Online Appendix

A Additional Comparison Results

A.1 Detailed Forecast Accuracy Statistics: Khan and Thomas (2008) Model

This section provides detailed forecast accuracy statistics for the forecasting rules used in the Khan and Thomas (2008) model application, broken down by aggregate productivity state Z. Table 9 compares the benchmark KS method with the NN-KS method.

Table 9: Shock-specific forecast accuracy

	Price p		Capital K'	
Statistic	KS	NN-KS	KS	NN-KS
Den Haan Statistics (%)				
Maximum	0.11	0.39	0.38	0.90
Mean	0.05	0.06	0.23	0.15
Root Mean Squared Error (RMSE) (%)				
$Z = Z_1$	0.06	0.08	0.06	0.06
$Z = Z_2$	0.05	0.07	0.05	0.07
$Z = Z_3$	0.05	0.06	0.05	0.07
$Z = Z_4$	0.04	0.03	0.05	0.04
$Z = Z_5$	0.04	0.07	0.05	0.06
Forecast Regression R^2				
$Z = Z_1$	1.0000	0.9969	1.0000	0.9994
$Z = Z_2$	1.0000	0.9983	1.0000	0.9993
$Z = Z_3$	1.0000	0.9986	1.0000	0.9996
$Z = Z_4$	1.0000	0.9996	1.0000	0.9999
$Z = Z_5$	1.0000	0.9981	1.0000	0.9996

Note: The table reports forecasting accuracy statistics conditional on aggregate shock Z. All values are expressed as percentage points of log deviations (original values multiplied by 100).

A.2 Detailed Forecast Accuracy Statistics: Bloom et al. (2018) Model

This section provides detailed forecast accuracy statistics for the Bloom et al. (2018) model application, broken down by the aggregate state (Z, S, S_{-1}), representing discretized grid points for aggregate productivity, current uncertainty, and lagged uncertainty. Table 10 compares the benchmark Krusell-Smith (KS) method with the NN-KS method. RMSE values are multiplied by 100.

Table 10: Shock-specific forecast accuracy

Aggregate State	Price p				Capital K			
(A, S, S_{-1})	KS		NN-KS		KS		NN-KS	
	RMSE (%)	R^2	RMSE (%)	R^2	RMSE (%)	R^2	RMSE (%)	R^2
Den Haan Statistics								
Maximum	3.52		3.62		5.89		6.47	
Mean	0.87		0.88		1.75		1.88	
(1,0,0)	0.61	0.83	0.73	0.75	0.35	0.98	0.31	0.99
(1,0,1)	0.16	0.97	0.06	1.00	0.03	1.00	0.08	1.00
(1,1,0)	0.30	0.94	0.29	0.94	0.47	0.95	0.57	0.95
(1,1,1)	0.31	0.92	0.37	0.90	0.12	1.00	0.13	1.00
(2,0,0)	0.55	0.85	0.52	0.88	0.34	0.98	0.34	0.99
(2,0,1)	0.16	0.99	0.06	1.00	0.08	1.00	0.10	1.00
(2,1,0)	0.39	0.89	0.40	0.89	0.25	0.98	0.27	0.99
(2,1,1)	0.35	0.94	0.37	0.94	0.11	1.00	0.12	1.00
(3,0,0)	0.45	0.92	0.46	0.92	0.30	0.99	0.28	0.99
(3,0,1)	0.18	0.99	0.16	0.99	0.09	1.00	0.10	1.00
(3,1,0)	0.21	0.98	0.20	0.99	0.44	0.98	0.39	0.99
(3,1,1)	0.33	0.96	0.42	0.94	0.11	1.00	0.12	1.00
(4,0,0)	0.53	0.87	0.53	0.89	0.34	0.99	0.31	0.99
(4,0,1)	0.21	0.98	0.20	0.98	0.08	1.00	0.10	1.00
(4,1,0)	0.24	0.97	0.21	0.98	0.44	0.97	0.39	0.98
(4,1,1)	0.36	0.93	0.45	0.91	0.12	1.00	0.12	1.00
(5,0,0)	0.57	0.86	0.59	0.85	0.33	0.99	0.31	0.99
(5,0,1)	0.21	0.97	0.33	0.92	0.17	1.00	0.18	1.00
(5,1,0)	0.31	0.96	0.28	0.97	0.11	1.00	0.09	1.00
(5,1,1)	0.32	0.94	0.39	0.92	0.11	1.00	0.11	1.00

Note: Table reports the forecasting accuracy statistics conditional on (Z_t, S_t, S_{t-1}). All values are expressed as percentage points of log deviations (original values multiplied by 100).

A.3 Microeconomic Investment-Rate Moments in Khan and Thomas (2008) model.

Table 11: Microeconomic investment-rate moments

Method	$\frac{i}{k}$	$\sigma\left(\frac{i}{k}\right)$	$\mathbb{P}\left(\frac{i}{k}=0\right)$	$\mathbb{P}\left(\frac{i}{k}\geq 0.2\right)$	$\mathbb{P}\left(\frac{i}{k}\leq -0.2\right)$	$\mathbb{P}\left(\frac{i}{k}>0\right)$	$\mathbb{P}\left(\frac{i}{k}<0\right)$
KS	0.0947	0.2597	0.7693	0.1724	0.0280	0.1890	0.0417
NN-KS	0.0997	0.2837	0.7298	0.1790	0.0428	0.2107	0.0596
(run s.d.)	0.0008	0.0046	0.0037	0.0021	0.0006	0.0036	0.0005

Note: For each method, entries report the mean value, across periods, of the indicated microeconomic moment of the cross-sectional distribution of investment rates i/k. For NN-KS, "mean" is the average across 5 independent runs, and "std" is the standard deviation across these runs. All statistics are computed from a 2,500-period unconditional simulation of the model, after discarding the first 500 periods as burn-in.

Table 11 summarizes microeconomic moments of the cross-sectional investment-rate distribution i/k under the benchmark KS method and the NN-KS method. For NN-KS, I report the mean across 5 independent runs, together with the standard deviation across runs. Overall, the NN-KS method generates moments that are systematically shifted relative to the KS benchmark, although they remain stable across runs (small standard deviations). Given that the mean Bellman error is lower under NN-KS than under the KS baseline, it is more plausible to interpret these differences as reflecting differences in how accurately the solution is computed, rather than solution inaccuracy per se, supporting the view that the NN-KS outcomes provide a valid characterization of the model's micro-level dynamics.

B Policy-function analysis

B.1 Accuracy of the discrete policy function

In this section, I compare the policy functions generated by the NN-KS method with those obtained from the original code of Bloom et al. (2018). I begin by eval-

uating the accuracy of the discrete adjustment decision. The evaluation is conducted on 3,367,000 grid points spanning the state variables $(z, \epsilon, k, n_{-1}, \sigma, \sigma_{-1}, K)$, with grid sizes $(5, 5, 91, 37, 2, 2, 10)$. Both methods are evaluated on the same grid points. For the forecasting rules, I use the coefficients obtained from the converged solution of the original code in Bloom et al. (2018).

To clarify how I measure "classification accuracy," note that the firm makes two binary adjustment decisions as in (47)–(50): whether to adjust capital (pay the fixed cost) and whether to adjust labor. Therefore, the joint discrete policy has four mutually exclusive alternatives,

$$\mathcal{D} \equiv \{(0,0), (1,0), (0,1), (1,1)\}, \tag{51}$$

where the first (second) element indicates the capital (labor) adjustment decision. For each grid point $s \equiv (z, \epsilon, k, n_{-1}, \sigma, \sigma_{-1}, K)$, the KS method implies a predicted class $d^{KS}(s) \in \mathcal{D}$ through the argmax over the corresponding value functions in (47)–(50). In contrast, NN-KS directly predicts a discrete class $d^{NN}(s) \in \mathcal{D}$ using the trained discrete policy network. A state is counted as correctly classified if $d^{NN}(s) = d^{KS}(s)$.

I then train three neural networks—corresponding to the value function, the discrete policy function, and the conditional policy function—for 300 iterations—while holding the forecasting rules fixed and omitting the price error term.[27]

Table 12 reports the classification performance of the discrete adjustment decision. The overall accuracy is 94.16 % (3,170,423 correct out of 3,367,000), implying a mismatch rate of 5.83%. The key point is that discrepancies in the continuation value between the correct choice and the wrong choice selected by the neural network are quantitatively small. Table 13 reports the magnitude of deviations conditional on the remaining mismatches in the discrete choices, focusing on capital

[27]Here, I apply a lenient criterion only to cases in which the KS method selects no investment while the NN selects positive investment. I treat the NN's choice as correct if the implied investment level lies within an adjacent grid point. This is motivated by the discrete nature of the choice problem: the KS method can select no investment even when the true optimal choice lies between adjacent grid points. This criterion applies to 13,603 evaluation points.

Table 12: Discrete-choice classification accuracy

Statistic	Count	Share (%)
Total evaluation points	3,367,000	100.00
Correctly classified	3,170,423	94.16
Misclassified	196,577	5.83

Table 13: Magnitude of deviations among remaining mismatches

Metric	Median $\|\Delta\|$	95th pct. $\|\Delta\|$	99th pct. $\|\Delta\|$	$\Pr(\|\Delta\| > 0.5)$
I deviation ($\|\Delta I\|$)	0.214	0.932	1.645	0.191
S deviation ($\|\Delta S\|$)	0.019	0.269	0.471	0.008
RHS deviation ($\|\Delta\text{RHS}\|$)	0.0024	0.0273	0.0885	0.000004

investment, human capital investment, and the right-hand side (RHS) of the Bellman equation. [28] Even when discrete choices do not coincide, the median RHS deviation remains small: the difference between the RHS evaluated at the KS-selected action and that implied by the NN–KS method is only 0.0024. While the RHS level in this implementation typically ranges from 20 to 45, the median absolute RHS discrepancy is only 0.0024, with the 95th and 99th percentiles equal to 0.0273 and 0.0885, respectively. These statistics indicate that, even when discrete misclassification occurs, the neural network tends to select an alternative whose continuation value is extremely close to that of the KS-selected choice. This provides further evidence that the discrete policy function is sufficiently well trained from an economic perspective.

B.2 Plots of policy function

Figures 8–13 illustrate the optimal decision rules for capital investment, $I(k)$, and labor adjustment, $S(n)$, across the state space. These are the results of the

[28] Since the discrete policy function selects the action that maximizes the continuation value among (47)–(50), the RHS deviation corresponds to the difference in the maximized continuation values implied by each method.

same neural networks trained in the B.1. Each figure consists of a 5×5 panel over the joint grid of aggregate and idiosyncratic productivity levels. Rows correspond to the five grid values of aggregate productivity z, and columns correspond to the five grid values of idiosyncratic productivity ε. Throughout, the stochastic volatility shocks are fixed at the low state ($\sigma = 0$ and $\sigma_{-1} = 0$), and aggregate capital K is held fixed.

Capital investment. Figures 8–10 plot the capital-investment policy $I(k)$ as a function of current capital k. Within each (z, ε) panel, I fix lagged labor n_{-1} at a given value and vary k along the horizontal axis. The three figures correspond to three different fixed values of n_{-1}. In these figures, the vertical-axis value 0 corresponds to inaction (i.e., no capital adjustment).

Labor adjustment. Figures 11–13 plot the labor-adjustment policy $S(n)$ as a function of lagged labor n_{-1}. Within each (z, ε) panel, I fix capital k at a given value and vary n_{-1} along the horizontal axis. The three figures correspond to three different fixed values of k. In these figures, the vertical-axis value 0 corresponds to inaction (i.e., no labor adjustment).

In all figures, the red line shows the KS method and the blue line shows the NN-KS method.

Figure 8: Capital investment $I(k)$ at a fixed $n_{-1} = 0.2673$

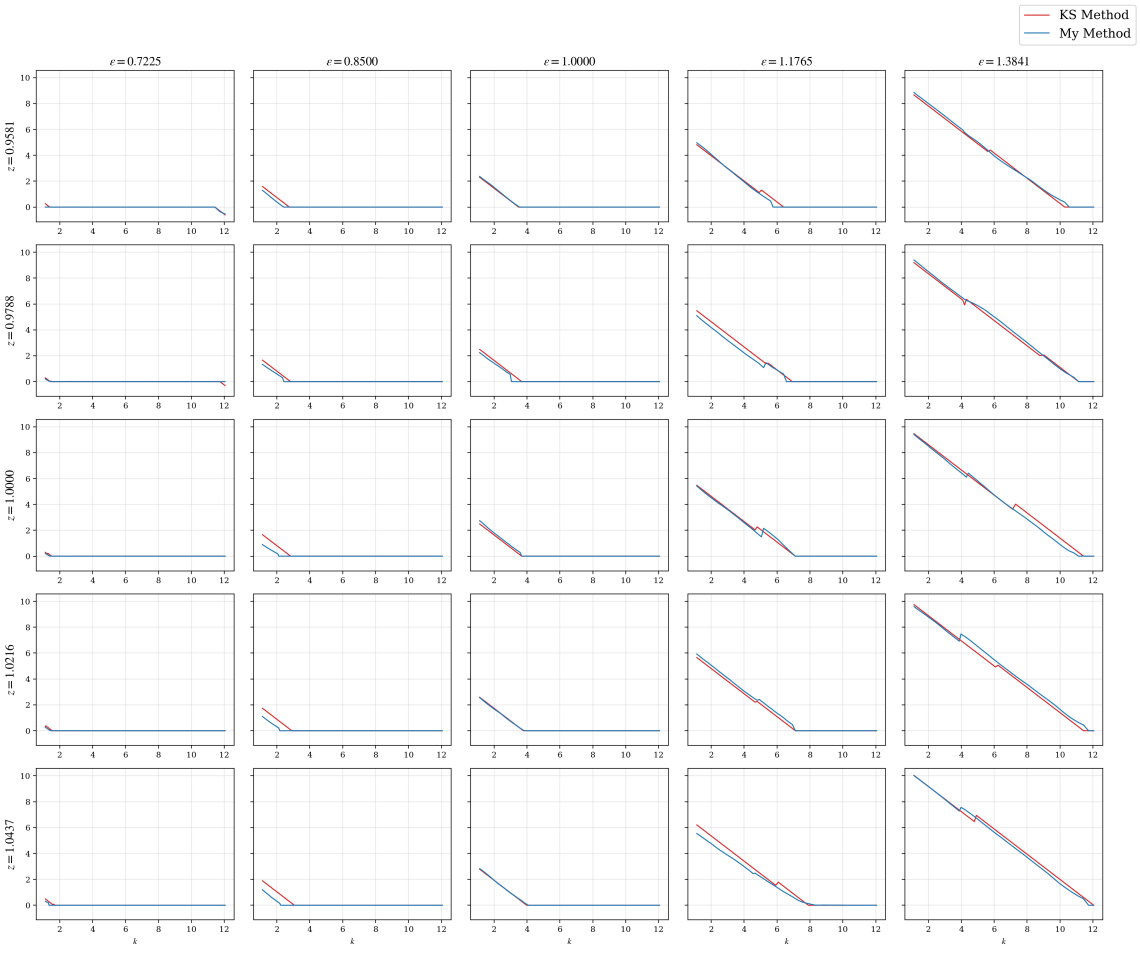


Figure 9: Capital investment $I(k)$ at a fixed $n_{-1} = 0.4997$

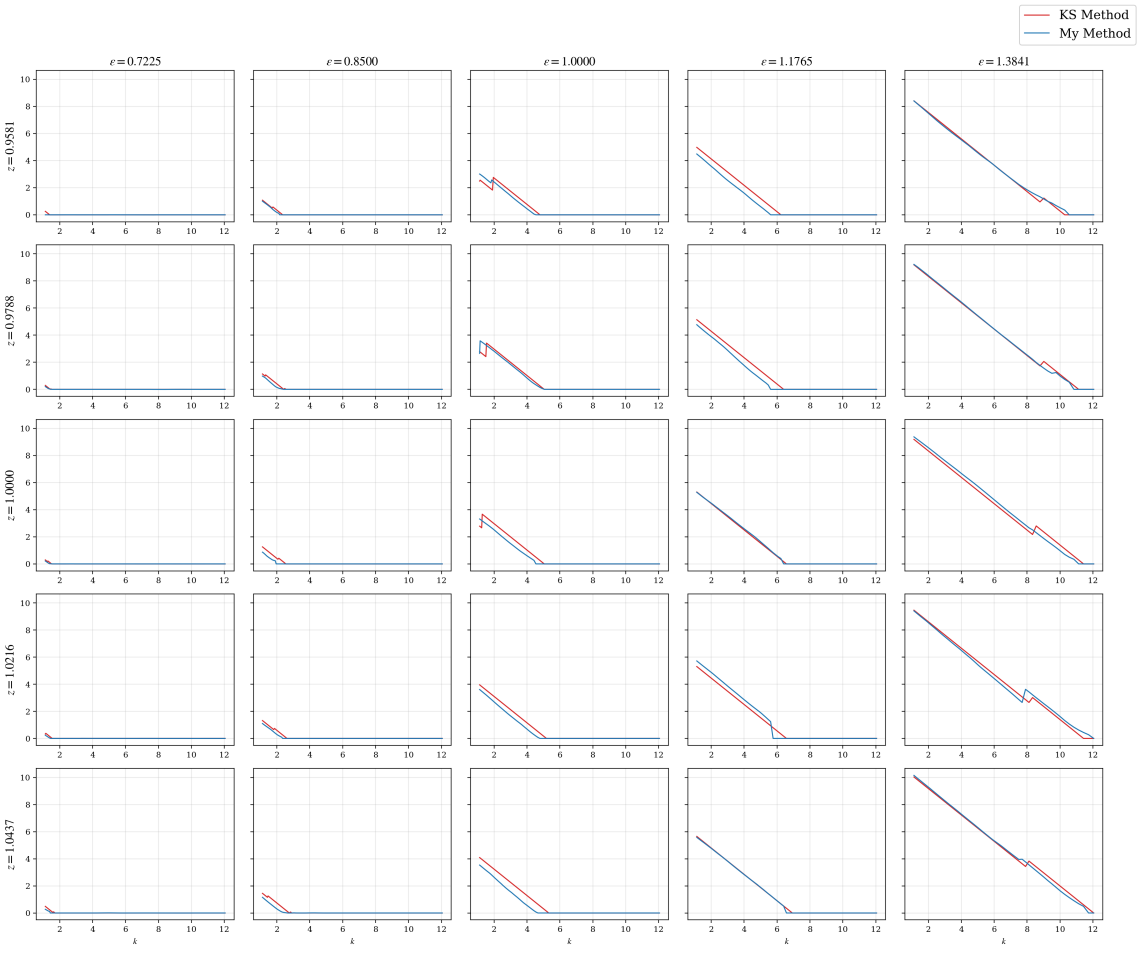


Figure 10: Capital investment $I(k)$ at a fixed $n_{-1} = 0.7320$

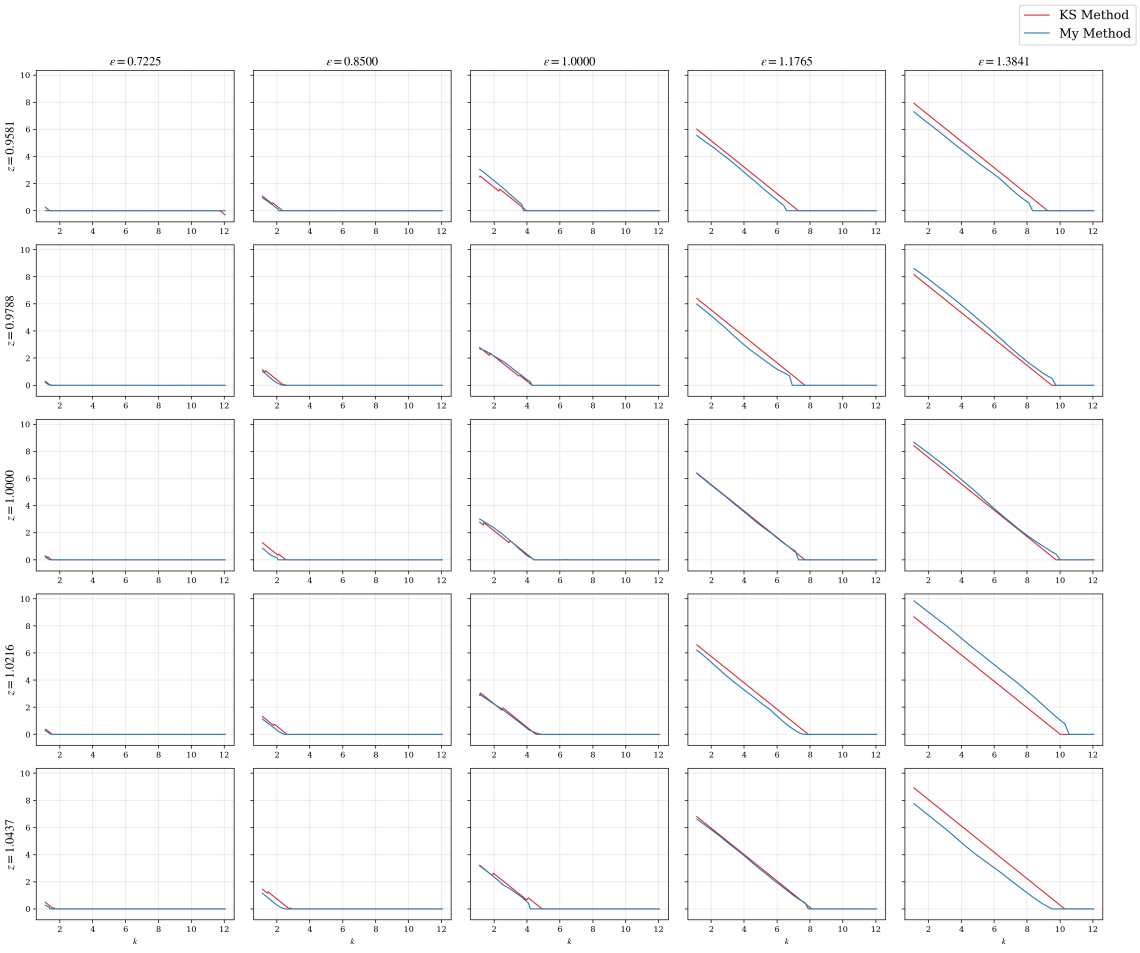


Figure 11: Labor adjustment $S(n)$ at a fixed $k = 3.8553$

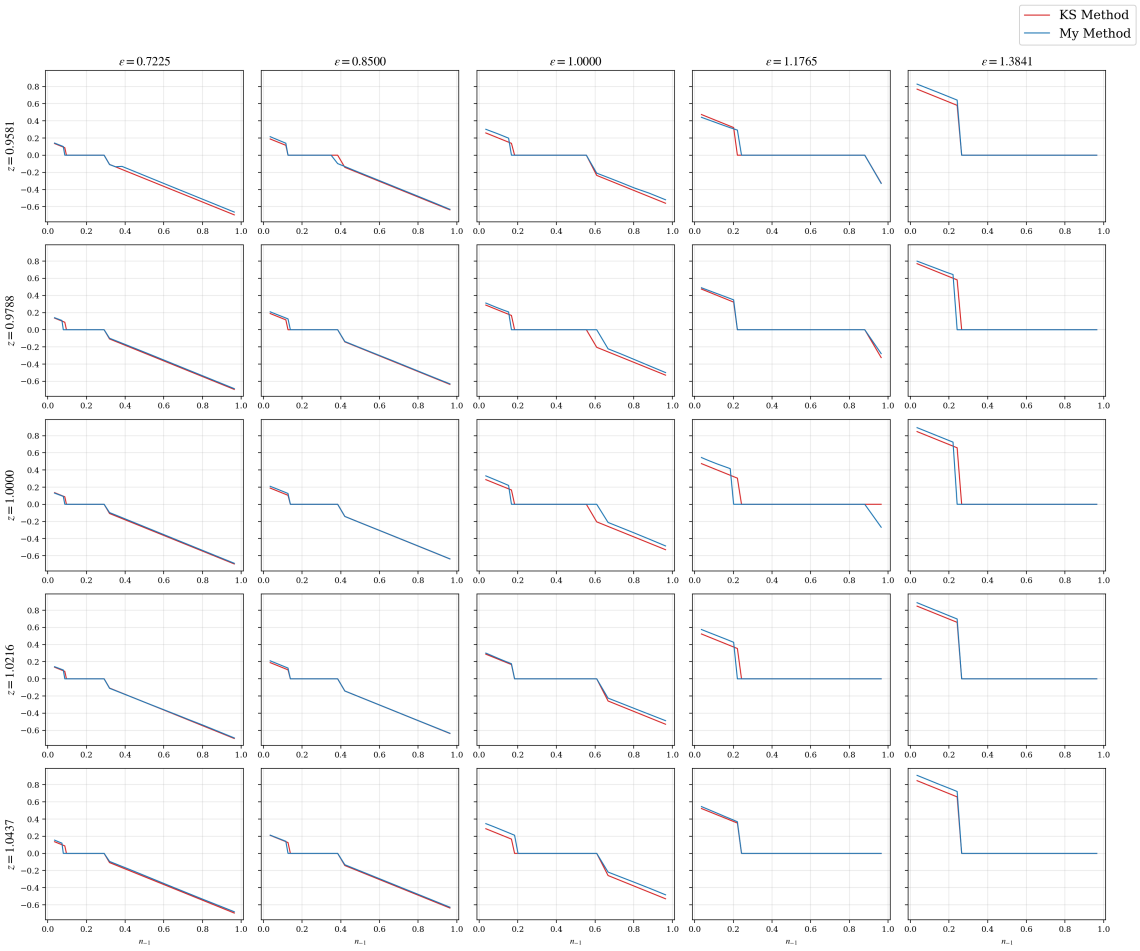


Figure 12: Labor adjustment $S(n)$ at a fixed $k = 6.5855$

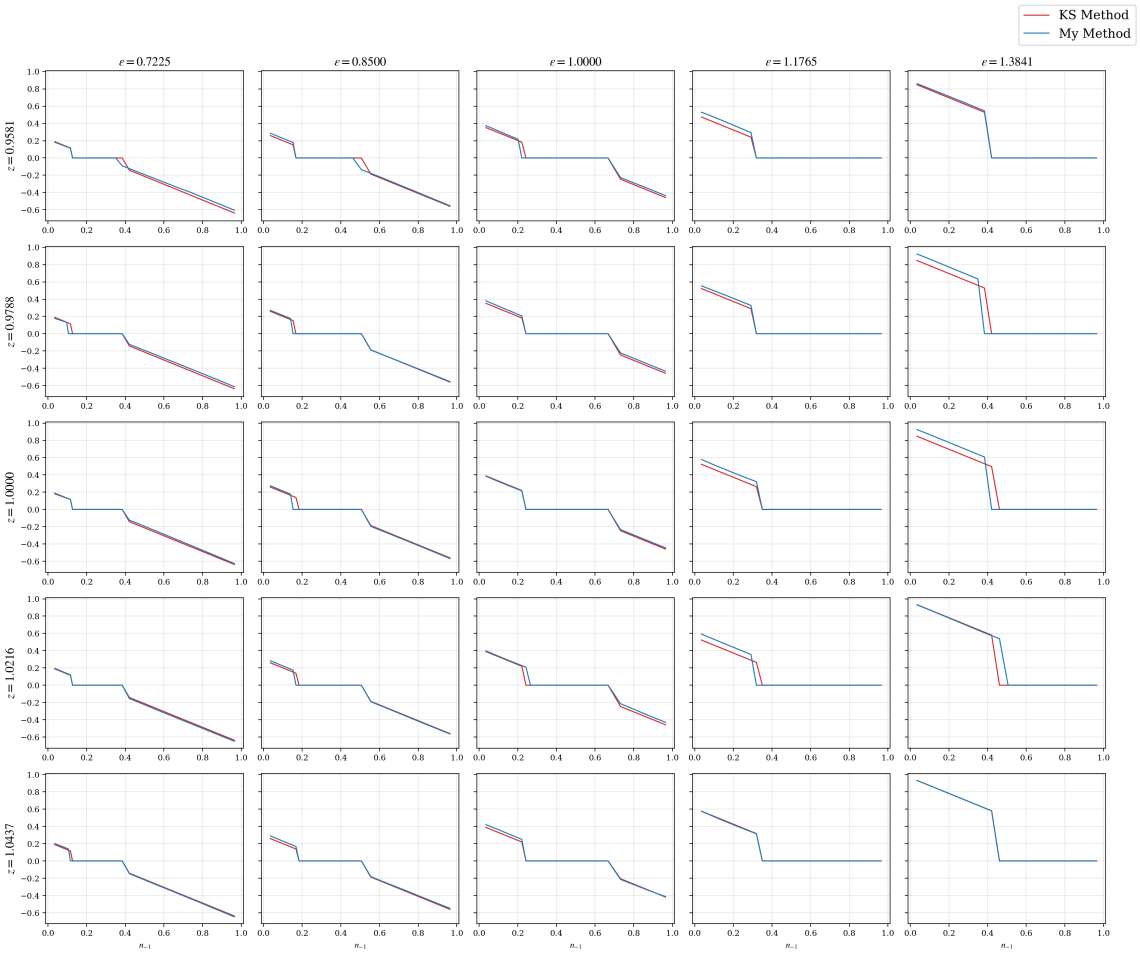
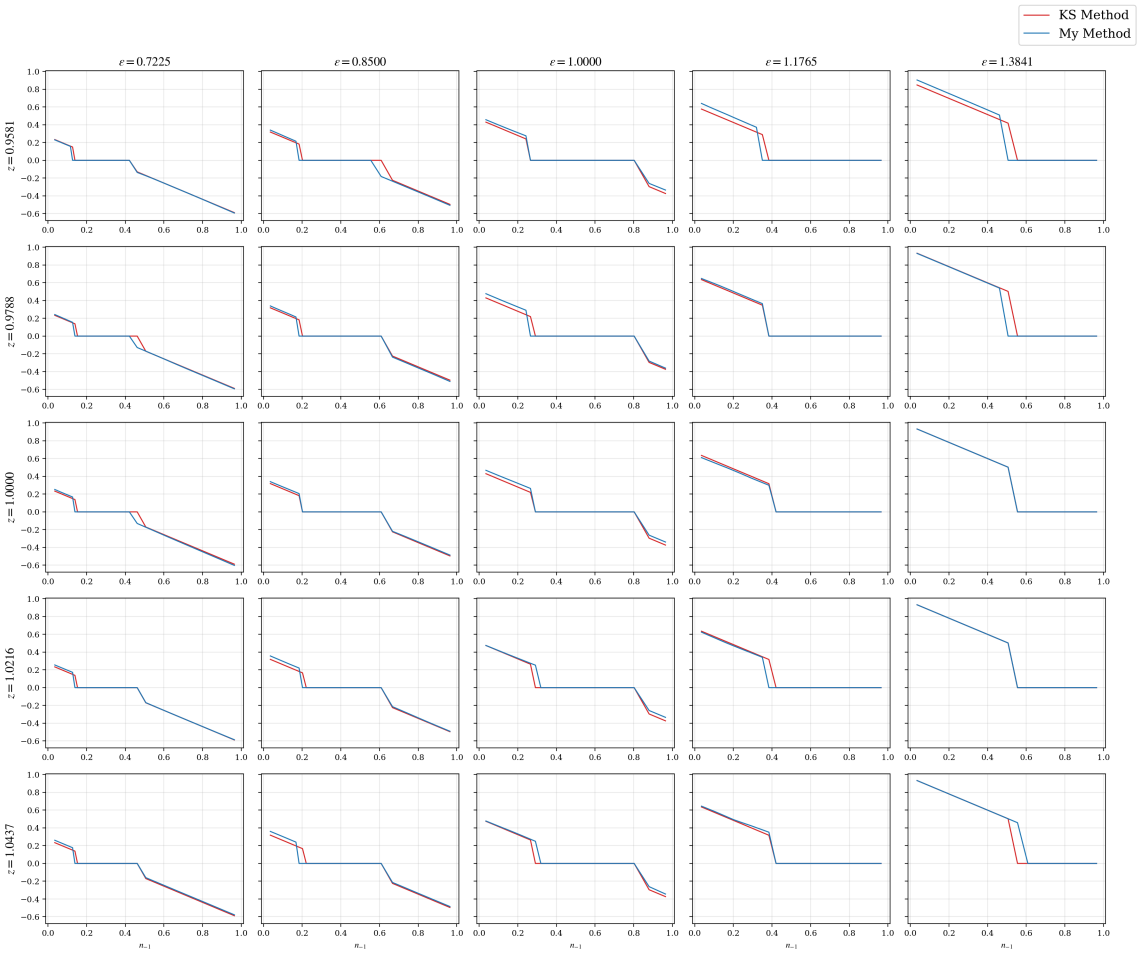


Figure 13: Labor adjustment $S(n)$ at a fixed $k = 9.3158$



B.3 Comparison across network architectures

This section compares the performance of five neural network architectures—128×2, 128×3, 128×4, 256×3, and 256×4—using the same accuracy evaluation procedure as in the previous section.

Figure 14 plots the validation Bellman error of the value function on a logarithmic scale.[29] Two clear patterns emerge. First, increasing network width from 128 to 256 substantially accelerates convergence: both the 256×3 and 256×4 architectures exhibit faster and more stable declines in the Bellman error compared to narrower networks. Second, while increasing depth from three to four hidden layers slightly improves early-stage convergence for the 256-width networks, the terminal Bellman errors of 256×3 and 256×4 are nearly identical, suggesting diminishing returns from additional depth once sufficient width is provided.

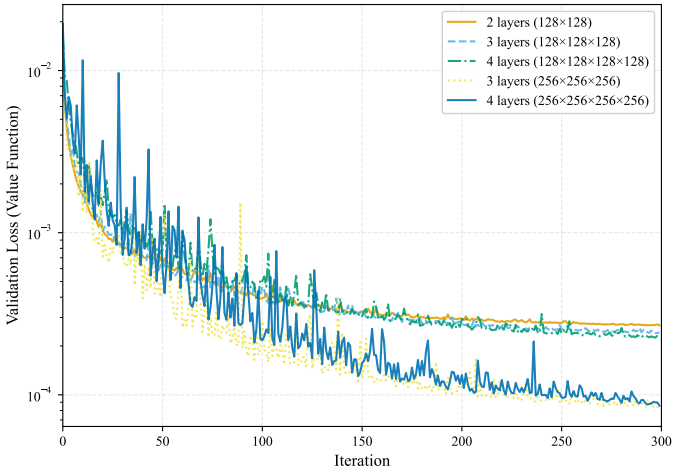
Table 14 reports the discrete-choice classification accuracy across architectures. Consistent with the convergence patterns observed in Figure 14, classification accuracy improves monotonically as network capacity increases up to 256×3. Moving from 128×2 to 128×4 raises accuracy from 90.18% to 91.51%, while increasing width to 256 yields a further improvement. Among all specifications, the 256×3 architecture achieves the highest accuracy (94.16%), corresponding to the lowest mismatch rate (5.83%). Notably, the deeper 256×4 network does not outperform 256×3 in terms of classification accuracy, despite having comparable Bellman errors.

B.4 Discussion: accuracy and computation time

This subsection discusses the trade-off between classification accuracy and computation time. Figure 14 shows that the validation Bellman error falls rapidly during the first 100 iterations and reaches roughly 2×10^{-4}, after which improvements become gradual. Over this range, discrete-choice accuracy typically reaches

[29] In one training iteration, I train three networks (value, discrete policy, and conditional policy) on datasets of 500,000 points each using batch size 256. Thus, one iteration corresponds to roughly $3 \times (500{,}000/256) \approx 1{,}900$ parameter-update steps (minibatch gradient updates).

Figure 14: Validation loss of the value function across different network architectures (log scale).



about 92–93%. By contrast, the next 200 iterations take roughly an additional one and a half hours while improving accuracy by only about one percentage point. Since this marginal gain has little effect on simulated aggregates in my experiments, in practice one can often stop training earlier without a noticeable loss in quantitative performance.

C Deep Learning Hyperparameters

This section details the hyperparameters used for training the neural network approximations of the value function (V^0_{nn}) and policy function (g_{nn}) in the two main model applications presented in the paper. The specific settings for the Khan and Thomas (2008) and Bloom et al. (2018) models are summarized in Table 15.

Table 14: Discrete-choice classification accuracy across network architectures

	128×2	128×3	128×4	256×3	256×4
Total evaluation points	3,367,000	3,367,000	3,367,000	3,367,000	3,367,000
Correctly classified	3,036,215	3,047,029	3,080,982	3,170,423	3,162,150
Correct share (%)	90.18	90.50	91.51	94.16	93.92
Misclassified	330,785	319,971	286,018	196,577	204,850
Misclassified share (%)	9.82	9.50	8.49	5.83	6.08

Note: Network architectures are denoted by width × number of hidden layers. All specifications are evaluated on the same grid of 3,367,000 state points.

Table 15: Neural Network Hyperparameters

Hyperparameter	Khan-Thomas (2008)	Bloom et al. (2018)
Network Architecture (Value/Policy)	2 hidden layers, 128 neurons each	3 hidden layers, 256 neurons each
Optimizer	ADAM	ADAM
Learning Rate Schedule	Decay $1 \times 10^{-3} \rightarrow 1 \times 10^{-5}$	Decay $5 \times 10^{-4} \rightarrow 5 \times 10^{-5}$
Batch Size (Value Function)	256	256
Batch Size (Policy Function)	256	256
Outer Loop Iterations (Max)	6	21
Target Network Update Rate (τ)	0.05	0.05

Note: The learning rate was gradually decayed over the training steps within each outer loop iteration, starting from 1×10^{-3} down to 1×10^{-5}. The target network update rate τ corresponds to the parameter in the soft update rule $\theta_{target} \leftarrow \tau\theta_{main} + (1 - \tau)\theta_{target}$.