Estimating Nonlinear Heterogeneous Agents Models with Neural Networks

Hanno Kase
ECB

Leonardo Melosi
University of Warwick, EUI, DNB & CEPR

Matthias Rottner
BIS & Deutsche Bundesbank

November 28, 2024

Introduction

- Key advancements in integrating heterogeneity into dynamic GE models
- Estimation of HA models is challenging
  - Estimation of parameters affecting the steady state (DSS) often unfeasible

Introduction

- Key advancements in integrating heterogeneity into dynamic GE models
- Estimation of HA models is challenging
  - Estimation of parameters affecting the steady state (DSS) often unfeasible
  - Available methods rely on linearizing the aggregate dynamics of the models

Introduction

- Key advancements in integrating heterogeneity into dynamic GE models
- Estimation of HA models is challenging
  - Estimation of parameters affecting the steady state (DSS) often unfeasible
  - Available methods rely on linearizing the aggregate dynamics of the models
- Yet, nonlinear aggregate dynamics are crucial to explain recent macro data
  - ZLB, deep recessions, sudden inflation rise

Introduction

- Key advancements in integrating heterogeneity into dynamic GE models
- Estimation of HA models is challenging
  - Estimation of parameters affecting the steady state (DSS) often unfeasible
  - Available methods rely on linearizing the aggregate dynamics of the models
- Yet, nonlinear aggregate dynamics are crucial to explain recent macro data
  - ZLB, deep recessions, sudden inflation rise
- New approach to estimate nonlinear HA models based on neural networks

  $\Rightarrow$ Likelihood estimation of a nonlinear HANK model using US data

  $\Rightarrow$ Estimation includes parameters affecting the steady state (DSS)

The challenges of estimating HA models

- Estimation requires many repetitions of two time-consuming steps

  1. Solve the model for a given parameters combination

  2. Evaluate an objective function (likelihood or a moment function)

The challenges of estimating HA models

- Estimation requires many repetitions of two time-consuming steps
  1. Solve the model for a given parameters combination
  2. Evaluate an objective function (likelihood or a moment function)
- This repetion is a major obstacle in estimating nonlinear HANK models.
  - Solving the models is too time consuming as they feature many states
  - Solving the DSS is also a computationally costly
  - Ex-ante unknown how many times you need to take these two steps

Structural estimation with neural networks

Key steps to make estimation of nonlinear HANK models with NNs feasible

Structural estimation with neural networks

Key steps to make estimation of nonlinear HANK models with NNs feasible

$\Rightarrow$ Parameters as pseudo-state variables of the policy functions to approximate,

Structural estimation with neural networks

Key steps to make estimation of nonlinear HANK models with NNs feasible

⇒ Parameters as pseudo-state variables of the policy functions to approximate,

- ⇒ NNs are trained to learn policy functions only once prior to estimation
- ⇒ Once trained, these NNs deliver quick and accurate solutions of the model for different values of parameters, including those influencing the DSS

Structural estimation with neural networks

Key steps to make estimation of nonlinear HANK models with NNs feasible

- ⇒ Parameters as pseudo-state variables of the policy functions to approximate,
  - ⇒ NNs are trained to learn policy functions only once prior to estimation
  - ⇒ Once trained, these NNs deliver quick and accurate solutions of the model for different values of parameters, including those influencing the DSS
- ⇒ For likelihood estimation, we introduce the neural-network particle filter
  - ⇒ To speed up likelihood evaluation
  - ⇒ To mitigate computational inaccuracies of standard particle filters

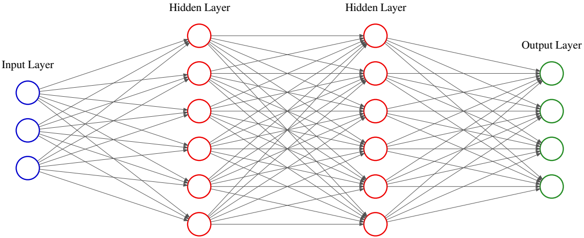
Estimation of a nonlinear (stylized) HANK model

- The model has aggregate shocks and a ZLB constraint
- Estimated using US aggregate data
- The model matches some of the moments of the data fairly well
- Parameter estimates echo those in estimated linearized RANK models
- Idiosyncratic income risk is a key contributor to aggregate volatility
    - Higher idiosyncratic risk $\Rightarrow$ more savings and lower interest rate
    - The risk of the ZLB increases and so does macro volatility
    - This mechanism explains 22% of GDP volatility
- Deflationary bias – depending on the ZLB risk – is estimated to be 30%

Literature

- **Solution methods for HA models**
  Krussel and Smith (1998); Algan et al., (2008); Reiter, (2009); Den Haan et al., (2010); Ahn et al., (2018); Boppart et al., (2018); Bayer et al., (2019); Auclert et al., (2021); and Winberry, (2021); Bhandari et al. (2023) and Bayer et al. (2024)
- **Estimation of HANK models with linearized aggregate dynamics**
  Challe et al., (2017); Bayer et al., (2024); Auclert et al., (2020); Lee, (2020); Auclert et al., (2021); Bilbiie et al., (2023); and Acharya et al., (2023)
- **NNs to solve complex dynamic macroeconomic models globally**
  Fernandez-Villaverde et al. (2020), Ebrahimi Kahou et al. (2021), Maliar et al. (2021), Azinovic et al. (2022), Duarte et al. (2024), and Valaitis and Villa (2024)
- **Solve HANK models with global methods**
  NN-based: Maliar and Maliar, (2020); Gorodnichenko et al., (2021); Fernandez-Villaverde et al., (2023); and Han et al., (2021)
  Alternative global methods: Schaab (2020); and Lin and Peruffo (2024)

What is a neural network?



- Consider a function $Y = \psi(X)$ to be approximated by a NN defined as

$$Y = \psi_{NN}(X|W),$$

- The NN combines mathematical functions performed at every neuron

What is a neural network? (cont'd)

- A single neuron assigns its inputs $x_1, x_2, \dots x_S$ some weights $w_1, w_2, \dots, w_S$ and computes their sum (adjusted by a bias w_0) to return a single output $\tilde{y}$

$$\tilde{y} = h(w_0 + \sum_{i=1}^{S} w_i x_i).$$

- The activation function $h(\cdot)$ helps the NN to capture nonlinear dynamics
- The vector W with all weights is optimized (trained) to minimize a loss fct
- All the optimizing weights across individual neurons (W) define a NN
- The neural network training usually exploits graphics processing units (GPUs) as they can process multiple computations simultaneously
  - e.g. PyTorch, Google Jax and TensorFlow

Model solution

Model solution and pseudo-state variables

- Solving the model amounts to approximate a set of policy functions:

$$\psi_t = \psi(\mathbb{S}_t | \underbrace{\tilde{\Theta}, \bar{\Theta}}_{\Theta})$$

Heterogeneity

- Key step: We could treat parameters $\tilde{\Theta}$ as pseudo-state variables

$$\psi_t = \psi(\mathbb{S}_t, \tilde{\Theta} | \bar{\Theta})$$

$\Rightarrow$ No need to re-train the NN multiple times at different parameter values

Model solution and pseudo-state variables

- Solving the model amounts to approximate a set of policy functions:

$$\psi_t = \psi(\mathbb{S}_t | \underbrace{\tilde{\Theta}, \bar{\Theta}}_{\Theta})$$

Heterogeneity

- Key step: We could treat parameters $\tilde{\Theta}$ as pseudo-state variables

$$\psi_t = \psi(\mathbb{S}_t, \tilde{\Theta} | \bar{\Theta})$$

$\Rightarrow$ No need to re-train the NN multiple times at different parameter values

- A NN can be trained to approximate the policy functions:

$$\psi_t = \psi_{NN}(\mathbb{S}_t, \tilde{\Theta} | \bar{\Theta}; W)$$

Example

Training of the neural network to solve the model

- Define a loss function as the weighted sum of squared residual errors

$$\Phi^L = \sum_{k=1}^{K} \alpha_k [F_k(\psi(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}))]^2.$$

Training of the neural network to solve the model

- Define a loss function as the weighted sum of squared residual errors

$$\Phi^L = \sum_{k=1}^{K} \alpha_k [F_k(\psi(\mathbb{S}_t, \tilde{\Theta}|\bar{\Theta}))]^2.$$

- The NN is trained by minimizing the average loss over batches of size B

$$\bar{\Phi}^L = \frac{1}{B} \sum_{b=1}^{B} \sum_{k=1}^{K} \alpha_k \left[F_k(\psi(\mathbb{S}_{t,b}, \tilde{\Theta}_b|\bar{\Theta})) \right]^2.$$

- This optimization step is repeated thousands of times

Why Does This Approach Work?

- Approximate policy functions *without conditioning* on a parameter value

  $\Rightarrow$ We only need to train the NNs once, prior to estimation

  $\Rightarrow$ No need to re-train the NN multiple times at different parameter values

- NNs are well-suited to dealing with high-dimensional problems (scalability)

  - We expand the dimensionality of the state vector by adding parameters
  - But the increase in computational burden remains manageable

Why Does This Approach Work?

- Approximate policy functions *without conditioning* on a parameter value
  - $\Rightarrow$ We only need to train the NNs once, prior to estimation
  - $\Rightarrow$ No need to re-train the NN multiple times at different parameter values
- NNs are well-suited to dealing with high-dimensional problems (scalability)
  - We expand the dimensionality of the state vector by adding parameters
  - But the increase in computational burden remains manageable
- Once the *extended* policy functions are approximated, the model's solution at any parameter values can be obtained in a fraction of seconds
  - $\Rightarrow$ Repetition of the solution and evaluation steps becomes manageable

Structural estimation

The Neural-Network Particle Filter

- Objective: evaluate the likelihood of the model More
  - For nonlinear models we can obtain the likelihood using the particle filter
  - This filter requires to track thousands of particles over multiple periods
  - Calculation is typically noisy and can be time-consuming
  - Some advantages of neural-network based particle filter
    1. Single likelihood evaluation can be done almost instantly
    2. Effectively smooths out noise from the particle filter

Training the neural network particle filter

- We obtain several thousand quasi-random parameters draws

Training the neural network particle filter

- We obtain several thousand quasi-random parameters draws
- We run the particle filter to obtain the likelihood value at each draw
    - We employ the trained NNs to approximate the policy functions and the DSS

Training the neural network particle filter

- We obtain several thousand quasi-random parameters draws
- We run the particle filter to obtain the likelihood value at each draw
  - We employ the trained NNs to approximate the policy functions and the DSS
- The NN is trained to predict the likelihood evaluated at these draws
  - Loss function: the mean squared errors between the likelihood approximated by the NN and computed by the particle filter
  - Minimize the *average* loss function over batches
- With the trained NN particle filter, the likelihood can be rapidly evaluated

◂ Overfitting

Cutting through the noise of the particle filter

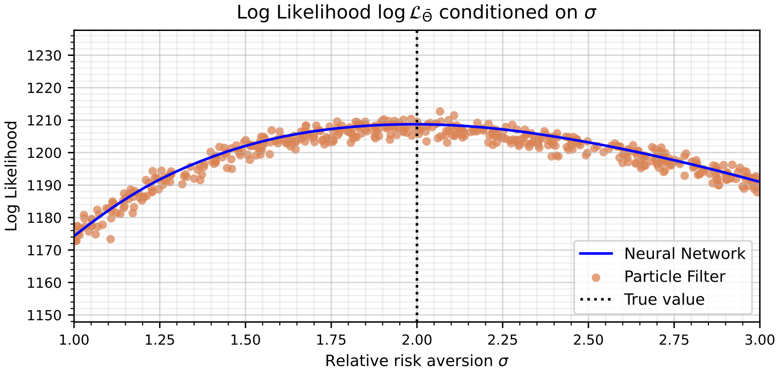


Figure: **Accuracy in likelihood evaluations: NN particle filter vs. standard particle filter.** The logorithm of the likelihood of the model as a function of the risk aversion parameter σ. The value of the fixed parameters are set to the middle of their bounds.

Back

Proofs of Concept

1. Neural network based solution vs. analytical one for a simple RANK model
   - Solution based on our neural network with pseudo-state variables

   $\Rightarrow$ Neural network solution coincides with true solution Results

Proofs of Concept

1. Neural network based solution vs. analytical one for a simple RANK model
   - Solution based on our neural network with pseudo-state variables

   ⇒ Neural network solution coincides with true solution Results

2. Neural network based estimation vs. alternative one for a nonlinear model
   - Laboratory is a RANK model with a zero lower bound

   ⇒ The estimation results are very similar Model Estimation Results

Proofs of Concept

1. Neural network based solution vs. analytical one for a simple RANK model
   - Solution based on our neural network with pseudo-state variables

   ⇒ Neural network solution coincides with true solution Results

2. Neural network based estimation vs. alternative one for a nonlinear model
   - Laboratory is a RANK model with a zero lower bound

   ⇒ The estimation results are very similar Model Estimation Results

3. Estimation of nonlinear HANK with simulated data
   - 10 parameters, also ones affecting idiosyncratic risk, are included

   ⇒ Estimates are close to true-data generating process Results

Estimation of a Nonlinear HANK model with the ZLB

- A nonlinear HANK model with the ZLB constraint, price adjustment costs à la Rotemberg, flexible wages, one asset, idiosyncratic shocks to households' labor income, borrowing limit, and aggregate uncertainty: preference shocks, shocks to TFP growth, and monetary policy ◂ The model

Estimation of a Nonlinear HANK model with the ZLB

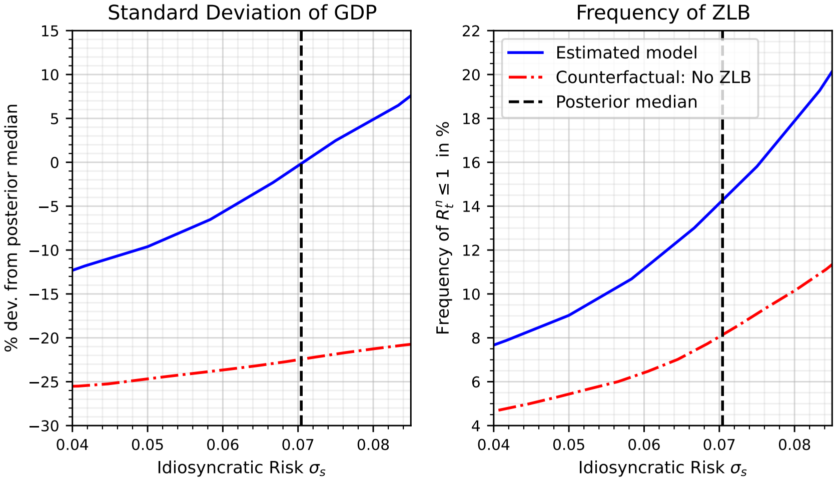
- A nonlinear HANK model with the ZLB constraint, price adjustment costs à la Rotemberg, flexible wages, one asset, idiosyncratic shocks to households' labor income, borrowing limit, and aggregate uncertainty: preference shocks, shocks to TFP growth, and monetary policy ◂ The model
- US time-series data from 1990:Q1 to 2019:Q4
  - GDP growth per capita, GDP deflator, and shadow interest rate
- Measurement error u_t follows a Gaussian distribution $\mathcal{N}(0, \Sigma_u)$
- The interactions between heterogeneity and aggregate nonlinearities allow for the identification of idiosyncratic risk, σ_s

Prior and Posterior Moments

Estimation								
Par.	Prior					Neural Network		
	Type	Mean	Std	Lower Bound	Upper Bound	Posterior Median	5%	95%
σ_s	Trc.N	0.050	0.01	0.025	0.1	0.070	0.057	0.081
$\underline{B}$	Trc.N	-0.5	0.025	-0.65	-0.35	-0.50	-0.54	-0.46
φ	Trc.N	100	5	70	120	101	94	107
θ_Π	Trc.N	2.25	0.125	1.75	2.75	2.43	2.20	2.67
θ_Y	Trc.N	1.0	0.025	0.75	1.25	0.96	0.92	1.00
ρ_z	Trc.N	0.4	0.025	0.2	0.6	0.43	0.39	0.47
ρ_{mp}	Trc.N	0.9	0.005	0.85	0.95	0.905	0.897	0.913
σ_ζ	Trc.N	0.015	0.1%	0.01	0.02	0.011	0.012	0.013
σ_z	Trc.N	0.4%	0.1%	0.3%	0.6%	0.47%	0.43%	0.53%
σ_{mp}	Trc.N	0.06%	0.01%	0.05%	0.2%	0.15%	0.14%	0.16%

Calibrated Parameters ◂ Convergence ◂ Aggregate policies

Interactions between nonlinearities and heterogeneity



⇒ The estimated idiosyncratic risk affected by the volatility of the observables

How good is what we got?

Standard deviations

	Model	Data
GDP growth	0.6947	0.5831
Inflation	1.1511	0.9045
Federal funds rate	2.561	2.7537

Autocorrelations

	Model	Data
GDP growth	0.1355	0.4050
Inflation	0.8146	0.5456
Federal funds rate	0.7219	0.9707

Avg. Gini coefficient

	Model	Data
Wealth distrib.	0.8793	0.8410

Conclusions

- Novel integrated neural-network based estimation procedure
  - Estimation of models with hundreds of state variables (HA, many countries or sectors) and nonlinear constraints possible
- Estimation of a HANK with individual and aggregate nonlinearities and risk
  - Interactions between nonlinearities, aggregate uncertainty, and heterogeneity
- New techniques to solve and estimate the economic models of the future

Example Codes

Code for the analytical example!

`https://github.com/tseep/estimating-hank-nn`

Example Codes

Code for the analytical example!

https://github.com/tseep/estimating-hank-nn

Or run the code directly in the cloud with Google Colab

https://colab.research.google.com/github/tseep/estimating-hank-nn/blob/main/examples/colab_analytical.ipynb

Model solution with neural networks

- Objective: Solving a nonlinear DSGE model
  - State variables $\mathbb{S}_t$, shocks ν_t and structural parameters Θ
- Model's dynamics summarized by a set of (nonlinear) transition equations

$$\mathbb{S}_t = f\left(\mathbb{S}_{t-1}, \nu_t; \Theta\right),$$

where function f is generally unknown and needs to be obtained numerically

- Heterogeneity: Approximate distributions with a finite number of agents
  - The state variables and the vector of shocks

$$\mathbb{S}_t = \left\{\left\{\mathbb{S}_t^i\right\}_{i=1}^L, \mathbb{S}_t^A\right\} \quad \text{and} \quad \nu_t = \left\{\left\{\nu_t^i\right\}_{i=1}^L, \nu_t^A\right\}.$$

  - Individual and aggregate policy functions

$$\psi_t^i = \psi^I(\mathbb{S}_t^i, \mathbb{S}_t|\Theta) \quad \text{and} \quad \psi_t^A = \psi^A(\mathbb{S}_t|\Theta).$$

Incorporation of Heterogeneity

- Heterogeneity usually assumes the existence of a continuum of agents
  - $\rightarrow$ Distribution of states and shocks is infinite

$$\int \mathbb{S}_t^i \, d\Omega \quad \text{and} \quad \int \nu_t^i \, d\Omega,$$

- We approximate the distribution with a finite number of agents L

$$\left\{\mathbb{S}_t^i\right\}_{i=1}^L \quad \text{and} \quad \left\{\nu_t^i\right\}_{i=1}^L .$$

- The state variables and the vector of shocks

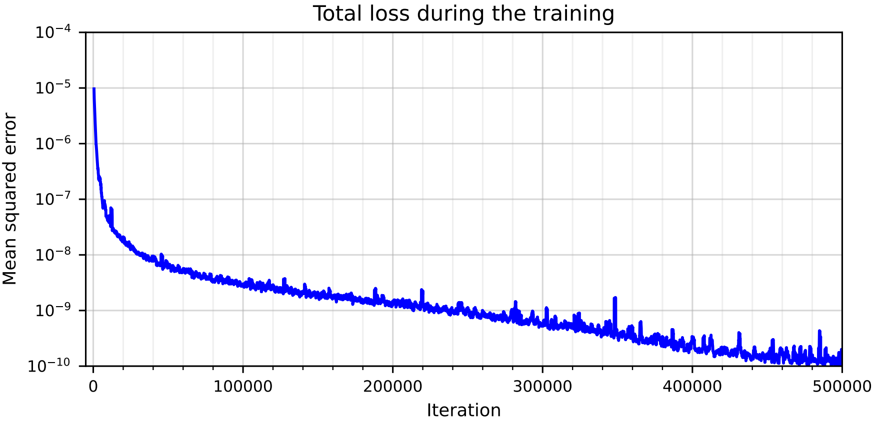
$$\mathbb{S}_t = \left\{\left\{\mathbb{S}_t^i\right\}_{i=1}^L, \mathbb{S}_t^A\right\} \quad \text{and} \quad \nu_t = \left\{\left\{\nu_t^i\right\}_{i=1}^L, \nu_t^A\right\},$$

- Individual and aggregate policy functions:

$$\psi_t^i = \psi^I(\mathbb{S}_t^i, \mathbb{S}_t \,|\, \bar{\Theta}) \quad \text{and} \quad \psi_t^A = \psi^A(\mathbb{S}_t \,|\, \bar{\Theta}).$$

Back

Training of Neural Network and Loss Function



Back

Particle Filter

- Observation equation connects the state variables with the observables $\mathbb{Y}_t$:

$$\mathbb{Y}_t = g(\mathbb{S}_t|\tilde{\Theta}) + u_t,$$

where g is a function and u_t is a measurement error

- Particle filter determines the likelihood

$$\mathcal{L}\left(\mathbb{Y}_{1:T};\tilde{\Theta}\right) = \Omega^{PF}\left(\mathbb{Y}_{1:T};\tilde{\Theta}\right)$$

- Particle filter can be noisy and very time consuming for complex models
- Using a filter to calculate the likelihood is still a bottleneck Back

Nonlinear RANK Model with ZLB

- Off-the-shelf New Keynesian model
  - Shocks to households' preference to consumption
  - Price rigidities a la Rotemberg
  - Zero lower bound constraint on the nominal interest rate

$$R_t = \max\left[1, R\left(\frac{\Pi_t}{\Pi}\right)^{\theta_\Pi}\left(\frac{Y_t}{Y}\right)^{\theta_Y}\right]$$

- We are interested in solving and estimating it in its nonlinear specification

◂ Back

Neural Network and Estimation

- Training NN over 100000 iterations and batch size of 200 economies
- We train the NN by drawing from the bounded parameter space
- Stochastic solution from simulating the model after each draw

- Observation equation with a sample size of 1000 periods

$$\begin{bmatrix} \text{Output Growth} \\ \text{Inflation} \\ \text{Interest Rate} \end{bmatrix} = \begin{bmatrix} 400\left(\frac{Y_t}{Y_{t-1}-1}\right) \\ 400\left(\Pi_t - 1\right) \\ 400\left(R_t - 1\right) \end{bmatrix} + u_t$$

- Estimation includes five structural parameters
- Priors are truncated normal densities
- 15000 data points to train neural network based particle filter Back

Back

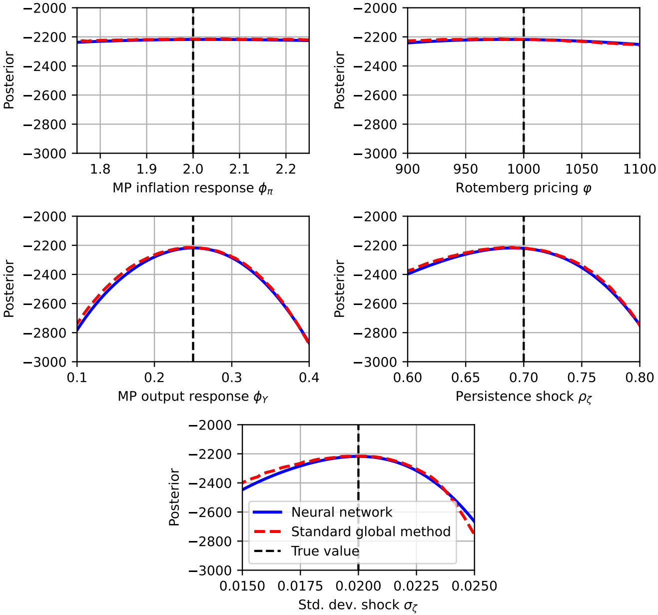
Estimation Results

Estimation							
Par.	Cal.	Neural Network			Conventional Approach		
	True Value	Posterior Median	5%	95%	Posterior Median	5%	95%
θ_Π	2.0	2.02	1.87	2.17	2.06	1.94	2.20
θ_Y	0.25	0.251	0.238	0.263	0.248	0.237	0.258
φ	1000	988.6	935.1	1036.7	973.7	911.2	1037.2
ρ_ζ	0.8	0.686	0.669	0.701	0.691	0.670	0.710
σ^ζ	0.02	0.020	0.020	0.021	0.020	0.019	0.020

- Neural network based estimation works very well
    - Posterior median is very close to the true value
- The bounds of neural network and conventional method are very similar
- Neural network method is much faster and much more scalable!

Back

Bayesian Estimation with NN: Posterior



Back

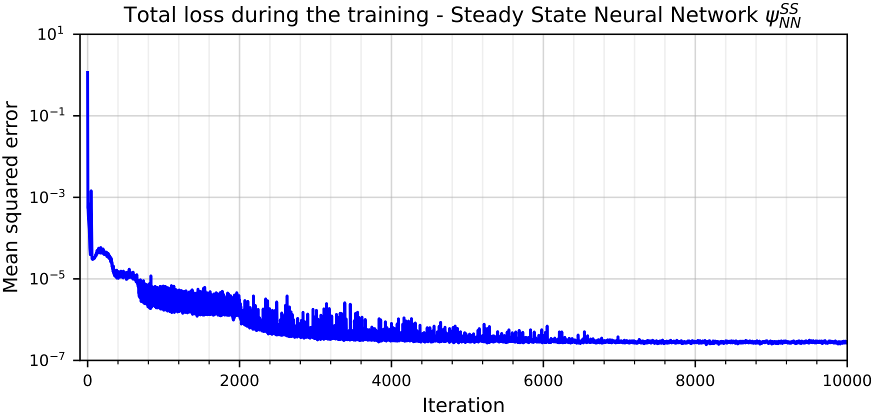
Calibration HANK Model

Calibrated Parameters					
Parameters		Value	Parameters		Value
β	Discount factor	0.9975	γ^{τ}	Tax progressivity	0.18
σ	Relative risk aversion	1	D	Government debt	1
η	Inverse Frisch elasticity	0.72	Π	Inflation target	1.00625
ϵ	Price elasticity demand	11	ρ_s	Persistence labor prod	0.9
χ	Disutility labor	0.74	ρ_{ζ}	Persistence pref. shock	0.7
g	Average growth rate	1.0033			

Back

Estimation - Step 1a: Model Solution and Neural Networks

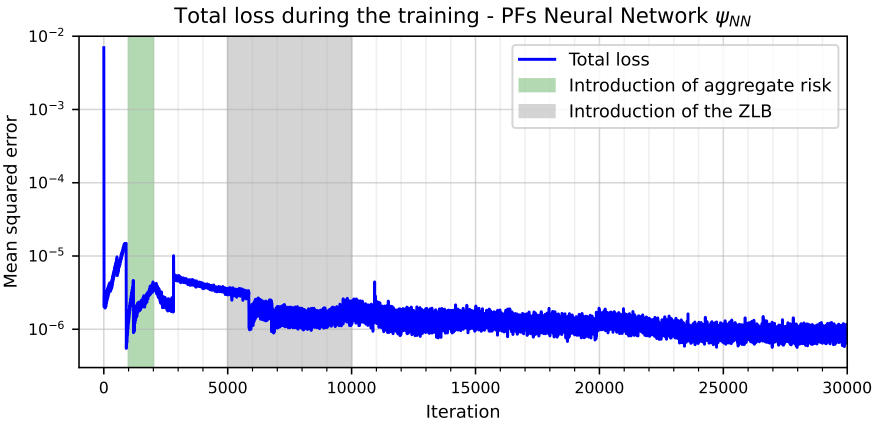
- NN training of model without aggregate risk (Aiyagari version)



Back

Estimation - Step 1b: Nonlinear Model Solution

- NN training of the full nonlinear model version
  - Stepwise introduction of aggregate risk and the ZLB



Back

Overfitting

- Overfitting occurs when NNs learn too much from the training sample
  - e.g. the noise generated by computational inaccuracies of the particle filter

$\Rightarrow$ Obtain a validation sample of randomly drawn parameters and likelihoods

  - We do not optimize the weights of the NN
  - We compute the loss function implied by the NN in the validation sample
  - We compare the loss in the validation sample to that in the training sample
  - Losses should be similar to dispel concerns of overfitting

Back

Overfitting

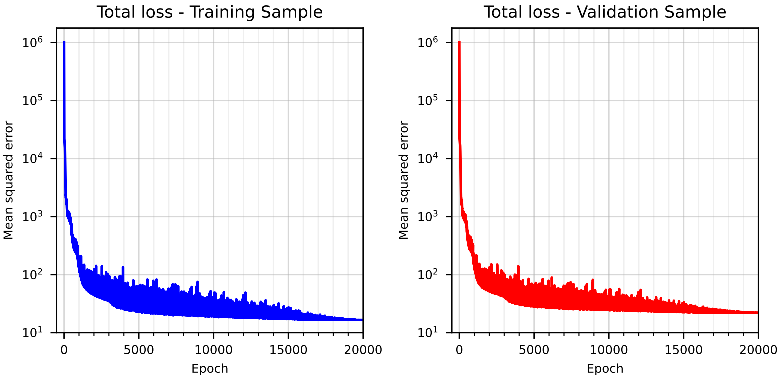


Figure: **Training and validation convergence.** The figure shows the total loss over the the training sample (left) and over the validation sample (right). An epoch is completed when all the points in the training or validation sample are utilized. The vertical axis is expressed in a logarithmic scale.

Back

Example: Linearized NK model

- Small off-the-shelf linearized three equation NK model with TFP shock
- Features a closed-form analytical solution

$$
\begin{aligned}
\hat{X}_t &= E_t\hat{X}_{t+1} - \sigma^{-1}\left(\phi_\Pi\hat{\Pi}_t + \phi_Y\hat{X}_t - E_t\hat{\Pi}_{t+1} - \hat{R}^*_t\right)\\
\hat{\Pi}_t &= \kappa\hat{X}_t + \beta E_t\hat{\Pi}_{t+1}\\
\hat{R}^*_t &= \rho_A\hat{R}^*_{t-1} + \sigma(\rho_A - 1)\omega\sigma_A\epsilon^A_t
\end{aligned}
$$

where $\hat{X}_t$ is the output gap, $\hat{\Pi}$ is inflation, R^*_t is the natural rate of interest, and ϵ^A_t is a TFP shock

Example: Solution to Linearized NK Model

- Solution to equation system depends on state variables and parameters

$$\begin{pmatrix} \hat{X}_t \\ \hat{\Pi}_t \end{pmatrix} = \psi \left(\underbrace{\hat{R}_t^*}_{\text{State } S_t}, \underbrace{\beta, \sigma, \eta, \phi, \theta_\Pi, \theta_Y, \rho_A, \sigma_A}_{\text{Parameters } \tilde{\Theta}} \right).$$

Example: Solution to Linearized NK Model

- Solution to equation system depends on state variables and parameters

$$\begin{pmatrix} \hat{X}_t \\ \hat{\Pi}_t \end{pmatrix} = \psi \left(\underbrace{\hat{R}^*_t}_{\text{State } S_t}, \underbrace{\beta, \sigma, \eta, \phi, \theta_\Pi, \theta_Y, \rho_A, \sigma_A}_{\text{Parameters } \tilde{\Theta}} \right).$$

- The analytical solution is given as

$$\hat{X}_t = \frac{1 - \beta\rho_A}{(\sigma(1-\rho_A) + \theta_Y)(1 - \beta\rho_A) + \kappa(\theta_\Pi - \rho_A)} \hat{R}^*_t,$$
$$\hat{\Pi}_t = \frac{\kappa}{(\sigma(1-\rho_A) + \theta_Y)(1 - \beta\rho_A) + \kappa(\theta_\Pi - \rho_A)} \hat{R}^*_t.$$

Solving the NK Model with a Neural Network

1. Approximate the policy function with a deep neural network:
   - Two policy functions:

$$\begin{pmatrix} \hat{X}_t \\ \hat{\Pi}_t \end{pmatrix} \approx \psi_{NN}(\underbrace{\hat{R}_t^*}_{\mathbb{S}_t}, \underbrace{\beta, \sigma, \eta, \phi, \theta_\Pi, \theta_Y, \rho_A, \sigma_A}_{\tilde{\Theta}})$$

2. Construct the loss function $\bar{\Phi}^L$ for optimization
   - Based on minimization of squared residual errors

$$err_{IS} = \hat{X}_t - \left(E_t \hat{X}_{t+1} - \sigma^{-1}\left(\phi_\Pi \hat{\Pi}_t + \phi_Y \hat{X}_t - E_t \hat{\Pi}_{t+1} - \hat{R}_t^*\right)\right)$$

$$err_{PC} = \hat{\Pi}_t - \left(\kappa \hat{X}_t + \beta E_t \hat{\Pi}_{t+1}\right)$$

   - Loss function weighs the errors and averages over batch size B of 500

$$\bar{\Phi}^L = \alpha_1 \frac{1}{B} \sum_{b=1}^{B} (err_{IS}{}^b)^2 + \alpha_2 \frac{1}{B} \sum_{b=1}^{B} (err_{PC}{}^b)^2$$

Back

Solving the NK Model with a NN (cont'd)

3. Train the deep neural networks using stochastic optimization
   - 500 000 iterations with a batch size of 1000
     1. Draw parameters from a bounded parameter space

Parameters		LB	UB	Parameters		LB	UB
β	Discount factor	0.95	0.99	θ_Π	MP inflation	1.25	2.5
σ	Relative risk aver.	1	3	θ_Y	MP output	0.0	0.5
η	Inverse Frisch	0.25	2	ρ_A	Persistence TFP	0.8	0.95
φ	Price duration	0.5	0.9	σ_A	Std. dev. TFP	0.02	0.1

     2. Draw points from the state space via simulation

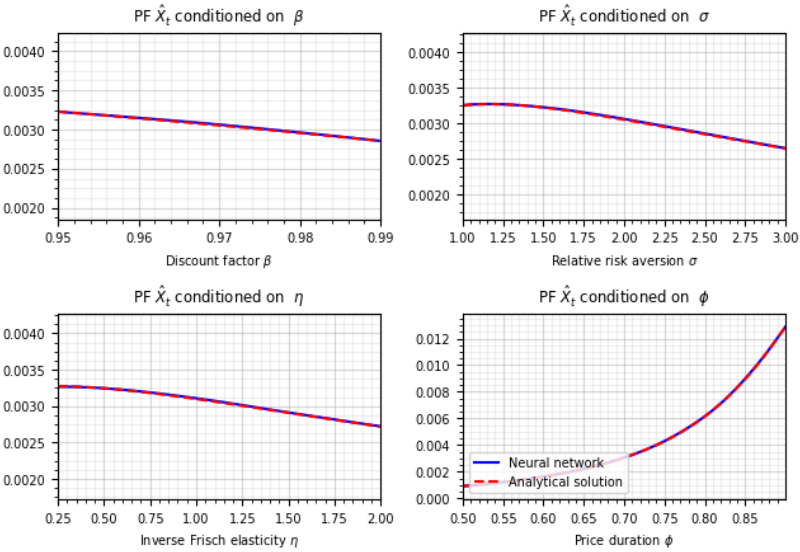
$$\hat{R}_t^* = \rho_A \hat{R}_{t-1}^* + \sigma(\rho_A - 1)\omega\sigma_A\epsilon_t^A$$

     3. Optimizer (ADAM) to choose the weights of the NN to minimize $\bar{\Phi}^L$

Training

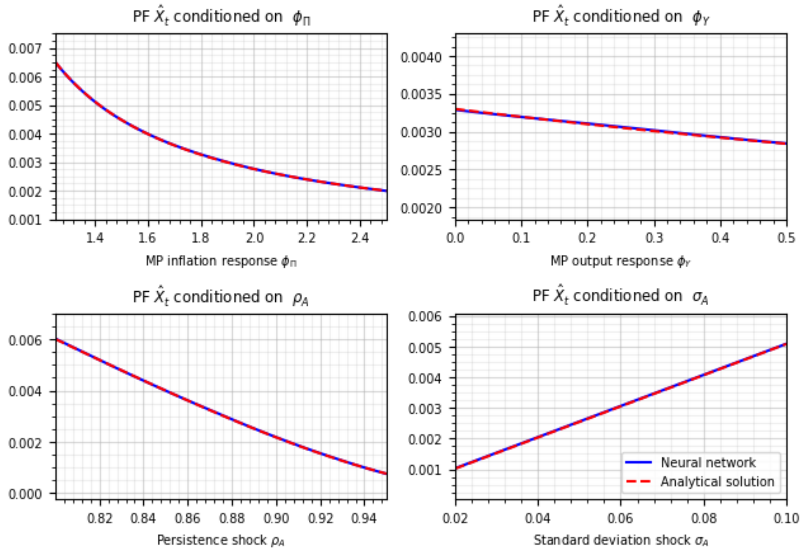
Back

PoF 1: NN-approximation of policy functions



Back

PoF 1: NN-approximation of policy functions (cont'd)



Back

State and pseudo state variables and policy functions

- Discretization of the number of agents: $L = 100$ agents
- 2 individual state variables:

$$\left\{ \tilde{B}^i_{t-1} \right\}_{i=1}^{L} \quad \text{and} \quad \left\{ s^i_t \right\}_{i=1}^{L}$$

- 4 aggregate state variables:

$$R^N_{t-1}, \quad \zeta_t, \quad z_t, \quad \text{and} \quad mp_t$$

- One idiosyncratic shock and three aggregate shocks

$$\left\{ \left\{ \epsilon^{s,i}_t \right\}_{i=1}^{L}, \epsilon^{\zeta}_t, \epsilon^{z}_t, \epsilon^{m}_t \right\}$$

- 10 pseudo state variables

$$\tilde{\Theta} = \{\sigma_s, \underline{B}, \varphi, \theta_\Pi, \theta_Y, \rho_z, \rho_m, \sigma_\zeta, \sigma_z, \sigma_m\}$$

- 10 calibrated parameters

$$\bar{\Theta} = \{\beta, \eta, \sigma, \bar{a}, \chi, \gamma^\tau, \Pi, D, \rho_s, \rho_\zeta\}$$

Back

Loss functions to minimize for the training of NNs

The error associated with the Fisher-Burmeister function – smooth way to represent the Kuhn-Tucker conditions: $\mu_t^i \geq 0$, $\left(\tilde{B}_t^i - \underline{B}\right) \geq 0$, and $\mu_t^i \times \left(\tilde{B}_t^i - \underline{B}\right) = 0$ – so as to enforce the borrowing limit at the individual household level:

$$\left\{ L^{1,i} = \left(\Psi^{FB} \left(1 - \bar{\lambda}_t^i, \tilde{B}_t^i - \underline{B} \right) \right)^2 \right\}_{i=1}^{L}, \tag{1}$$

where $\bar{\lambda}_t^i$ are the multipliers associated with the Euler equation of each household i and $L^{1,i}$ is the squared error of agent i.

Loss functions to minimize for the training of NNs (cont'd)

$$L^2 = \left(\left[\varphi \left(\frac{\Pi_t}{\Pi} - 1 \right) \frac{\Pi_t}{\Pi} \right] - (1-\epsilon) - \epsilon MC_t \right.$$
$$\left. - \beta \varphi \frac{1}{M} \sum_{m=1}^{M} \left[\left(\frac{\exp(\zeta_{t+1}^m)}{\exp(\zeta_t)} \right) \left(\frac{\tilde{C}_{t+1}^m}{\tilde{C}_t} \right)^{-\sigma} \left(\frac{\Pi_{t+1}^m}{\Pi} - 1 \right) \frac{\Pi_{t+1}^m}{\Pi} \frac{\tilde{Y}_{t+1}^m}{\tilde{Y}_t} \right] \right)^2, \tag{2}$$

$$L^3 = \left(D - \frac{1}{L} \sum_{i=1}^{L} B_t^i \right)^2, \tag{3}$$

$$L^4 = \frac{1}{M} \sum_{m=1}^{M} \left(D - \frac{1}{L} \sum_{i=1}^{L} B_{t+1}^{i,m} \right)^2, \tag{4}$$

$$L^5 = \left(\tilde{Y}_t - \tilde{C}_t \right)^2, \tag{5}$$

$$L^6 = \frac{1}{M} \sum_{m=1}^{M} \left(\tilde{Y}_{t+1}^m - \tilde{C}_{t+1}^m \right)^2. \tag{6}$$

◂ Back

PoF 3: Estimation of nonlinear HANK with simulated data

Estimation									
Par.	True	Prior					NN		
	Value	Type	Mean	Std	Lower Bound	Upper Bound	Posterior Median	5%	95%
Parameters affecting the DSS									
$100\sigma_s$	5.00	Trc.N	5.00	1.000	2.50	10.0	4.28	3.17	5.31
$\underline{B}$	−0.50	Trc.N	−0.50	0.010	−0.65	−0.35	−0.50	−0.54	−0.46
Other parameters									
φ	100	Trc.N	100	5.000	70	120	100	92	108
θ_Π	2.25	Trc.N	2.25	0.125	1.75	2.75	2.40	2.25	2.55
θ_Y	1.00	Trc.N	1.00	0.025	0.75	1.25	1.01	0.97	1.05
ρ_z	0.40	Trc.N	0.40	0.025	0.20	0.60	0.40	0.37	0.45
ρ_m	0.90	Trc.N	0.90	0.005	0.85	0.95	0.90	0.89	0.91
$100\sigma_\zeta$	1.50	Trc.N	1.50	0.100	1.00	2.00	1.45	1.34	1.57
$100\sigma_z$	0.40	Trc.N	0.40	0.100	0.30	0.60	0.36	0.32	0.40
$100\sigma_m$	0.06	Trc.N	0.06	0.010	0.05	0.20	0.06	0.05	0.07

Back

Estimation of Nonlinear HANK with Neural Networks

- HANK with individual and aggregate nonlinearities
  - Households face idiosyncratic income risk s_t^i and a borrowing limit $\underline{B}$

$$E_0 \sum_{t=0}^{\infty} \beta^t \exp(\zeta_t) \left[\left(\frac{1}{1-\sigma} \right) \left(\frac{C_t}{A_t} \right)^{1-\sigma} - \chi \left(\frac{1}{1+\eta} \right) (H_t^i)^{1+\eta} \right]$$

$$\text{s.t. } C_t^i + B_t^i = \tau_t \left(\frac{W_t}{A_t} \exp(s_t^i) H_t^i \right)^{1-\gamma_\tau} + \frac{R_{t-1}}{\Pi_t} B_{t-1}^i + Div_t \exp(s_t^i)$$

$$B_t^i \geq \underline{B}$$

  where idiosyncratic risk follows an AR(1) process: $s_t^i = \rho_s s_{t-1}^i + \sigma_s \epsilon_t^{s,i}$

  - Aggregate shocks: preference ζ, growth rate z_t, and monetary policy mp_t

Estimation of Nonlinear HANK with Neural Networks

- HANK with individual and aggregate nonlinearities
  - Households face idiosyncratic income risk s_t^i and a borrowing limit $\underline{B}$

$$E_0 \sum_{t=0}^{\infty} \beta^t \exp(\zeta_t) \left[\left(\frac{1}{1-\sigma} \right) \left(\frac{C_t}{A_t} \right)^{1-\sigma} - \chi \left(\frac{1}{1+\eta} \right) (H_t^i)^{1+\eta} \right]$$

$$\text{s.t.} \quad C_t^i + B_t^i = \tau_t \left(\frac{W_t}{A_t} \exp(s_t^i) H_t^i \right)^{1-\gamma_\tau} + \frac{R_{t-1}}{\Pi_t} B_{t-1}^i + Div_t \exp(s_t^i)$$

$$B_t^i \geq \underline{B}$$

  where idiosyncratic risk follows an AR(1) process: $s_t^i = \rho_s s_{t-1}^i + \sigma_s \epsilon_t^{s,i}$

  - Aggregate shocks: preference ζ, growth rate z_t, and monetary policy mp_t
  - Monopolistically competitive firms and Rotemberg pricing

Estimation of Nonlinear HANK with Neural Networks

- HANK with individual and aggregate nonlinearities
  - Households face idiosyncratic income risk s_t^i and a borrowing limit $\underline{B}$

$$E_0 \sum_{t=0}^{\infty} \beta^t \exp(\zeta_t) \left[\left(\frac{1}{1-\sigma} \right) \left(\frac{C_t}{A_t} \right)^{1-\sigma} - \chi \left(\frac{1}{1+\eta} \right) (H_t^i)^{1+\eta} \right]$$

$$\text{s.t.} \quad C_t^i + B_t^i = \tau_t \left(\frac{W_t}{A_t} \exp(s_t^i) H_t^i \right)^{1-\gamma_\tau} + \frac{R_{t-1}}{\Pi_t} B_{t-1}^i + Div_t \exp(s_t^i)$$

$$B_t^i \geq \underline{B}$$

  where idiosyncratic risk follows an AR(1) process: $s_t^i = \rho_s s_{t-1}^i + \sigma_s \epsilon_t^{s,i}$

  - Aggregate shocks: preference ζ, growth rate z_t, and monetary policy mp_t
  - Monopolistically competitive firms and Rotemberg pricing
  - Monetary policy is constrained by the zero lower bound

$$R_t = \max \left[1, \; R \left(\frac{\Pi_t}{\Pi} \right)^{\theta_\Pi} \left(\frac{Y_t}{A_t Y} \right)^{\theta_Y} \exp(mp_t) \right]$$

Back

Estimation - Convergence

- NN training of the full nonlinear model version
  - Stepwise introduction of aggregate risk and the ZLB

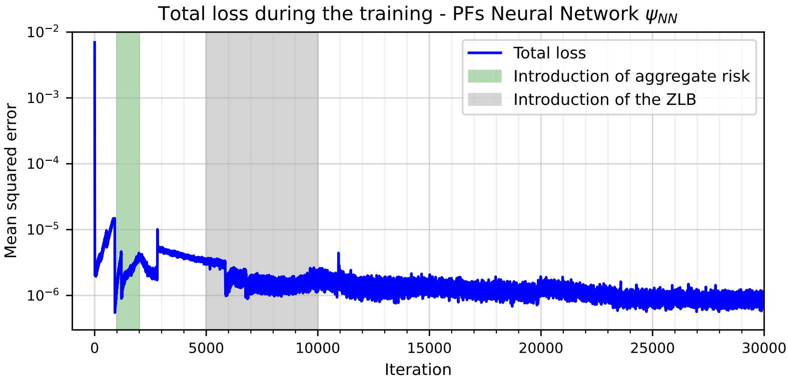
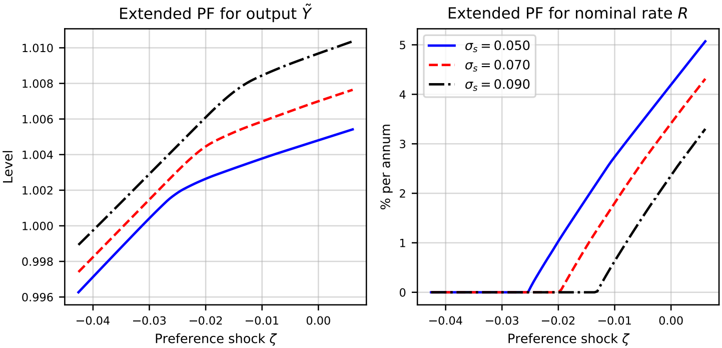


Figure: **Convergence of the NN solution for the HANK model.** The figure shows the dynamics of the mean squared error during the training of the extended individual and aggregate policy functions. The shaded areas indicate the periods in which we introduce aggregate risk and the ZLB. The vertical axis has a logarithmic scale.

Estimation: Aggregate Policy Functions

- Policy functions for output and inflation for varying preference shock
  - Zero lower bound creates nonlinearity
  - Degree of nonlinearity depends on degree of idiosyncratic risk



Back