Estimating Nonlinear Heterogeneous Agent Models with Neural Networks

Hanno Kase, Leonardo Melosi, Matthias Rottner

September 4, 2025

Introduction

- Over last three decades, researchers have focused on integrating heterogeneity in macroeconomic models.
- DGE Models: Enables study of distributional issues, economic fluctuations, and stabilization policies in a micro-founded framework.
- Accounting for these non-linear dynamics is essential to understand macroeconomic events, including recurring and prolonged periods at Zero Lower Bound (ZLB), deep recessions and recent rise in inflation.
- This paper introduces Neural-Network (NN) based nethod for solving and estimating models with heterogeneous agents, fully incorporating non-linear dynamics.

A neural-network framework for estimation

- Begin with a primer on neural networks.
- How NN can be used efficiently to characterize policy functions of a model treating parameters as psedo-state variables.
- How NN can be trained to characterize the deterministic steady state of the model.
- Training the NN particle filter and estimating likelihood functions.
- Solving Models:
  - Model #1: small scale linearized DSGE model
  - Model #2: Nonlinear RANK model
  - Model #3: Nonlinear HANK model

What are neural networks?

Consider a function:

$$Y = \psi(X)$$

where

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

are column vectors.

This function can then be approximated by a neural network as:

$$Y \approx \psi_{\text{NN}}(X \mid W) \tag{1}$$

with W denoting the weights that define the NN.

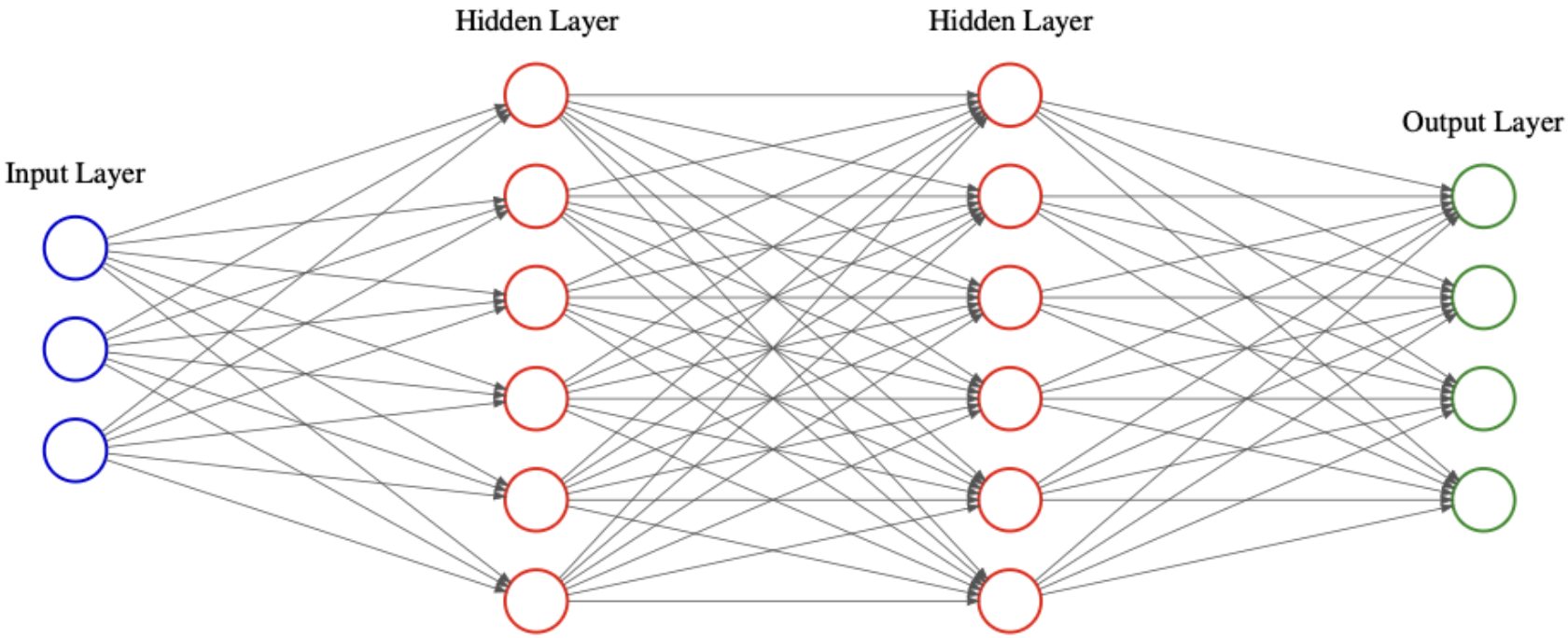


Figure 1: Illustration of a neural network. An illustrative example of a NN with three inputs (blue circles), two hidden layers (the two rows of red circles) with six neurons (the red circles) each, and four outputs (the green circles).

Neural networks

A single neuron is an affine function with:

- inputs $x_1, x_2 \ldots x_M$
- weights $w_1, w_2 \ldots w_M$
- bias w_0
- nonlinear activation function $g(\cdot)$

From all of this, we get a single output $\widetilde{y}$

$$\widetilde{y} = g\left(w_0 + \sum_{i=1}^{M} w_i x_i\right) \tag{2}$$

Neural networks

In matrix notation, each layer of the NN can be written as

$$y = g(\mathcal{A}^h(x)) \quad \text{where} \quad \mathcal{A}^h(x) \equiv W^h x \tag{3}$$

x, y and W are all vectors now.

Optimization of the weights depend on the back-propagation algorithm, which computes all partial derivatives of the loss with respect to the weights of the NN.

Accuracy of the NN can be achieved by adjusting the learning rate during the training.

Neural-network solution method

Consider a large class of DGE models by the following transition equations:

$$\mathbb{S}_t = f(\mathbb{S}_{t-1}, \nu_t | \Theta) \tag{4}$$

Solving this model gives O policy functions that map state variables to a set of control variables ψ_t for given values of the model parameters

$$\Theta : \psi_t = \psi(\mathbb{S}_t | \Theta)$$

These policy functions satisfy set of K equilibrium conditions derived from the model equations

$$F(\psi(\mathbb{S}_t | \Theta)) = 0 \tag{5}$$

Neural-network solution method

Extended policy functions:

The parameters to be estimated are considered as pseudo-state variables when approximating policy functions

$$\psi_t = \psi(\mathbb{S}_t, \widetilde{\Theta}|\bar{\Theta}) \tag{6}$$

Θ is split into:

- $\widetilde{\Theta}$ - set of parameters to estimate
- $\bar{\Theta}$ - set of parameters that are fixed in estimation

The transition equation then becomes:

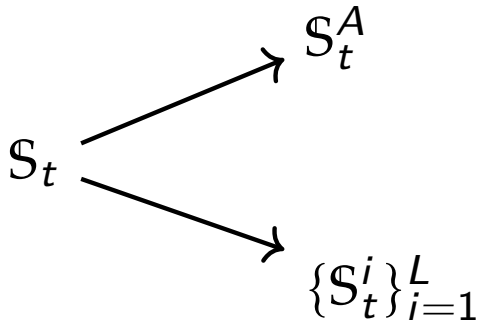
$$\mathbb{S}_t = f(\mathbb{S}_{t-1}, \nu_t, \widetilde{\Theta}|\bar{\Theta}) \tag{7}$$

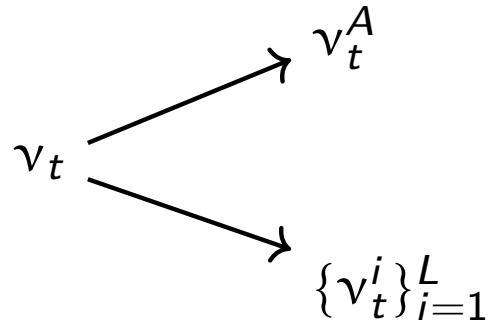
Neural-network solution method

Agents' heterogeneity:

Discretize the model population by keeping track of a finite number L of agents.

Transition equation can be written in the same form as in Equation (4), but now deal with both individuals and aggregate levels.





Neural-network solution method

Two sets of policy functions to approximate:

- Aggregate policy functions: $\Psi^A(\cdot)$
- Individual policy function: $\Psi^I(\cdot)$

The extended forms would look like:

$$\Psi_t^A = \Psi^A(\mathbb{S}_t, \widetilde{\Theta}|\bar{\Theta}) \quad \text{and} \quad \Psi_t^i = \Psi^I(\mathbb{S}_t^i, \mathbb{S}_t, \widetilde{\Theta}|\bar{\Theta}) \tag{8}$$

Then, the number of variables we have:

- $S = S^i \times L + S^A$
- $U = U^i \times L + U^A$
- $O = O^i \times L + O^A$

Neural-network solution method

Neural networks and training:

Set up two NNs $\psi^I_{NN}(\cdot)$ and $\psi^A_{NN}(\cdot)$ to approximate individual and aggregate policy functions $\psi^I(\cdot)$ and $\psi^A(\cdot)$.

Train the two NNs by choosing their weights W to minimize weighted squared residual error for the equilibrium conditions:

$$\Phi^L = \sum_{k=1}^{K} \alpha_k [F_k(\psi_{NN}(\mathbb{S}_t, \widetilde{\Theta}|\bar{\Theta}))]^2 \tag{9}$$

In this framework, the number of optimality conditions is as much as the number of policy functions. So,

$$K \geqslant O^i \times L + O^A$$

Neural-network solution method

For faster computations in GPUs, the authors have also proposed the following methods (parallelization):

Two NNs are trained on a batch with size B

- B pseudo-randomly drawn vectors, $\mathbb{S}_{t,b}$
- parameters to estimate: $\widetilde{\Theta}_b$

In each step, minimize the loss functions Φ^L averaged across batch size B.

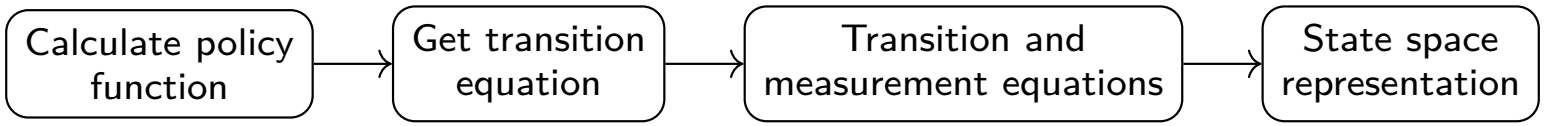
$$\bar{\Phi}^L = \frac{1}{B} \sum_{b=1}^{B} \sum_{k=1}^{K} \alpha_k [F_k(\psi_{NN}(\mathbb{S}_{t,b}, \widetilde{\Theta}_b | \bar{\Theta}))]^2 \tag{10}$$

Neural Network Particle Filter

In DSGE models, likelihood functions are calculated in two steps:

1. Solution step:

The model's policy functions are characterized here.



2. Evaluation step:

Having a filter on state space representation can give us the likelihood function.

Work of a filter: models one step ahead predictive densities over available sample period

Neural Network Particle Filter

The filter looks like:

$$p_{\bar{\Theta}}(y_t|y_{t-1},...y_1,\widetilde{\Theta})$$

The product of model's predictive densities in every period $y \in 1,...T$ is the likelihood function:

$$\mathcal{L}_{\bar{\Theta}} = \prod_{t=1}^{T} p_{\bar{\Theta}}(y_t|y_{t-1},...y_1,\widetilde{\Theta}) \tag{11}$$

Traditionally, we require MC filters (Fernandez-Villaverde and Rubio-Ramirez, 2007), and even in this case, this is used as a benchmark

Training the NN particle filter

NN particle filter can be trained in three steps:

- **Step 1**: Obtain several 1000s of quasi-random draws from the parameter space.
- **Step 2**: Execute the standard particle filter to derive likelihoods at each of these draws
- **Step 3**: NN particle filter is trained to minimize the MSE between predicted and actual likelihood values.

Solving Model #1

Solving a linearized small scale DSGE model:

There's a NK-DSGE model with a TFP shock. Model is log-linearized around the unique steady-state equilibrium, which results in following equations:

$$\widehat{Y}_t = E_t \widehat{Y}_{t+1} - \sigma^{-1} \left(\widehat{R}_t - E_t \widehat{\Pi}_{t+1} \right), \tag{12}$$

$$\widehat{\Pi}_t = \kappa \left(\widehat{Y}_t - \widehat{Y}_t^* \right) + \beta E_t \widehat{\Pi}_{t+1}, \tag{13}$$

$$\widehat{R}_t = \phi_\Pi \widehat{\Pi}_t + \phi_Y \widehat{X}_t, \tag{14}$$

$$\widehat{Y}_t^* = \omega \widehat{A}_t, \tag{15}$$

$$\widehat{A}_t = \rho_A \widehat{A}_{t-1} + \sigma_A \epsilon_t^A, \tag{16}$$

Solving Model #1

Here are some definitions:

- Output: $\widehat{Y}_t = \dfrac{Y_t - Y}{Y}$
- Inflation: $\widehat{\Pi}_t = \Pi_t - \Pi$
- Nominal interest rate: $\widehat{R}_t = R_t - R$
- Output in the flex-price economy: $\widehat{Y}_t^* = \dfrac{Y_t^* - Y^*}{Y^*}$
- TFP: $\widehat{A}_t = \dfrac{A_t - A}{A}$
- Output gap: $\widehat{X}_t = \widehat{Y}_t - \widehat{Y}_t^*$
- Shock: $\epsilon_t^A \sim \mathcal{N}(0, 1)$

Solving Model #1

After some manipulations in the equations, they become:

$$\widehat{X}_t = E_t\widehat{X}_{t+1} - \sigma^{-1}\left(\phi_\Pi\widehat{\Pi}_t + \phi_Y\widehat{X}_t - E_t\widehat{\Pi}_{t+1} - \widehat{R}_t^*\right), \tag{17}$$

$$\widehat{\Pi}_t = \kappa\widehat{X}_t + \beta E_t\widehat{\Pi}_{t+1}, \tag{18}$$

$$\widehat{R}_t^* = \rho_A\widehat{R}_{t-1}^* + \sigma(\rho_A - 1)\,\omega\sigma_A\epsilon_t^A, \tag{19}$$

where $\widehat{R}_t^\star$ is the natural rate of interest which follows an exogenous process derived from the TFP process.

Solving Model #1

The analytical solution for the output gap and the inflation gap is as follows:

$$\widehat{X}_t = \frac{(1-\beta\rho_A)}{(\sigma(1-\rho_A)+\theta_Y)(1-\beta\rho_A)+\kappa(\theta_\Pi-\rho_A)}\,\widehat{R}_t^*, \tag{20}$$

$$\widehat{\Pi}_t = \frac{\kappa}{(\sigma(1-\rho_A)+\theta_Y)(1-\beta\rho_A)+\kappa(\theta_\Pi-\rho_A)}\,\widehat{R}_t^*. \tag{21}$$

The analytical solution for the output gap and the inflation gap depends both on the state variable $\widehat{R}_t^\star$ and the parameters.

Mapping of Model #1 to the general framework

The state variable, structural shock and the control variables are as follows:

$$\mathbb{S}_t = \{\widehat{R}_t^{\star}\}, \quad \text{and} \quad \nu_t = \{\epsilon_t^A\}, \quad \text{and} \quad \psi_t = \{\widehat{X}_t, \widehat{\Pi}\} \tag{22}$$

In this model, there are no fixed parameters. So $\overline{\Theta}$ is an empty set.

The estimated parameters are as follows:

$$\widetilde{\Theta} = \{\beta, \sigma, \eta, \phi, \theta_{\Pi}, \theta_Y, \rho_A, \sigma_A\} \tag{23}$$

The NN is then trained to determine the output gap and inflation:

$$\begin{pmatrix} \widehat{X}_t \\ \widehat{\Pi}_t \end{pmatrix} = \psi_{NN}\left(\mathbb{S}_t, \widetilde{\Theta} \mid \overline{\Theta}\right). \tag{24}$$

Mapping of Model #1 to the general framework

- Use 500,000 iterations to train the NN by minimizing the residual error for the Euler equation and Phillips curve.
- NN with five hidden layers with 256 neurons each.
- Continuously differentiable exponential linear units (CELU)
- AdamW optimizer
- Learning rate: Cosine annealing LR

For parallel computing, B = 1000.

Mapping of Model #1 to the general framework

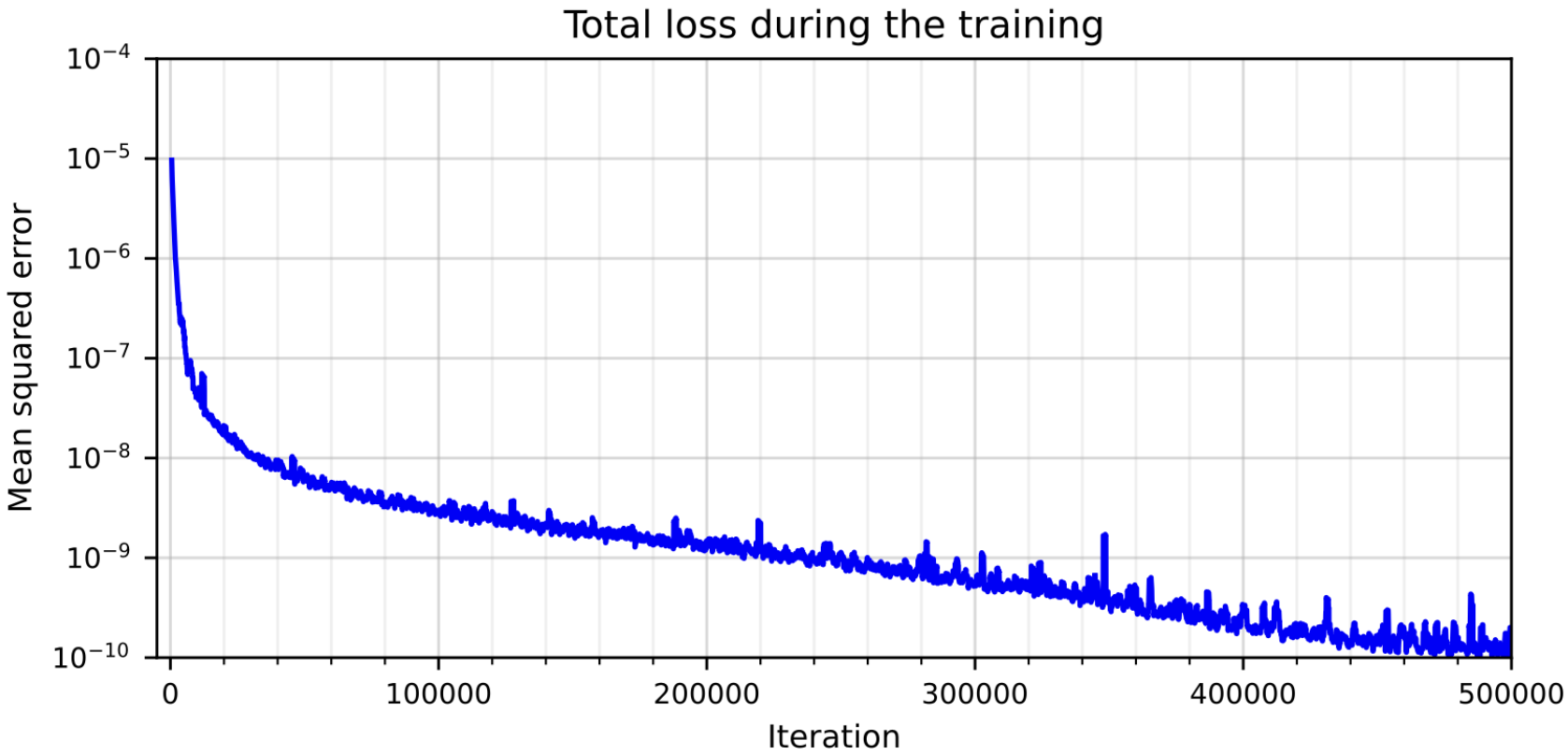


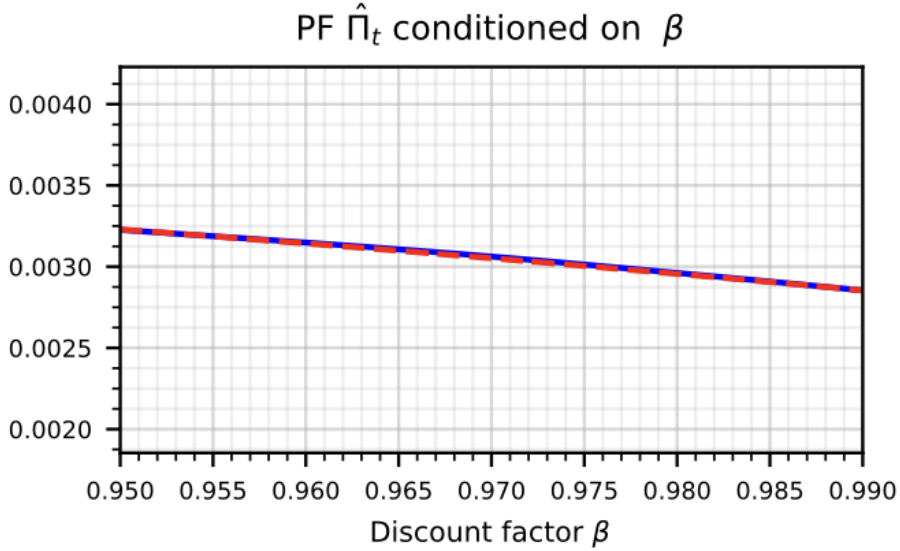
Figure 2: Convergence of the neural network solution. The figure shows the dynamics of the mean squared error. The loss is calculated as a moving average over 1000 iterations. The vertical axis has a logarithmic scale.

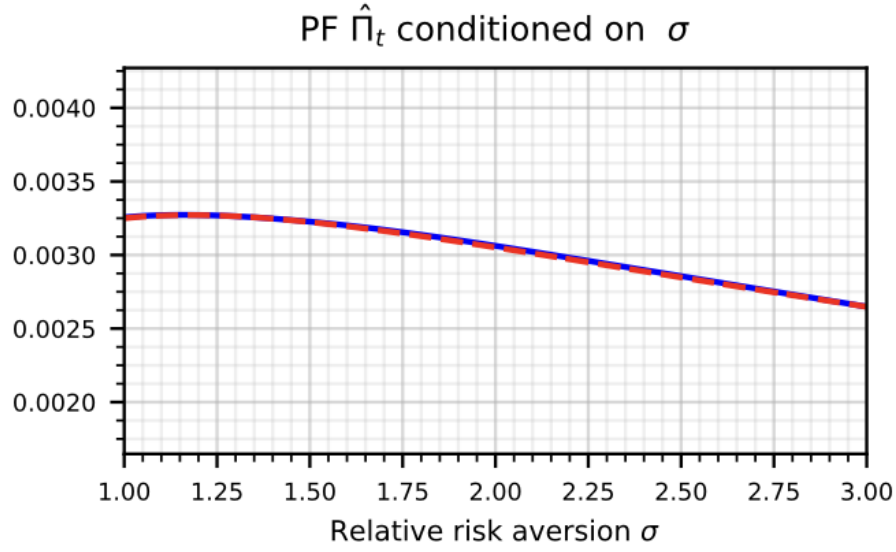
Mapping of Model #1 to the general framework

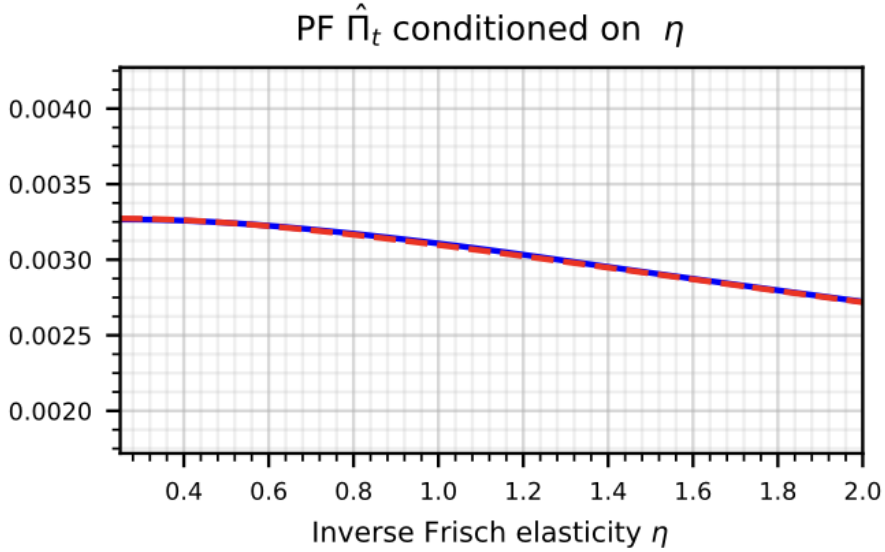
With the analytical solutions of the linearized DSGE model already in place, authors have considered the following values for the parameters:

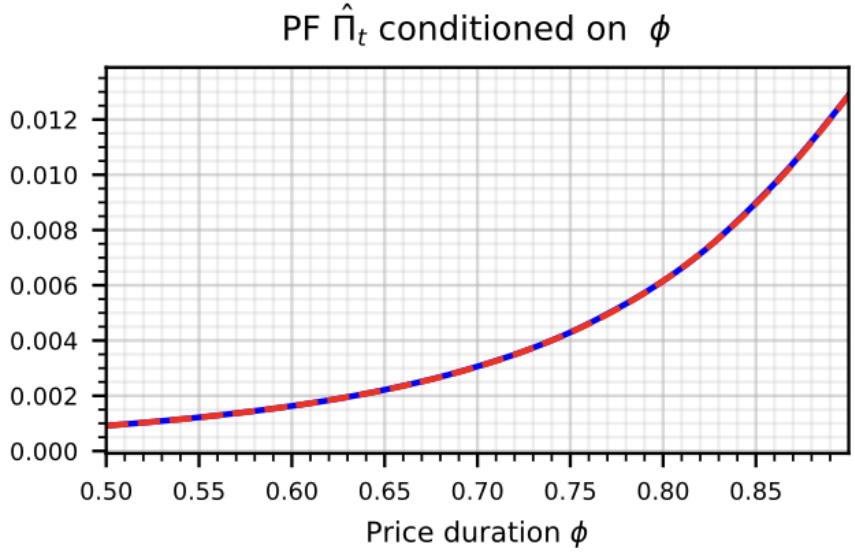
Parameters		LB	UB	Parameters		LB	UB
β	Discount factor	0.95	0.99	θ_Π	Mon.pol. inflation response	1.25	2.5
σ	Relative risk aversion	1	3	θ_Y	Mon.pol. output response	0.0	0.5
η	Inverse Frisch elasticity	1	4	ρ_A	Persistence TFP shock	0.8	0.95
φ	Price duration	0.5	0.9	σ_A	Std. dev. TFP shock	0.02	0.1

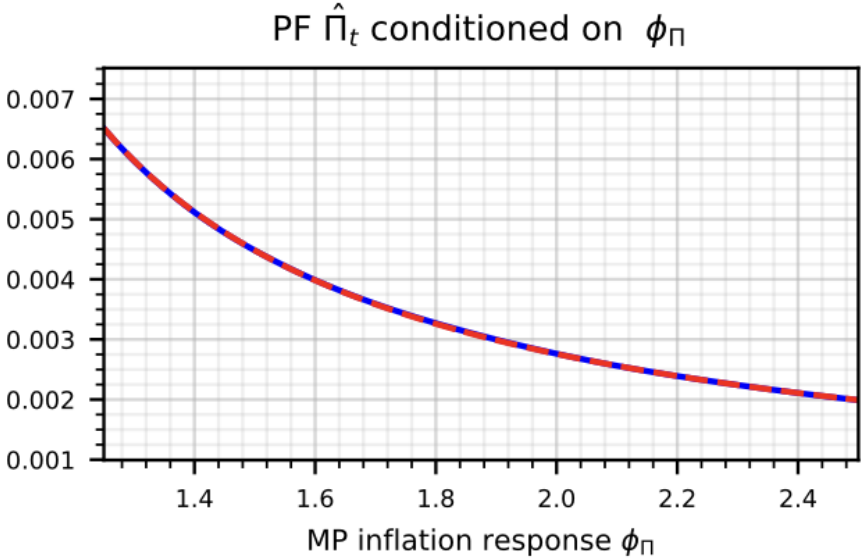
Table 1: Model parameters values – first proof of concept. The table shows the parameters of the three-equation New Keynesian model. The lower bound (LB) and upper bound (UB) show how we truncate the parameter space when training the NN to approximate the model solution.

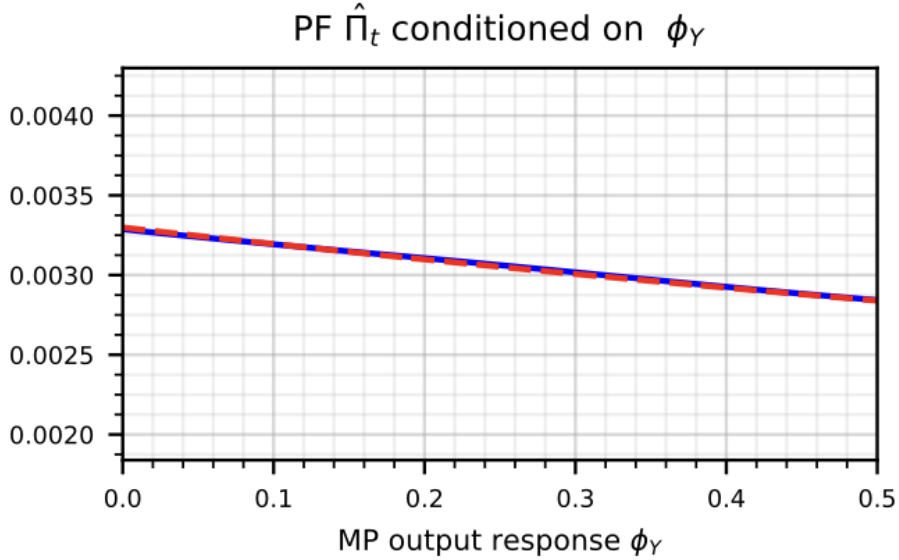

PF $\hat{\Pi}_t$ conditioned on β
0.0040
0.0035
0.0030
0.0025
0.0020
0.950 0.955 0.960 0.965 0.970 0.975 0.980 0.985 0.990
Discount factor β

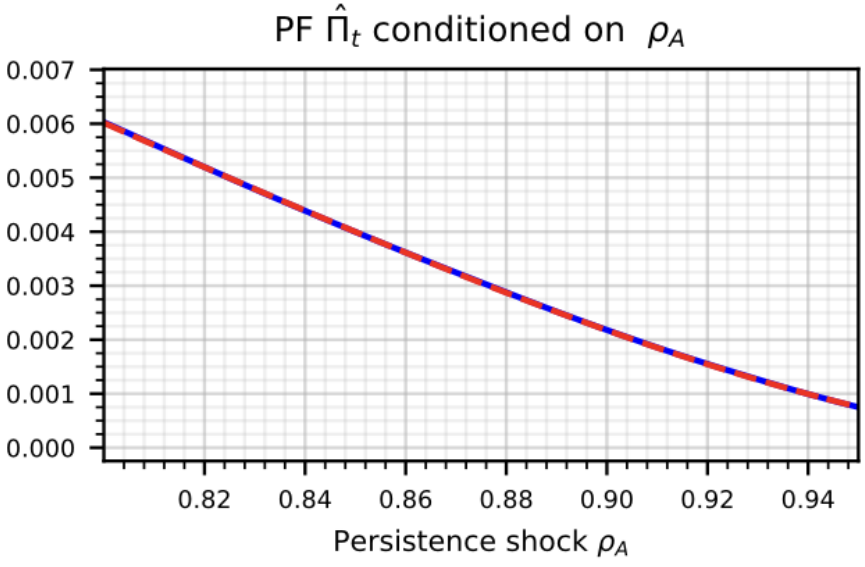

PF $\hat{\Pi}_t$ conditioned on σ
0.0040
0.0035
0.0030
0.0025
0.0020
1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00
Relative risk aversion σ

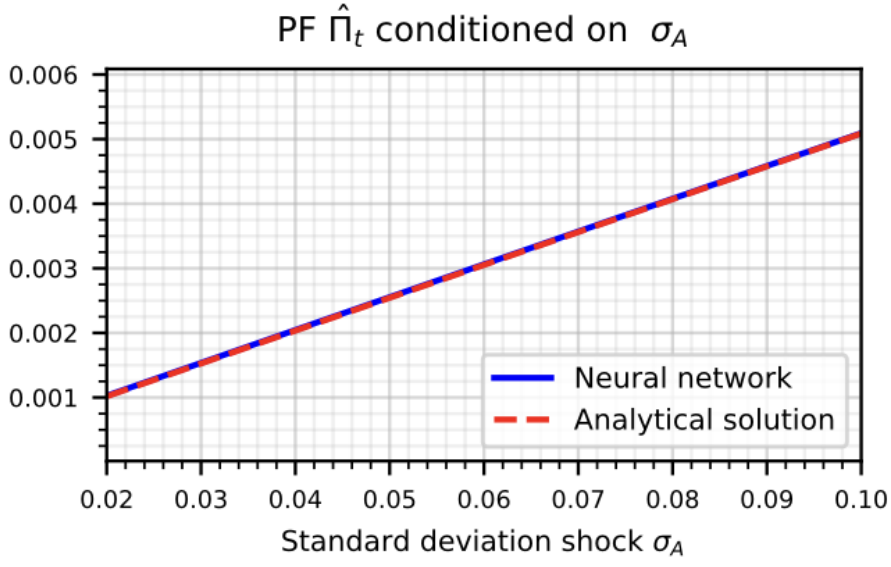

PF $\hat{\Pi}_t$ conditioned on η
0.0040
0.0035
0.0030
0.0025
0.0020
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
Inverse Frisch elasticity η

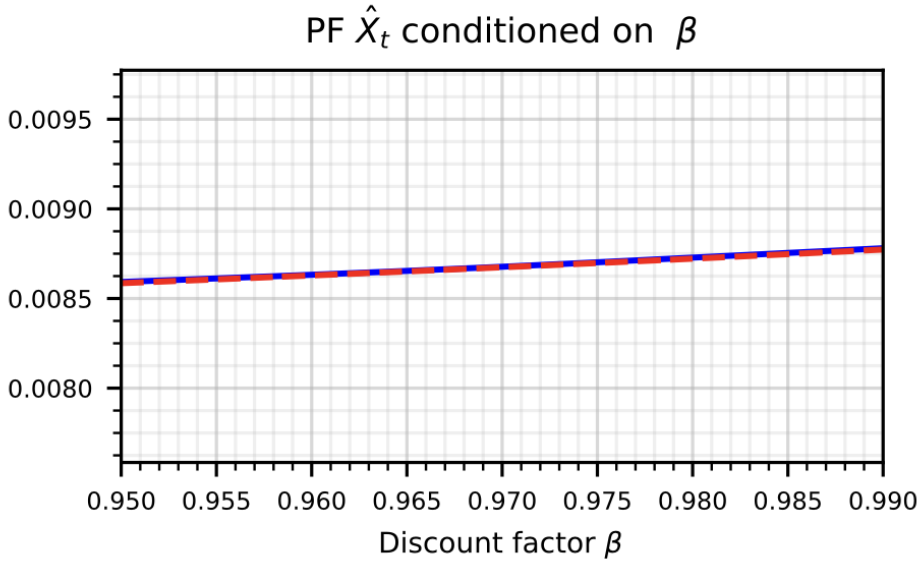

PF $\hat{\Pi}_t$ conditioned on ϕ
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85
Price duration ϕ

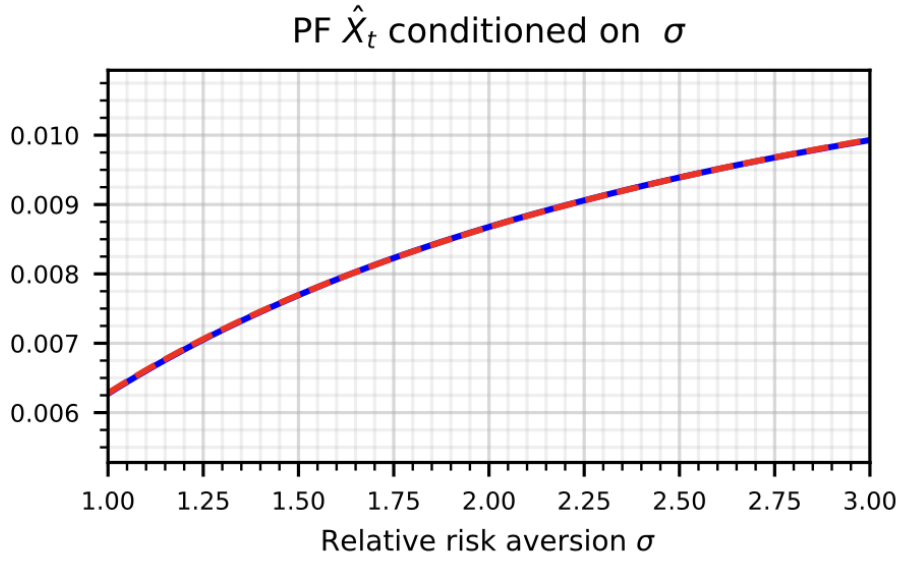

PF $\hat{\Pi}_t$ conditioned on ϕ_Π
0.007
0.006
0.005
0.004
0.003
0.002
0.001
1.4
1.6
1.8
2.0
2.2
2.4
MP inflation response ϕ_Π

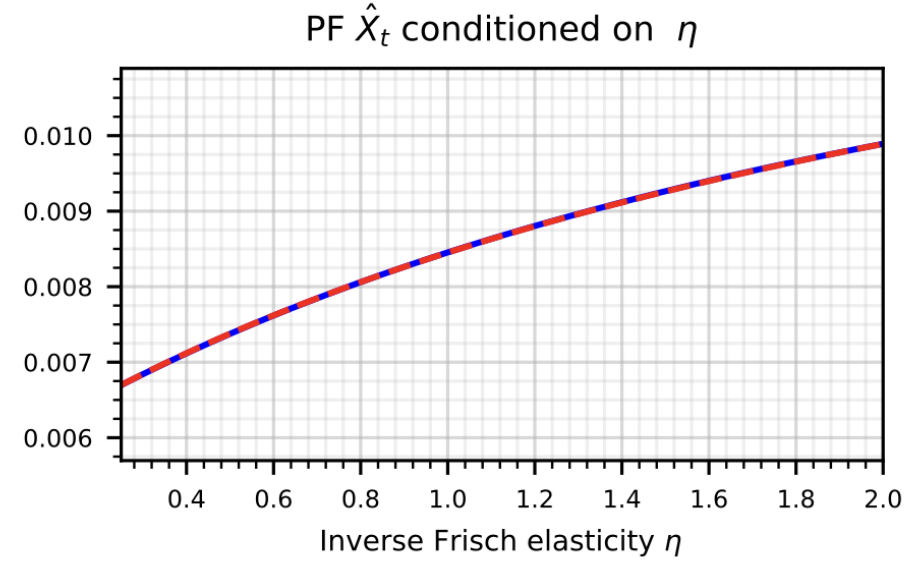

PF $\hat{\Pi}_t$ conditioned on ϕ_Y
0.0040
0.0035
0.0030
0.0025
0.0020
0.0
0.1
0.2
0.3
0.4
0.5
MP output response ϕ_Y

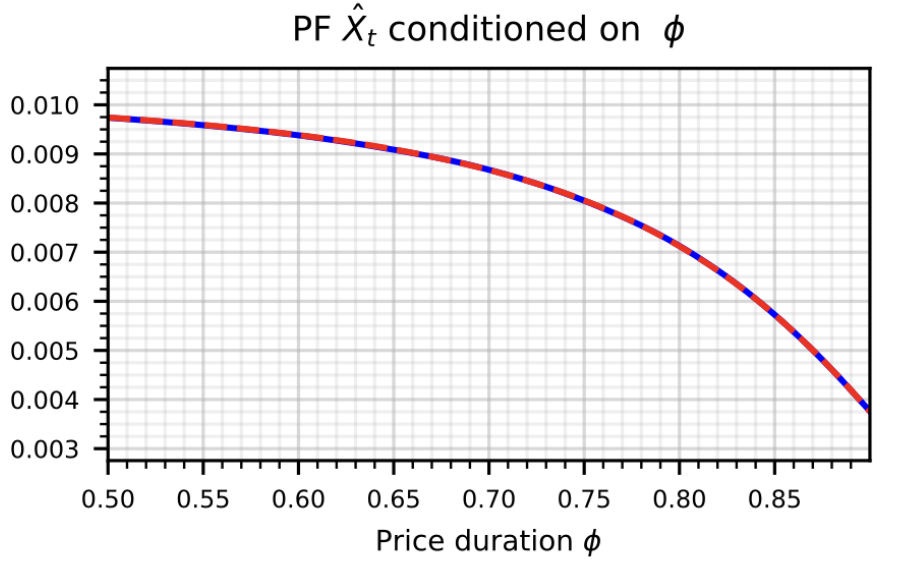

PF $\hat{\Pi}_t$ conditioned on ρ_A
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0.82
0.84
0.86
0.88
0.90
0.92
0.94
Persistence shock ρ_A

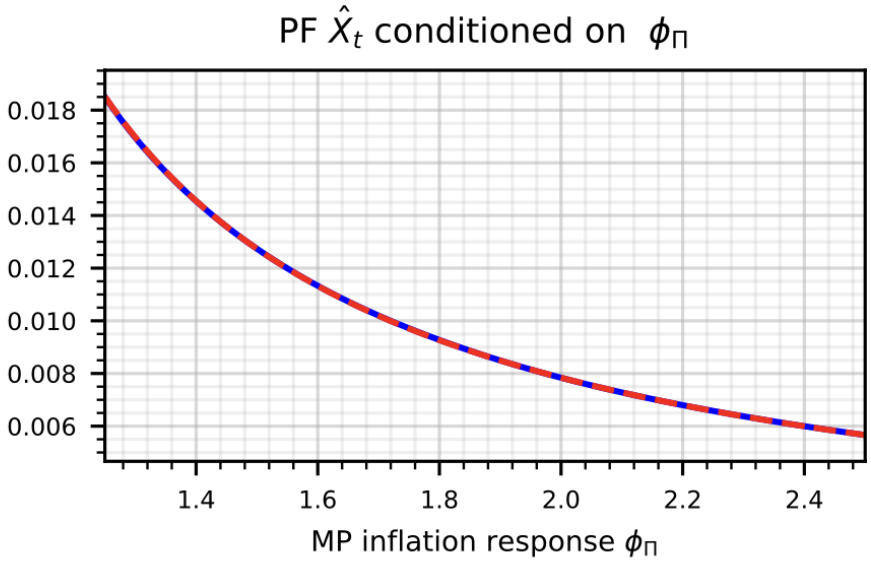

PF $\hat{\Pi}_t$ conditioned on σ_A
0.006
0.005
0.004
0.003
0.002
0.001
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
Standard deviation shock σ_A
Neural network
Analytical solution

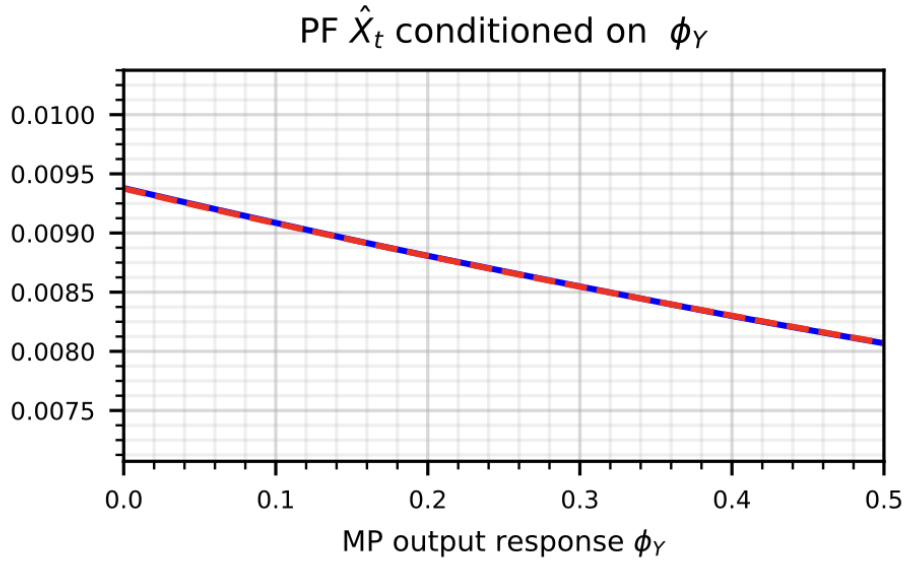

PF $\hat{X}_t$ conditioned on β
0.0095
0.0090
0.0085
0.0080
0.950 0.955 0.960 0.965 0.970 0.975 0.980 0.985 0.990
Discount factor β

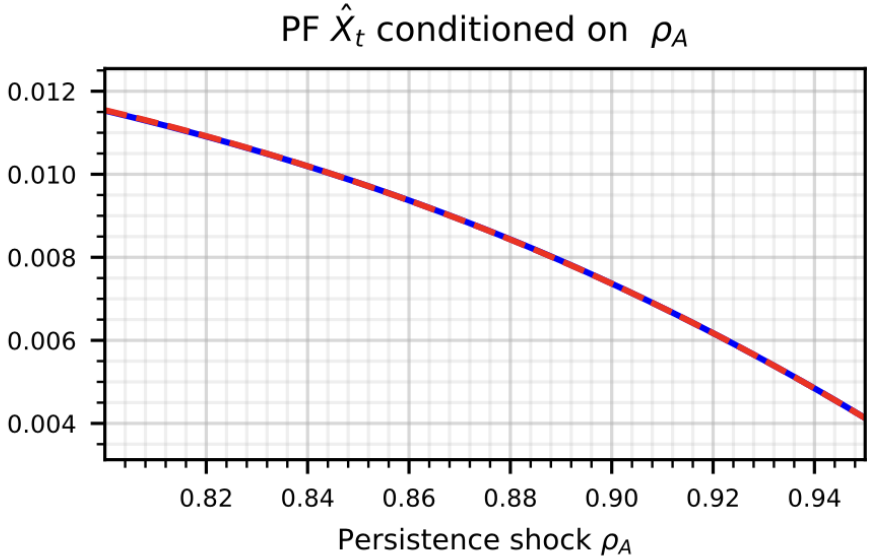

PF $\hat{X}_t$ conditioned on σ
0.010
0.009
0.008
0.007
0.006
1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00
Relative risk aversion σ

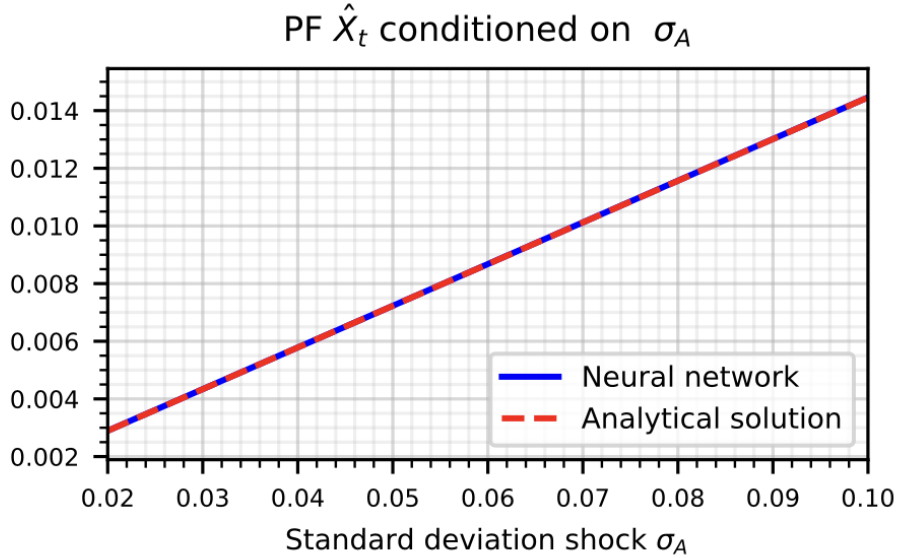

PF $\hat{X}_t$ conditioned on η
0.010
0.009
0.008
0.007
0.006
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
Inverse Frisch elasticity η


PF $\hat{X}_t$ conditioned on ϕ
0.010
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85
Price duration ϕ


PF $\hat{X}_t$ conditioned on ϕ_Π
0.018
0.016
0.014
0.012
0.010
0.008
0.006
1.4
1.6
1.8
2.0
2.2
2.4
MP inflation response ϕ_Π


PF $\hat{X}_t$ conditioned on ϕ_Y
0.0100
0.0095
0.0090
0.0085
0.0080
0.0075
0.0
0.1
0.2
0.3
0.4
0.5
MP output response ϕ_Y


PF $\hat{X}_t$ conditioned on ρ_A
0.012
0.010
0.008
0.006
0.004
0.82
0.84
0.86
0.88
0.90
0.92
0.94
Persistence shock ρ_A


PF $\hat{X}_t$ conditioned on σ_A
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
Standard deviation shock σ_A
Neural network
Analytical solution

Training NN particle filter for Model #1

- There are 10,000 quasi-random parameter draws
- Might be potential issues with overfitting
- 80-20 split (training - validation splits)
- From the 8000 parameter draws (training), choose a batch size of 100 particles. Hence, optimization step is performed 80 times in each epoch.
- There are 20,000 epochs. Therefore, there are 1.6 million steps.

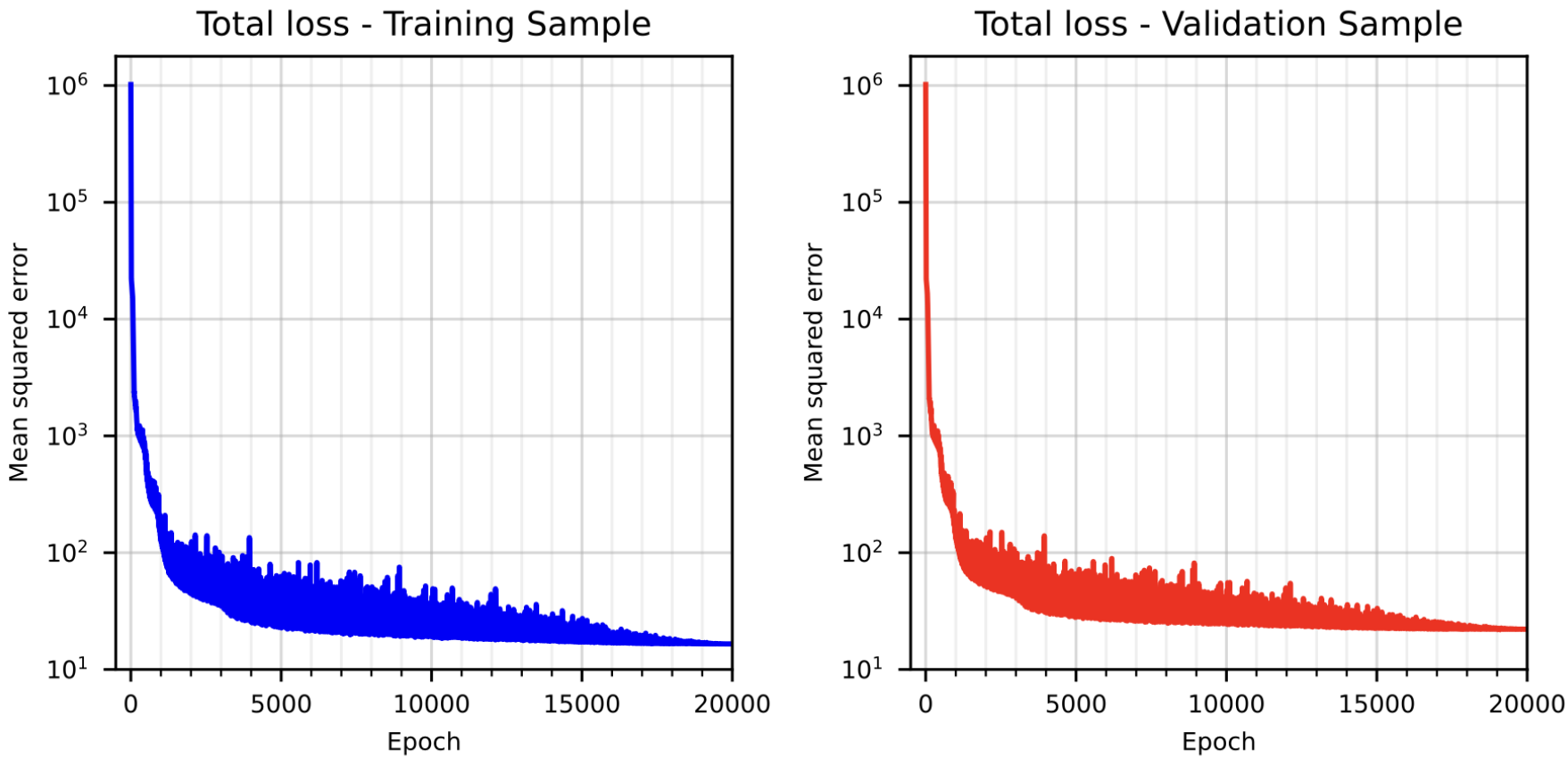


Figure 4: Training and validation convergence. The figure shows the total loss over the training sample (left) and over the validation sample (right). An epoch is completed when all the training or validation sample points are utilized. The vertical axis is expressed in a logarithmic scale.

Dealing with further overfitting issues in Model #1

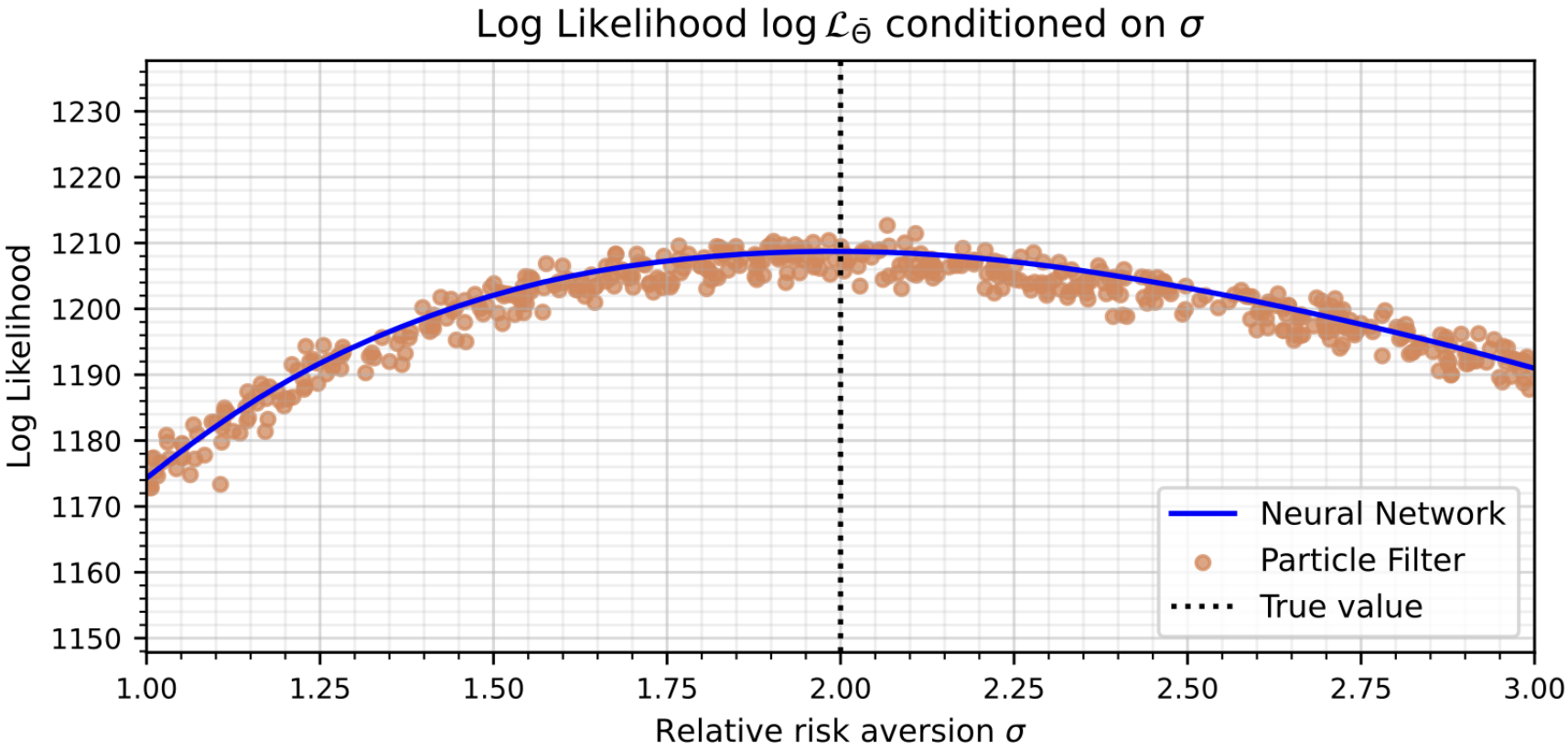
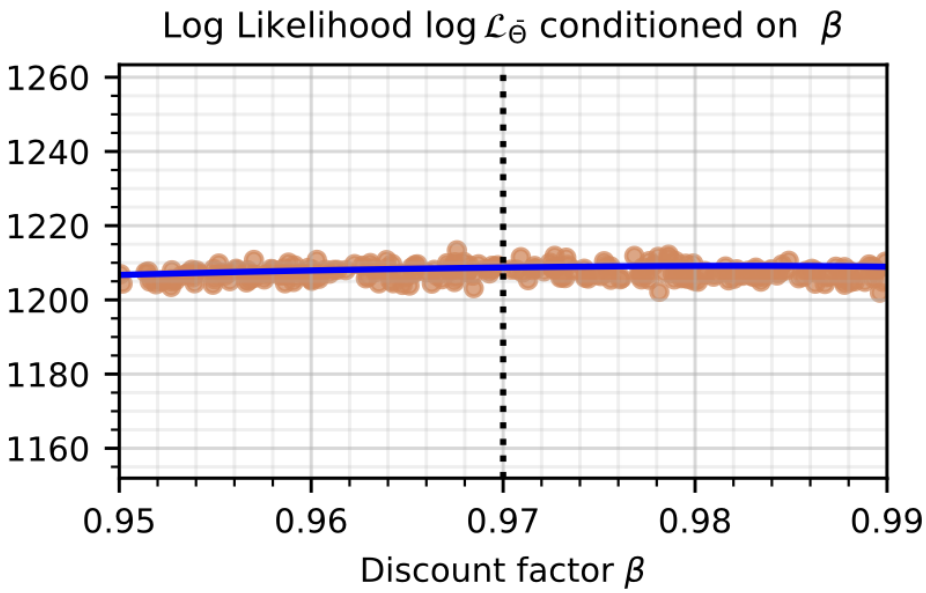
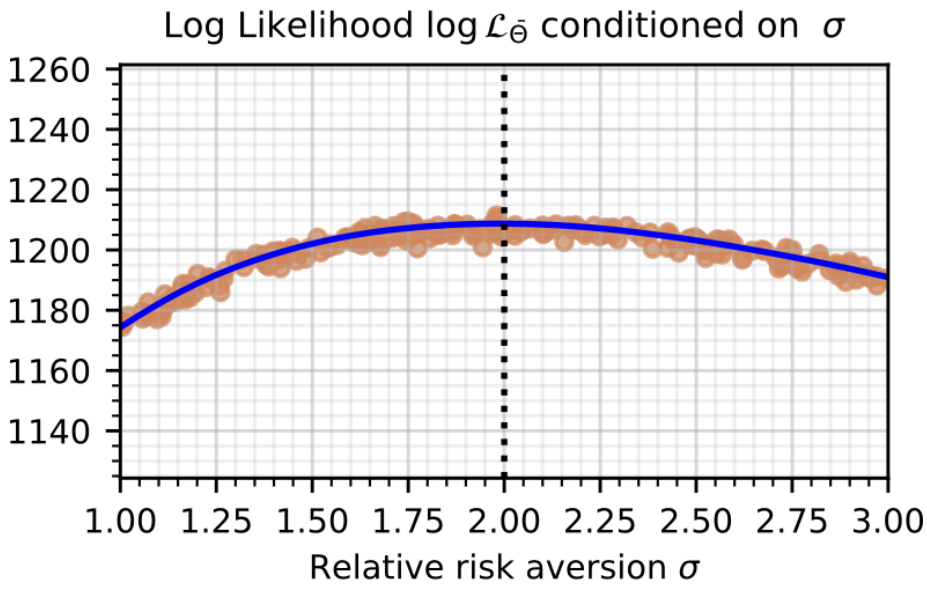
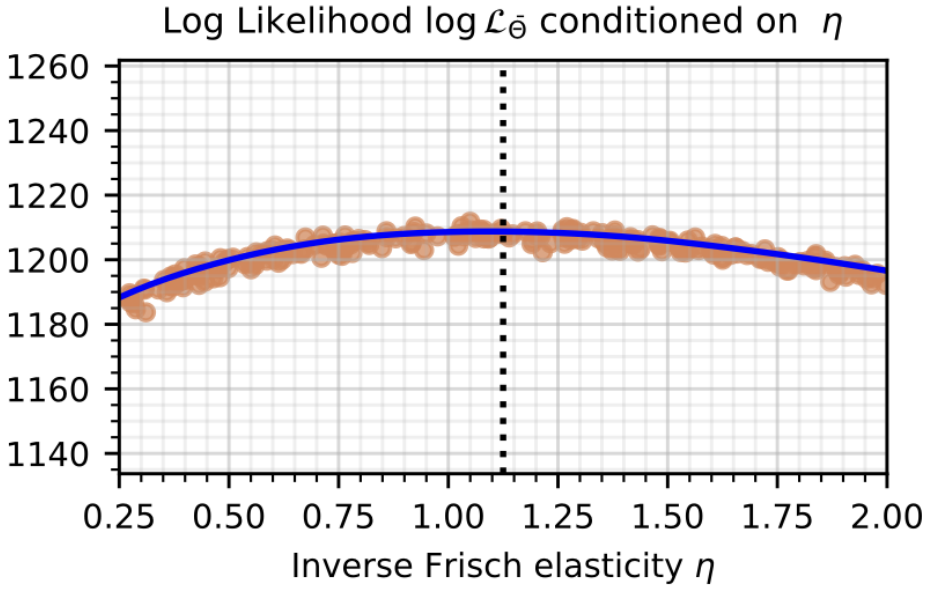
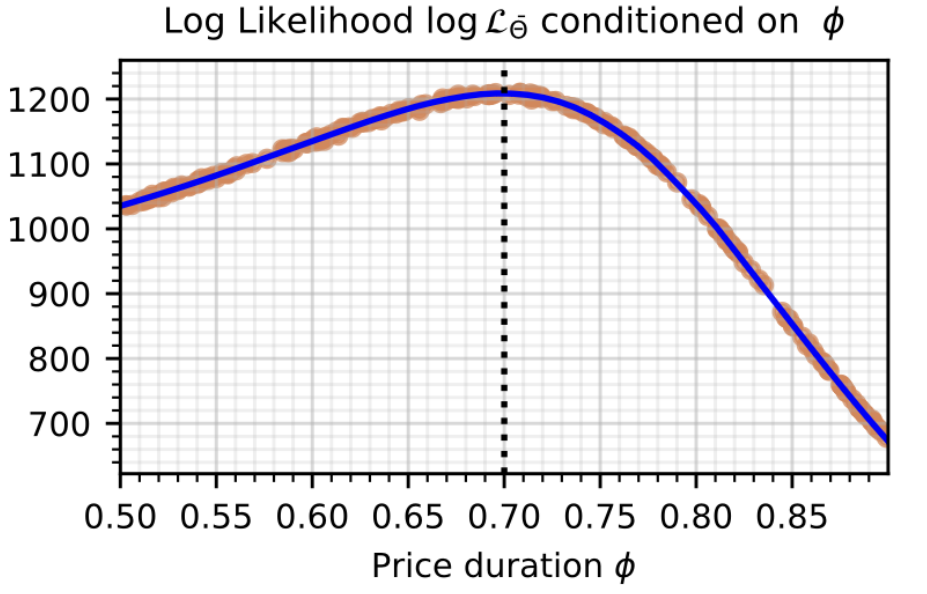


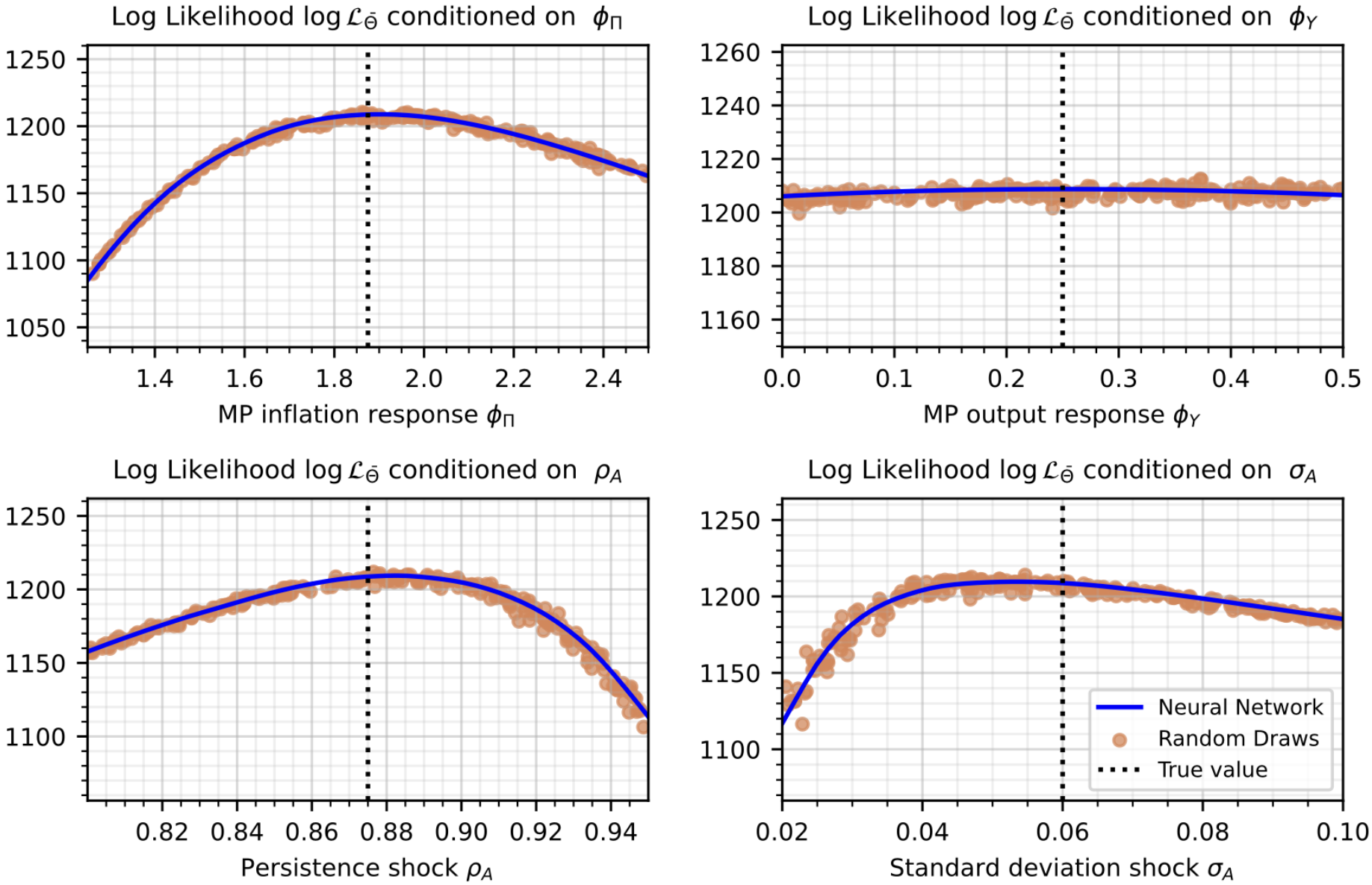
Figure 5: Accuracy in likelihood evaluations: NN particle filter vs. standard particle filter. The logarithm of the likelihood of the model as a function of the risk aversion parameter σ. The value of the fixed parameters is set to the middle of their bounds.


Log Likelihood log $\mathcal{L}_{\bar{\Theta}}$ conditioned on β
1260
1240
1220
1200
1180
1160
0.95
0.96
0.97
0.98
0.99
Discount factor β


Log Likelihood log $\mathcal{L}_{\bar{\Theta}}$ conditioned on σ
1260
1240
1220
1200
1180
1160
1140
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Relative risk aversion σ


Log Likelihood log $\mathcal{L}_{\bar{\Theta}}$ conditioned on η
1260
1240
1220
1200
1180
1160
1140
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Inverse Frisch elasticity η


Log Likelihood log $\mathcal{L}_{\bar{\Theta}}$ conditioned on ϕ
1200
1100
1000
900
800
700
0.50
0.55
0.60
0.65
0.70
0.75
0.80
0.85
Price duration ϕ


Log Likelihood log $\mathcal{L}_{\bar{\Theta}}$ conditioned on ϕ_Π
MP inflation response ϕ_Π
Log Likelihood log $\mathcal{L}_{\bar{\Theta}}$ conditioned on ϕ_Y
MP output response ϕ_Y
Log Likelihood log $\mathcal{L}_{\bar{\Theta}}$ conditioned on ρ_A
Persistence shock ρ_A
Log Likelihood log $\mathcal{L}_{\bar{\Theta}}$ conditioned on σ_A
Standard deviation shock σ_A
Neural Network
Random Draws
True value

What's next?

- Solving a nonlinear RANK model using NN
- Solving a nonlinear HANK model using NN