we just need to evaluate the policy function at $a = K_t(z)$. The value of $K_t(z)$ may not be on the grid of a, so we use linear interpolation.

We have two important assumptions to derive Equation (8). One is that the policy function is linear at $a = K_t(z)$. The other is that the ratio of capital conditioned on z to aggregate capital, $\psi(z) = K_t(z)/K_t$, is constant over aggregate fluctuations. The first assumption is acceptable especially in the household consumption-saving problem in Krusell and Smith (1998), in which the saving function is almost linear except for near the origin where poor households face borrowing constraints. Sunakawa (2020) shows that the algorithm also works with nonlinear (S, s)–type policy functions such as in Khan and Thomas (2003, 2008).

Also, Sunakawa (2020) shows that we can incorporate more than two states for the idiosyncratic shock by using the *epsilon-indexed* aggregation. Note that the original analysis by den Haan and Rendahl (2010) has dealt with only two states for the idiosyncratic shock in the Krusell and Smith (1998) model (i.e., employment and unemployment) and they do so by increasing the number of aggregate states indexed by employment status, $K(z)$ for $z \in \{e, u\}$.[17] Instead, we do not need to increase the number of aggregate states with the epsilon-indexed approach, as shown in Equation (8). The present paper shows that we can also use the epsilon-indexed aggregation in the continuous-time framework as in Sunakawa (2020).

[17] The policy function becomes the form of $s(k, z, K_t(e), K_t(u), Z_t)$.

Summary of XPA Algorithm

In summary, we perform computations to solve the Krusell–Smith model with the XPA algorithm as follows. As mentioned above, the XPA algorithm is fast because it does not use any simulations.

1. Compute the stationary distribution without aggregate uncertainty to obtain the conditional capital ratio $\psi(z)$ and the bias correction term for correcting the forecasting rule $\xi(z)$.

2. (Inner loop) Solve the HJB equation for the policy function taking the forecasting rule as given. See Appendix C for details.

3. (Outer loop) Compute the forecasting rule without simulations taking the policy function as given. The bias correction is also done.

4. Repeat steps 2–3 until the forecasting rule converges.

C Solving the HJB equation with the upwind scheme

Fernández-Villaverde et al. (2019b) solve the standard Krusell–Smith model in continuous time using the KS algorithm. We extend their original programming code in the following two dimensions: (i) We use the upwind scheme not only individual wealth, a, but also K and Z, which is necessary for the XPA algorithm.[18] (ii) We exclude the direct effect of aggregate variables K and Z on the transition matrix for simulation.[19]

We discretize the individual wealth and productivity by grid points (a_j, z_i) for $i = 1, 2$ and $j = 1, ..., J$ so that $g_{t,i,j} = g_t(a_j, z_i)$ holds. We also discretize the aggregate capital and productivity by grid points (K_l, Z_m) for $l = 1, ..., L$ and $m = 1, ..., M$ so that $v_{i,j,l,m} = v(a_j, z_i, K_l, Z_m)$ holds. We are going to solve the HJB equation (7) with an iterative method. Let $v^n_{i,j,l,m}$ be the value function at n–th iteration. Then we have (note that we use the implicit method so that $t + 1$

[18] As shown in Appendix D, the upwind scheme for K and Z also improves the accuracy of the KS algorithm.

[19] In the current and next sections, we acknowledge referring to Fernández-Villaverde et al. (2019b) (hereafter FVHN) for some derivations and notations.

variables are in the right hand side)

$$\begin{aligned}\frac{v^{n+1}_{i,j,l,m}-v^{n}_{i,j,l,m}}{\Delta}+\rho v^{n+1}_{i,j,l,m} =& \frac{(c^{n}_{i,j,l,m})^{1-\gamma}-1}{1-\gamma}+\partial_a^{(F)}v^{n+1}_{i,j,l,m}s^{n}_{i,j,l,m,F}\mathbf{1}_{s^{n}_{i,j,l,m,F}>0}+\partial_a^{(B)}v^{n+1}_{i,j,l,m}s^{n}_{i,j,l,m,B}\mathbf{1}_{s^{n}_{i,j,l,m,B}<0}\\ &+\lambda_i(v^{n+1}_{-i,j,l,m}-v^{n+1}_{i,j,l,m})+\partial_K^{(F)}v^{n+1}_{i,j,l,m}h_{l,m}\mathbf{1}_{h_{l,m}>0}+\partial_K^{(B)}v^{n+1}_{i,j,l,m}h_{l,m}\mathbf{1}_{h_{l,m}<0}\\ &+\partial_Z^{(F)}v^{n+1}_{i,j,l,m}\theta(\bar{Z}-Z_m)\mathbf{1}_{\bar{Z}-Z_m>0}+\partial_Z^{(B)}v^{n+1}_{i,j,l,m}\theta(\bar{Z}-Z_m)\mathbf{1}_{\bar{Z}-Z_m<0}+\frac{\sigma^2}{2}\partial^2_{ZZ}v^{n+1}_{i,j,l,m}\end{aligned} \tag{9}$$

where

$$\begin{aligned}\partial_a^{(F)}v^{n+1}_{i,j,l,m}&=\frac{v^{n+1}_{i,j+1,l,m}-v^{n+1}_{i,j,l,m}}{\Delta a},\quad \partial_a^{(B)}v^{n+1}_{i,j,l,m}=\frac{v^{n+1}_{i,j,l,m}-v^{n+1}_{i,j-1,l,m}}{\Delta a},\\ \partial_K^{(F)}v^{n+1}_{i,j,l,m}&=\frac{v^{n+1}_{i,j,l+1,m}-v^{n+1}_{i,j,l,m}}{\Delta K},\quad \partial_K^{(B)}v^{n+1}_{i,j,l,m}=\frac{v^{n+1}_{i,j,l,m}-v^{n+1}_{i,j,l-1,m}}{\Delta K},\\ \partial_Z^{(F)}v^{n+1}_{i,j,l,m}&=\frac{v^{n+1}_{i,j,l,m+1}-v^{n+1}_{i,j,l,m}}{\Delta Z},\quad \partial_Z^{(B)}v^{n+1}_{i,j,l,m}=\frac{v^{n+1}_{i,j,l,m}-v^{n+1}_{i,j,l,m-1}}{\Delta Z},\\ \partial_{ZZ}v^{n+1}_{i,j,l,m}&=\frac{v^{n+1}_{i,j,l,m+1}+v^{n+1}_{i,j,l,m-1}-2v^{n+1}_{i,j,l,m}}{(\Delta Z)^2},\end{aligned}$$

and

$$s^{n}_{i,j,l,m,F}=w_{l,m}z_i+r_{l,m}a_j-\left[\frac{1}{\partial_a^{(F)}v^{n}_{i,j,l,m}}\right]^{1/\gamma},\quad s^{n}_{i,j,l,m,B}=w_{l,m}z_i+r_{l,m}a_j-\left[\frac{1}{\partial_a^{(B)}v^{n}_{i,j,l,m}}\right]^{1/\gamma}.$$

Also, the optimal consumption is set to

$$c^{n}_{i,j,l,m}=(\partial_a v^{n}_{i,j,l,m})^{-\gamma}$$

where

$$\partial_a v^{n}_{i,j,l,m}=\partial_a^{(F)}v^{n}_{i,j,l,m}\mathbf{1}_{s^{n}_{i,j,l,m,F}>0}+\partial_a^{(B)}v^{n}_{i,j,l,m}\mathbf{1}_{s^{n}_{i,j,l,m,B}<0}+\partial_a\bar{v}^{n}_{i,j,l,m}\mathbf{1}_{s^{n}_{i,j,l,m,F}\leq 0}\mathbf{1}_{s^{n}_{i,j,l,m,B}\geq 0}.$$

In the above expression, $\partial_a\bar{v}^{n}_{i,j,l,m}=(\bar{c}^{n}_{i,j,l,m})^{-\gamma}$ where $\bar{c}^{n}_{i,j,l,m}$ is the consumption level such that

$s^n(a_i) = 0$. Equation (9) can be rearranged as

$$\begin{aligned}\frac{v^{n+1}_{i,j,l,m} - v^n_{i,j,l,m}}{\Delta} + \rho v^{n+1}_{i,j,l,m} =& \frac{(c^n_{i,j,l,m})^{1-\gamma} - 1}{1-\gamma} \\ &+ \alpha^n_{i,j,l,m} v^{n+1}_{i,j-1,l,m} + \beta^n_{i,j,l,m} v^{n+1}_{i,j,l,m} + \xi^n_{i,j,l,m} v^{n+1}_{i,j+1,l,m} + \lambda_i v^{n+1}_{-i,j,l,m} \\ &+ \alpha^n_{K,l,m} v^{n+1}_{i,j,l-1,m} + \xi^n_{K,l,m} v^{n+1}_{i,j,l+1,m} \\ &+ \alpha^n_{Z,m} v^{n+1}_{i,j,l,m+1} + \xi^n_{Z,m} v^{n+1}_{i,j,l,m-1}\end{aligned}$$

where

$$\begin{aligned}\alpha^n_{i,j,l,m} &= \frac{-s^n_{i,j,l,m,B}\mathbf{1}_{s^n_{i,j,l,m,B}<0}}{\Delta a}, \\ \beta^n_{i,j,l,m} &= \frac{s^n_{i,j,l,m,B}\mathbf{1}_{s^n_{i,j,l,m,B}<0} - s^n_{i,j,l,m,F}\mathbf{1}_{s^n_{i,j,l,m,F}>0}}{\Delta a} - \lambda_i \\ &\quad - \frac{h_{l,m}(-\mathbf{1}_{h_{l,m}<0} + \mathbf{1}_{h_{l,m}>0})}{\Delta K} - \frac{\theta(\bar{Z} - Z_m)(-\mathbf{1}_{\bar{Z}-Z_m<0} + \mathbf{1}_{\bar{Z}-Z_m>0})}{\Delta Z} - \frac{\sigma^2}{(\Delta Z)^2}, \\ \xi^n_{i,j,l,m} &= \frac{s^n_{i,j,l,m,F}\mathbf{1}_{s^n_{i,j,l,m,F}>0}}{\Delta a}, \\ \alpha^n_{K,l,m} &= -\frac{h_{l,m}\mathbf{1}_{h_{l,m}<0}}{\Delta K}, \\ \xi^n_{K,l,m} &= \frac{h_{l,m}\mathbf{1}_{h_{l,m}>0}}{\Delta K}, \\ \alpha^n_{Z,m} &= -\frac{\theta(\bar{Z} - Z_m)\mathbf{1}_{\bar{Z}-Z_m<0}}{\Delta Z} + \frac{\sigma^2}{2(\Delta Z)^2}, \\ \xi^n_{Z,m} &= \frac{\theta(\bar{Z} - Z_m)\mathbf{1}_{\bar{Z}-Z_m>0}}{\Delta Z} + \frac{\sigma^2}{2(\Delta Z)^2}.\end{aligned}$$

This equation is a system of $2 \times J \times L \times M$ linear equations and can be stacked into matrix forms as

$$\frac{1}{\Delta}(\boldsymbol{v}^{n+1} - \boldsymbol{v}^n) + \rho \boldsymbol{v}^{n+1} = \boldsymbol{u}^n + \boldsymbol{A}^n \boldsymbol{v}^{n+1} \tag{10}$$

where

$$\boldsymbol{A}^n = -\begin{bmatrix} \boldsymbol{A}_1^n & \alpha_{Z,1}^n \boldsymbol{I}_{2J\times L} & \boldsymbol{0}_{2J\times L} & \cdots & \boldsymbol{0}_{2J\times L} & \boldsymbol{0}_{2J\times L} \\ \xi_{Z,2}^n \boldsymbol{I}_{2J\times L} & \boldsymbol{A}_2^n & \alpha_{Z,2}^n \boldsymbol{I}_{2J\times L} & \cdots & \boldsymbol{0}_{2J\times L} & \boldsymbol{0}_{2J\times L} \\ \boldsymbol{0}_{2J\times L} & \xi_{Z,3}^n \boldsymbol{I}_{2J\times L} & \boldsymbol{A}_3^n & \cdots & \boldsymbol{0}_{2J\times L} & \boldsymbol{0}_{2J\times L} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & \xi_{Z,M-1}^n \boldsymbol{I}_{2J\times L} & \boldsymbol{A}_{M-1}^n & \alpha_{Z,M-1}^n \boldsymbol{I}_{2J\times L} \\ \boldsymbol{0}_{2J\times L} & \boldsymbol{0}_{2J\times L} & \cdots & \boldsymbol{0}_{2J\times L} & \xi_{Z,M}^n \boldsymbol{I}_{2J\times L} & \boldsymbol{A}_M^n \end{bmatrix},$$

$$\boldsymbol{A}_m^n = -\begin{bmatrix} \boldsymbol{A}_{1,m}^n & \alpha_{K,1,m}^n \boldsymbol{I}_{2J} & \boldsymbol{0}_{2J} & \cdots & \boldsymbol{0}_{2J} & \boldsymbol{0}_{2J} \\ \xi_{K,2,m}^n \boldsymbol{I}_{2J} & \boldsymbol{A}_{2,m}^n & \alpha_{K,2,m}^n \boldsymbol{I}_{2J} & \cdots & \boldsymbol{0}_{2J} & \boldsymbol{0}_{2J} \\ \boldsymbol{0}_{2J} & \xi_{K,3,m}^n \boldsymbol{I}_{2J} & \boldsymbol{A}_{3,m}^n & \cdots & \boldsymbol{0}_{2J} & \boldsymbol{0}_{2J} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & \xi_{K,L-1,m}^n \boldsymbol{I}_{2J} & \boldsymbol{A}_{L-1}^n & \alpha_{K,L-1,m}^n \boldsymbol{I}_{2J} \\ \boldsymbol{0}_{2J} & \boldsymbol{0}_{2J} & \cdots & \boldsymbol{0}_{2J} & \xi_{K,L,m}^n \boldsymbol{I}_{2J} & \boldsymbol{A}_L^n \end{bmatrix},$$

$$\boldsymbol{A}_{l,m}^n = -\begin{bmatrix} \beta_{1,1,l,m}^n & \xi_{1,1,l,m}^n & 0 & & & \lambda_1 & & & & \\ \alpha_{1,2,l,m}^n & \beta_{1,2,l,m}^n & \xi_{1,2,l,m}^n & 0 & & & \lambda_1 & & & \\ 0 & \alpha_{1,3,l,m}^n & \beta_{1,3,l,m}^n & \xi_{1,3,l,m}^n & 0 & & & \lambda_1 & & \\ & & \ddots & \ddots & \ddots & & & & \ddots & \\ 0 & & 0 & \alpha_{1,J,l,m}^n & \beta_{1,J,l,m}^n & 0 & & & & \lambda_1 \\ \lambda_2 & & & & 0 & \beta_{2,1,l,m}^n & \xi_{2,1,l,m}^n & 0 & & \\ & \lambda_2 & & & & \alpha_{2,2,l,m}^n & \beta_{2,2,l,m}^n & \xi_{2,2,l,m}^n & 0 & \\ & & \lambda_2 & & & 0 & \alpha_{2,3,l,m}^n & \beta_{2,3,l,m}^n & \xi_{2,3,l,m}^n & 0 \\ & & & \ddots & & & & \ddots & \ddots & \ddots \\ & & & & \lambda_2 & 0 & & 0 & \alpha_{2,J,l,m}^n & \beta_{2,J,l,m}^n \end{bmatrix},$$

and

$$\boldsymbol{u}^n = \begin{bmatrix} \boldsymbol{u}_1^n \\ \boldsymbol{u}_2^n \\ \vdots \\ \boldsymbol{u}_M^n \end{bmatrix}, \boldsymbol{v}^n = \begin{bmatrix} \boldsymbol{v}_1^n \\ \boldsymbol{v}_2^n \\ \vdots \\ \boldsymbol{v}_M^n \end{bmatrix}, \boldsymbol{u}_m^n = \begin{bmatrix} \boldsymbol{u}_{1,m}^n \\ \boldsymbol{u}_{2,m}^n \\ \vdots \\ \boldsymbol{u}_{L,m}^n \end{bmatrix}, \boldsymbol{v}_m^n = \begin{bmatrix} \boldsymbol{v}_{1,m}^n \\ \boldsymbol{v}_{2,m}^n \\ \vdots \\ \boldsymbol{v}_{L,m}^n \end{bmatrix},$$

$$\boldsymbol{u}_{l,m}^n = \begin{bmatrix} \frac{(c_{1,1,l,m}^n)^{1-\gamma}}{1-\gamma} \\ \frac{(c_{1,2,l,m}^n)^{1-\gamma}}{1-\gamma} \\ \vdots \\ \frac{(c_{1,J,l,m}^n)^{1-\gamma}}{1-\gamma} \\ \frac{(c_{2,1,l,m}^n)^{1-\gamma}}{1-\gamma} \\ \vdots \\ \frac{(c_{2,J,l,m}^n)^{1-\gamma}}{1-\gamma} \end{bmatrix}, \boldsymbol{v}_{l,m}^n = \begin{bmatrix} v_{1,1,l,m}^n \\ v_{1,2,l,m}^n \\ \vdots \\ v_{1,J,l,m}^n \\ v_{2,1,l,m}^n \\ \vdots \\ v_{2,J,l,m}^n \end{bmatrix}.$$

Given $\boldsymbol{v}^n$, the system can be solved for $\boldsymbol{v}^{n+1}$. An iterative procedure can be applied to (10) to obtain a converged $\boldsymbol{v} = \boldsymbol{v}^{n^*}$ where n^* is such that $\left\|\boldsymbol{v}^{n^*+1} - \boldsymbol{v}^{n^*}\right\|$ is below a very small number.

We also define the transition matrix without the derivatives with regard to K and Z

$$\tilde{\boldsymbol{A}}_{l,m}^n = -\begin{bmatrix} \tilde{\beta}_{1,1,l,m}^n & \xi_{1,1,l,m}^n & 0 & & & \lambda_1 & & & & & \\ \alpha_{1,2,l,m}^n & \tilde{\beta}_{1,2,l,m}^n & \xi_{1,2,l,m}^n & 0 & & & \lambda_1 & & & & \\ 0 & \alpha_{1,3,l,m}^n & \tilde{\beta}_{1,3,l,m}^n & \xi_{1,3,l,m}^n & 0 & & & \lambda_1 & & & \\ & & \ddots & \ddots & \ddots & & & & \ddots & & \\ 0 & & 0 & \alpha_{1,J,l,m}^n & \tilde{\beta}_{1,J,l,m}^n & 0 & & & & & \lambda_1 \\ \lambda_2 & & & & 0 & \tilde{\beta}_{2,1,l,m}^n & \xi_{2,1,l,m}^n & 0 & & & \\ & \lambda_2 & & & & \alpha_{2,2,l,m}^n & \tilde{\beta}_{2,2,l,m}^n & \xi_{2,2,l,m}^n & 0 & & \\ & & \lambda_2 & & & 0 & \alpha_{2,3,l,m}^n & \tilde{\beta}_{2,3,l,m}^n & \xi_{2,3,l,m}^n & 0 & \\ & & & \ddots & & & & \ddots & \ddots & \ddots & \\ & & & & \lambda_2 & 0 & & 0 & \alpha_{2,J,l,m}^n & \tilde{\beta}_{2,J,l,m}^n \end{bmatrix},$$

where

$$\tilde{\beta}^n_{i,j,l,m} = \frac{s^n_{i,j,l,m,B}\mathbf{1}_{s^n_{i,j,l,m,B}<0} - s^n_{i,j,l,m,F}\mathbf{1}_{s^n_{i,j,l,m,F}>0}}{\Delta a} - \lambda_i,$$

which is used when solving the Kolmogorov Forward equation for simulation.

D Solving the Kolmogorov Forward equation for simulation

In the outer loop in the KS algorithm and nonlinear simulation in all the algorithms to obtain the sequence of $\{K^*_t\}$, we solve the Kolmogorov Forward (KF) equation with an infinite difference scheme. In the XPA and KS algorithms, we use a converged $\tilde{\boldsymbol{A}}_{l,m} = \tilde{\boldsymbol{A}}^{n^*}_{l,m}$ at each grid point of (K_l, Z_m) in the inner loop above for the transition matrix of the wealth–productivity density. Note that we exclude the direct effect of aggregate variables K and Z (i.e., the derivatives with regard to K and Z) on the transition matrix for simulation.

The law of motion of the wealth–productivity density is expressed as a form of the KF equation

$$\frac{g_{t+1,i,j} - g_{t,i,j}}{\Delta t} = -\partial_a s_t(a_j, z_i) g_{t+1,i,j} - \lambda_i g_{t+1,i,j} + \lambda_i g_{t+1,-i,j} \tag{11}$$

This equation can be expressed as a matrix form

$$\boldsymbol{g}_{t+1} - \boldsymbol{g}_t = \Delta t \boldsymbol{\Gamma}_t \boldsymbol{g}_{t+1} \tag{12}$$

where $\boldsymbol{g}_t = [g_{t,1,1}, g_{t,1,2}, ..., g_{t,1,J}, g_{t,2,1}, ..., g_{t,2,J}]^T$ is a vector of the wealth–productivity density at grid points of (a_j, z_i) and $\boldsymbol{\Gamma}_t$ is a transition matrix.

In the XPA and KS algorithms, we have already obtained a converged matrix in the inner loop for $\boldsymbol{\Gamma}_t$.[20] As (K_t, Z_t) in simulation may not be on grid points of (K_l, Z_m) used in the inner loop, we use bilinear interpolation of matrices at the nearest four grid points. That is, $\boldsymbol{\Gamma}_t = \tilde{\boldsymbol{A}}^T_t$ is the

[20] See FVHN for more details.

transpose of

$$\begin{aligned}\tilde{\boldsymbol{A}}_t =&(1-\omega_{Z,t})((1-w_{K,t})\tilde{\boldsymbol{A}}_{l^*,m^*}+w_{K,t}\tilde{\boldsymbol{A}}_{l^*+1,m^*})\\&+w_{Z,t}((1-w_{K,t})\tilde{\boldsymbol{A}}_{l^*,m^*+1}+w_{K,t}\tilde{\boldsymbol{A}}_{l^*+1,m^*+1})\end{aligned}$$

where $w_{K,t}=(K_t-K_{l^*})/(K_{l^*+1}-K_{l^*})$ and $w_{Z,t}=(Z_t-Z_{m^*})/(Z_{m^*+1}-Z_{m^*})$ and each of l^*, m^* satisfies $K_t\in[K_{l^*},K_{l^*+1}]$ or $Z_t\in[Z_{m^*},Z_{m^*+1}]$ at each period t.

Difference from FVHN FVHN solve the HJB equation with the upwind scheme only for a in the KS algorithm. We find that using the upwind scheme not only for a but also for K and Z is necessary for the XPA algorithm. We find that it is also important to improve the KS algorithm in terms of the R–squared and the Den Haan errors as shown in Table 2.

When FVHN solve the KF equation in a form of (12) in the KS algorithm, they use a converged $\boldsymbol{A}_{l,m}=\boldsymbol{A}^{n^*}_{l,m}$ (without tilde) including the terms of derivatives with K and Z as the transition matrix of the wealth–productivity density. We exclude these terms as it is more consistent with the nonlinear simulation for the REITER algorithm as explained in Appendix E. However, compared with the upwind scheme mentioned above, we find that this has only a marginal effect.

Table 2: Den Haan errors: alternative methods for the KS algorithm

a. Our approach

Agg Shock σ (%)	0.01	0.1	0.7	1.0	3.0	5.0
$\varepsilon^{MAX}_{DH\ KS}$ (%)	0.002	0.015	0.097	0.134	0.493	0.885
$\varepsilon^{MEAN}_{DH\ KS}$ (%)	0.001	0.011	0.075	0.104	0.286	0.438
R^2	0.999	0.999	0.999	0.999	0.999	0.998

b. Alternative approach (as in FVHN)

Agg Shock σ (%)	0.01	0.1	0.7	1.0	3.0	5.0
$\varepsilon^{MAX}_{DH\ KS}$ (%)	0.005	0.047	0.322	0.456	1.349	2.463
$\varepsilon^{MEAN}_{DH\ KS}$ (%)	0.001	0.012	0.084	0.120	0.364	0.627
R^2	0.994	0.994	0.994	0.993	0.991	0.988

E Den Haan errors for the REITER algorithm

As in Ahn et al. (2018); Bayer and Luetticke (2020), to obtain the sequence of $\{K_t^*\}$ in the REITER algorithm, we solve for the optimal savings plan $s_t^*(a_j, z_i)$ at every pair of the wealth and productivity values (a_j, z_i) in the histogram under the equilibrium factor prices (w_t, r_t) in every period t.[21] To solve for the policy function of savings, we utilize the value function obtained in the solution with linearization.[22] Then we calculate the transition matrix for the law of motion of the wealth–productivity density (i.e., $\mathbf{\Gamma}_t$ in (12)) from the policy function.

First, we calculate the optimal savings plan $s_{t,i,j}^* = s_t^*(a_j, z_i)$ at each grid point in period t for the KF equation (11). Having the current aggregate state $\{g_{t,i,j}\}$ and Z_t, we have

$$\begin{aligned} K_t &= \sum_i \sum_j a_j g_{t,i,j}, \\ w_t &= (1-\alpha) e^{Z_t} (K_t / L_t)^{\alpha}, \\ r_t &= \alpha e^{Z_t} (K_t / L_t)^{\alpha - 1} - \delta. \end{aligned}$$

From the household's budget constraint, we have savings as $s_{t,i,j} = s_t(a_j, z_i) = w_t z_i + r_t a_j - c_{t,i,j}$. Having the value function from the linearized solution at each grid point at date t, denoted by $\{\tilde{v}_{t,i,j}\}$, we also have *the optimal value of consumption at each* (a_j, z_i):

$$c_{t,i,j}^* = c_t^*(a_j, z_i) = (\partial_a \tilde{v}_{t,i,j})^{-\gamma}. \tag{13}$$

We use the upward scheme to approximate the derivative of the value function as well. Therefore,

$$\begin{aligned} s_{t,i,j,F}^* &= w_t z_i + r_t a_j - \left[\frac{1}{\partial_a^{(F)} \tilde{v}_{t,i,j}} \right]^{1/\gamma}, \\ s_{t,i,j,B}^* &= w_t z_i + r_t a_j - \left[\frac{1}{\partial_a^{(B)} \tilde{v}_{t,i,j}} \right]^{1/\gamma}, \end{aligned}$$

[21] In general, we need to solve for the equilibrium prices that clear markets. The market clearing condition for capital is met in the first place in our case with the standard Krusell-Smith model.

[22] Bayer and Luetticke (2020) solve the optimization problem by utilizing the expected value function that is obtained in the solution with linearization.

where

$$\partial_a^{(F)}\tilde{v}_{t,i,j} = \frac{\tilde{v}_{t,i,j+1} - \tilde{v}_{t,i,j}}{\Delta a},$$
$$\partial_a^{(B)}\tilde{v}_{t,i,j} = \frac{\tilde{v}_{t,i,j} - \tilde{v}_{t,i,j-1}}{\Delta a},$$

Then, Equation (11) can be rearranged as

$$\frac{g_{t+1,i,j} - g_{t,i,j}}{\Delta t} = \alpha_{t,i,j} g_{t+1,i,j+1} + \beta_{t,i,j} g_{t+1,i,j} + \xi_{t,i,j} g_{t+1,i,j-1} + \lambda_{-i} g_{t+1,-i,j}$$

where

$$\alpha_{t,i,j} = \frac{-s^*_{t,i,j,B}\mathbf{1}_{s^*_{t,i,j,B}<0}}{\Delta a},$$
$$\beta_{t,i,j} = \frac{s^*_{t,i,j,B}\mathbf{1}_{s^*_{t,i,j,B}<0} - s^*_{t,i,j,F}\mathbf{1}_{s^*_{t,i,j,F}>0}}{\Delta a} - \lambda_i,$$
$$\xi_{t,i,j} = \frac{s^*_{t,i,j,F}\mathbf{1}_{s^*_{t,i,j,F}>0}}{\Delta a}.$$

This equation can be stacked into a matrix form as Equation (12) such that

$$\mathbf{\Gamma}_t = \begin{bmatrix}
\beta_{t,1,1} & \xi_{t,1,1} & 0 & & & \lambda_1 & & & & \\
\alpha_{t,1,2} & \beta_{t,1,2} & \xi_{t,1,2} & 0 & & & \lambda_1 & & & \\
0 & \alpha_{t,1,3} & \beta_{t,1,3} & \xi_{t,1,3} & 0 & & & \lambda_1 & & \\
 & & \ddots & \ddots & \ddots & & & & \ddots & \\
0 & & 0 & \alpha_{t,1,J} & \beta_{t,1,J} & 0 & & & & \lambda_1 \\
\lambda_2 & & & & 0 & \beta_{t,2,1} & \xi_{t,2,1} & 0 & & \\
 & \lambda_2 & & & & \alpha_{t,2,2} & \beta_{t,2,2} & \xi_{t,2,2} & 0 & \\
 & & \lambda_2 & & & 0 & \alpha_{t,2,3} & \beta_{t,2,3} & \xi_{t,2,3} & 0 \\
 & & & \ddots & & & & \ddots & \ddots & \ddots \\
 & & & & \lambda_2 & 0 & & 0 & \alpha_{t,2,J} & \beta_{t,2,J}
\end{bmatrix}^T.$$

F Further numerical results

Benchmark parameters

We use the parameters in Table 3 in the benchmark case, following those used in Ahn et al. (2018). Later, we change the volatility and persistence parameters of the TFP to examine its effect on the accuracy of solving the model between different algorithms. We set the range of grid points for K as $[0.8\bar{K}, 1.2\bar{K}]$ where $\bar{K}$ is the value of aggregate capital in the stationary distribution. The range of grid points for Z is $[-\bar{m}\sigma, \bar{m}\sigma]$ where $\bar{m}$ is a real number. We set $\bar{m} = 4$. We divide the state space of (K_l, Z_m) by 3 grid points for each.

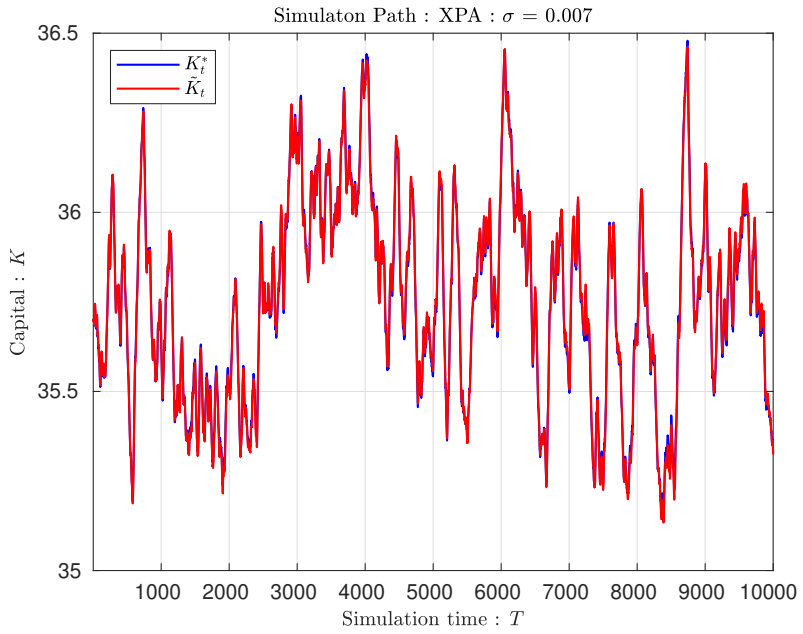
Table 3: Parameter values

Parameters	Benchmark Value
γ : Relative Risk Aversion	1.0
ρ : Rate of Time Preference	0.01
α : Capital Share	0.36
δ : Rate of Capital Depreciation	0.025
τ : Tax Rate of Labor Income	0.011
b : Rate of Compensation	0.15
$1-\mu$: Persistence of TFP	0.75
σ : Volatility of TFP	0.07
z : Idiosyncratic of Labor Productivity	$z_u = 0, z_e = 1$
λ : Probability of Labor Productivity	$\lambda_e = 0.50, \lambda_u = 0.03$

In Figure 4, we show the results of the simulation path in XPA. The red line $\{\tilde{K}_t\}_{t\in[0,T]}$ in the figure shows the simulation obtained only from the forecasting rule for 10,000 periods, while the blue line $\{K_t^*\}_{t\in[0,T]}$ shows the simulation obtained from the nonlinear model including the household HJB equation (with the forecasting rule) and the Kolmogorov Forward equation for 10,000 periods. This is known as the Den haan's (2010) fundamental plot showing the accuracy of the solution. It is clear from the figure that the capital paths resulting from these simulations are very close.[23]

[23] If the red and blue lines are close, households approximately act according to the correct forecasting rule. If not, households are using the wrong forecasting rule. Therefore, the divergence between the two lines indicates that the model is not solved correctly based on rational expectations.

Figure 4: Simulation path in XPA (Den haan's fundamental plot)



We also check the robustness of our results with respect to the persistence of TFP. That is, the persistence of TFP, $1-\mu$, is lowered from $\mu = 0.25$ to $\mu = 0.5$ or $\mu = 0.75$. In Table 4, we compare the mean of the simulation path derived from the nonlinear model for 10,000 periods in XPA (compared with that in KS) and the mean of the simulation path derived from the linearized model in REITER (compared with that in KS) for each μ. We see that the gap between XPA and KS is much smaller than the gap between REITER and KS for alternative values of μ as well.

In Table 5, we show that the Den haan errors of KS, XPA, and REITER for each μ. We use the sequences of $\{K_t^*\}_{t\in[0,T]}$ and $\{\tilde{K}_t\}_{t\in[0,T]}$ to measure the Den haan errors

$$\varepsilon_{DH}^{MAX} \equiv 100 \cdot \max_{t\in[0,T]} |\ln \tilde{K}_t - \ln K_t^*|,$$
$$\varepsilon_{DH}^{MEAN} \equiv 100 \cdot \frac{\sum_{t\in[0,T]} |\ln \tilde{K}_t - \ln K_t^*|}{T}.$$

It is clear that, regardless of the value of μ, the Den haan errors of XPA are smallest compared to those of REITER and KS when σ is large.

Table 4: Simulation of capital paths

a. Case of $\mu = 0.25$ (benchmark)

Agg Shock σ (%)	0.01	0.1	0.7	1.0	3.0	5.0
XPA–KS (%)	0.001	0.013	0.097	0.160	0.529	1.079
REITER–KS (%)	0.002	0.022	0.062	0.084	2.832	6.688

a. Case of $\mu = 0.50$

Agg Shock σ (%)	0.01	0.1	0.7	1.0	3.0	5.0
XPA–KS (%)	0.001	0.007	0.055	0.098	0.341	0.711
REITER–KS (%)	0.002	0.015	0.056	0.051	1.524	4.127

b. Case of $\mu = 0.75$

Agg Shock σ (%)	0.01	0.1	0.7	1.0	3.0	5.0
XPA–KS (%)	0.000	0.005	0.039	0.074	0.270	0.571
REITER–KS (%)	0.001	0.011	0.047	0.053	0.922	2.908

Table 5: Den Haan errors: robustness

a. Case of $\mu = 0.25$ (benchmark)

Agg Shock σ (%)	0.01	0.1	0.7	1.0	3.0	5.0
$\varepsilon^{MAX}_{DH\ XPA}$ (%)	0.001	0.011	0.084	0.125	0.336	0.580
$\varepsilon^{MEAN}_{DH\ XPA}$ (%)	0.000	0.004	0.028	0.038	0.091	0.140
$\varepsilon^{MAX}_{DH\ KS}$ (%)	0.002	0.015	0.097	0.134	0.493	0.885
$\varepsilon^{MEAN}_{DH\ KS}$ (%)	0.001	0.011	0.075	0.104	0.286	0.438
$\varepsilon^{MAX}_{DH\ REITER}$ (%)	0.000	0.003	0.144	0.387	4.379	9.759
$\varepsilon^{MEAN}_{DH\ REITER}$ (%)	0.000	0.002	0.106	0.273	3.381	7.546

a. Case of $\mu = 0.50$

Agg Shock σ (%)	0.01	0.1	0.7	1.0	3.0	5.0
$\varepsilon^{MAX}_{DH\ XPA}$ (%)	0.001	0.008	0.048	0.062	0.157	0.308
$\varepsilon^{MEAN}_{DH\ XPA}$ (%)	0.000	0.003	0.019	0.024	0.053	0.083
$\varepsilon^{MAX}_{DH\ KS}$ (%)	0.001	0.009	0.059	0.083	0.278	0.483
$\varepsilon^{MEAN}_{DH\ KS}$ (%)	0.001	0.007	0.044	0.060	0.155	0.235
$\varepsilon^{MAX}_{DH\ REITER}$ (%)	0.000	0.001	0.069	0.155	2.431	5.961
$\varepsilon^{MEAN}_{DH\ REITER}$ (%)	0.000	0.001	0.057	0.126	1.989	4.864

b. Case of $\mu = 0.75$

Agg Shock σ (%)	0.01	0.1	0.7	1.0	3.0	5.0
$\varepsilon^{MAX}_{DH\ XPA}$ (%)	0.001	0.006	0.040	0.052	0.110	0.185
$\varepsilon^{MEAN}_{DH\ XPA}$ (%)	0.000	0.003	0.017	0.022	0.041	0.055
$\varepsilon^{MAX}_{DH\ KS}$ (%)	0.001	0.006	0.042	0.059	0.200	0.350
$\varepsilon^{MEAN}_{DH\ KS}$ (%)	0.001	0.005	0.031	0.043	0.107	0.163
$\varepsilon^{MAX}_{DH\ REITER}$ (%)	0.000	0.001	0.046	0.097	1.593	4.263
$\varepsilon^{MEAN}_{DH\ REITER}$ (%)	0.000	0.001	0.039	0.081	1.312	3.537