Replication package for: Estimating Heterogeneous Agent Models with Neural Networks

Hanno Kase, Leonardo Melosi and Matthias Rottner. *Econometrica*.

This README follows the template of the Social Science Data Editors (https://social-science-data-editors.github.io/template_README/).

Overview

The paper solves and estimates five models with neural networks. One folder per model holds the code and the outputs:

- `analytical/`: three-equation New Keynesian model with a closed-form solution
- `rankzlb/`: RANK model with a zero lower bound (ZLB); `rankzlb_ti/` is its time-iteration reference
- `hankmp/`: one-asset HANK model with a monetary-policy shock, compared with the sequence-space Jacobian (SSJ) method
- `hankquant/`: quantitative HANK model, estimated on synthetic data
- `hankzlb/`: one-asset HANK model with a ZLB, estimated on US data

The programs produce every table and figure of the paper and of the Online Appendix. They share the library `estimating_hank_nn/`. The `outputs/` folders hold all trained networks, particle-filter surrogates, likelihood datasets and posterior summaries. The programs load them by default.

The paper's PDF numbers the Online Appendix exhibits Figures 10-20 and Tables 4-6. The stand-alone Online Appendix restarts at Figure 1 and Table 1. The exhibit list at the end gives both numbers.

Data Availability and Provenance Statements

Statement about Rights

- We certify that the author(s) of the manuscript have legitimate access to and permission to use the data used in this manuscript.
- We certify that the author(s) of the manuscript have documented permission to redistribute/publish the data contained within this replication package. Appropriate permission is documented in the LICENSE.txt file.

License for Data

Data and outputs: Creative Commons Attribution 4.0 International (CC BY 4.0). Code: MIT. See LICENSE.txt.

Summary of Availability

- All data **are** publicly available.

Details on each Data Source

The paper uses two small US datasets. Both come from public sources. Both are in the package as CSV files with the script that built them. All other data are simulated by the programs from a fixed seed.

hankzlb/data/data_hankzlb.csv (estimation data of Section 4). Real per-capita GDP growth, GDP-deflator inflation and the effective federal funds rate, quarterly, 1984Q1-2022Q4. The estimation uses 1990Q1-2019Q4. Sources: GDPC1 and GDPDEF (U.S. Bureau of Economic Analysis), CNP16OV (U.S. Bureau of Labor Statistics) and FEDFUNDS (Board of Governors of the Federal Reserve System), all from FRED / ALFRED, Federal Reserve Bank of St. Louis. GDPC1, GDPDEF and CNP16OV are the ALFRED vintage of 2023-04-01. Per-capita output uses an HP-filtered population as in Bilbiie, Primiceri and Tambalotti (2023). The rate is set to zero below 0.2 percent. `hankzlb/data/build_data.py` rebuilds the file (free download, no registration; packages in `requirements-data.txt`). `hankzlb/data/README.md` gives the details. The data are in the public domain.

hankquant/data/fred_data_transformed_BBL.csv (prior predictive check, Figure 17). Quarterly growth rates of per-capita output, consumption, investment, the real wage and hours, GDP-deflator inflation and the federal funds rate, 1954Q1-2019Q4, built as in Bayer, Born and Luetticke (2024). FRED series: GDPC1, PCECC96, GPDIC1, COMPNFB, HOANBS, GDPCTPI, GPDI, PCND, PCDG, PCESV, GCE, GDPDEF, CNP16OV and FEDFUNDS. The file also holds the Wu-Xia (2016) shadow rate of the Federal Reserve Bank of Atlanta (column `nominal_interest_rate_shadow`), which the paper does not use. Fetched on 2025-09-30. `hankquant/data/build_data.py` rebuilds the file (free download, no registration). The FRED series are in the public domain. The shadow rate is freely downloadable.

Dataset list

Data file	Source	Provided
`hankzlb/data/data_hankzlb.csv`	FRED / ALFRED; built by `hankzlb/data/build_data.py`	Yes
`hankzlb/data/data_actual.csv`	Earlier download of the same series; compared with the file above by `hankzlb/data/compare_data_actual.py`	Yes
`hankquant/data/fred_data_transformed_BBL.csv`	FRED and Federal Reserve Bank of Atlanta; built by `hankquant/data/build_data.py`	Yes
`analytical/outputs/pf/synthetic_data.csv`	Simulated by `analytical/main_analytical.py` at the calibration of Table 1, in the run that made the package (GPU); the default run reads it	Yes
`analytical/outputs/pf/pf_points_sigma.csv`, `pf_points_all.csv`	Particle-filter log-likelihoods at random parameter values (the points of Figures 3 and 12), from the same run; the default run reads them	Yes
`hankmp/outputs/baseline/results/*.csv`	Simulated data behind Figures 5, 6, 7 and 16 (policy curves, asset histogram, Euler-error histograms, impulse responses), from the run that made the package; the default run reads them	Yes
`hankzlb/outputs/baseline/results/*.csv`, `tables/moments_comparison.csv`	Simulated moments and the Figure 9 sweep, from the run that made the package; the default run reads them	Yes
`hankquant/outputs/baseline/pf/synthetic_data.csv`	Synthetic dataset of the quantitative HANK estimation, simulated in the run that made the package; the particle-filter dataset was built on it, and the estimation reads it	Yes
`*/outputs/**`	Generated by the programs: reference CSV files, trained networks, particle-filter surrogates, likelihood datasets, posterior summaries, simulation caches	Yes

Computational requirements

Software Requirements

- Python 3.12 (the outputs in the package were made with 3.12.14). `requirements.txt` pins the packages: torch 2.11.0, torchvision 0.26.0, numpy 2.4.4, scipy 1.17.1, numba 0.65.1, matplotlib 3.10.8, pandas 3.0.2, sequence-jacobian 1.0.0, tqdm 4.67.3, ipykernel 7.2.0. Those runs used the CUDA 13.0 build of torch. For a different CUDA driver, install the matching torch wheel from https://pytorch.org first. CPU-only torch works.
- `requirements-data.txt`: packages to re-download the US data. A replication does not need them.
- Linux with bash.

```
conda create -n estimating-hank-nn python=3.12 -y
conda activate estimating-hank-nn
pip install -r requirements.txt
```

Controlled Randomness

Every program seeds torch, numpy and Python's `random` with 1234 through `seed_all` in `estimating_hank_nn/helpers.py`, and sets cuDNN to deterministic mode:

Program	Seed call
`analytical/main_analytical.py`	line 46
`rankzlb/main_rankzlb.py`	line 46
`rankzlb_ti/main_rankzlb_ti_posteriors.py`	line 42 (and line 201 before the RWMH chain)
`hankmp/main_hankmp.py`	line 92
`hankmp/main_hankmp_ssj.py`	no random numbers
`hankzlb/main_hankzlb.py`	line 109
`hankzlb/main_hankzlb_estimation.py`	line 100
`hankquant/main_hankquant.py`	line 138
`hankquant/main_hankquant_pf_dataset.py`	line 32 (passed to the particle filter)
`hankquant/main_hankquant_estimation.py`	line 97
`hankquant/main_hankquant_predictive.py`	lines 83, 189 and 199

`check_reproducibility*.sh` in each model folder runs the `--smoke` pipeline twice and checks that the logs agree exactly. Runs are bit-exact on the CPU, and on a GPU with the same card, driver and torch build. On other hardware, retrained networks reproduce the exhibits up to small numerical differences.

The logs in `*/outputs/**/logs/` are from the default run of `run_all.sh` on the workstation below. A program that loads networks or estimation results from `outputs/` writes `<program>_loaded.log`. A run that recomputes everything writes `<program>.log`.

Memory, Runtime, Storage Requirements

- `./run_all.sh` (default, from the networks, estimation results and simulated data in `outputs/`): about 1 minute on the workstation below, on its GPU or on its CPU.
- `./run_all.sh --smoke`: about 5 minutes on a CPU.
- `./run_all.sh --no-load` (everything from scratch): about 5 days on the RTX 5090 below.
- Storage: 250 MB for the package. A full rerun writes about 1.5 GB more.

The outputs in the package were produced on one workstation: AMD Ryzen 7 9800X3D (8 cores), 64 GB RAM, NVIDIA GeForce RTX 5090 (32 GB), driver 580.178, CUDA 13.0, cuDNN 9.19, Ubuntu 24.04.4 LTS, kernel 7.0.0-38-generic. A GPU is not required. Training and filtering are much slower on a CPU. Times of the runs that produced them:

Program	From scratch	Default run
`analytical/main_analytical.py`	training 242 min; PF dataset 8 min; surrogate 2 min	1 min
`rankzlb_ti/main_rankzlb_ti_posteriors.py` (CPU)	grid posteriors 5 min; RWMH about 9 h	1 min
`rankzlb/main_rankzlb.py`	training 12 + 12 min; PF dataset 42 min; surrogate 2 min; RWMH 1 min	1 min
`hankmp/main_hankmp_ssj.py` (CPU)	1 min	1 min
`hankmp/main_hankmp.py`	DSS 43 min; full model 47 min	1 min
`hankzlb/main_hankzlb.py`	DSS 99 min; full model 207 min; ZLB 206 min	1 min
`hankzlb/main_hankzlb_estimation.py`	PF dataset 577 min; surrogate 1.5 min; RWMH 32 min; counterfactual 2 min	1 min
`hankquant/main_hankquant.py`	DSS 217 min; full model 940 min; Figure 8 sweep about 100 min (`--no-load-errors`)	1 min
`hankquant/main_hankquant_pf_dataset.py`	about 49 h (20 parts in sequence, 17.6 s per likelihood)	loaded
`hankquant/main_hankquant_estimation.py`	surrogate 3 min; RWMH 19 min	1 min
`hankquant/main_hankquant_predictive.py`	49 min (`--no-load`)	1 min

Description of programs/code

```
README.pdf, README.md   this file
LICENSE.txt             MIT (code), CC BY 4.0 (data and outputs)
requirements.txt        pinned Python packages
run_all.sh              master script: ./run_all.sh [--smoke | --no-load]
estimating_hank_nn/     shared library
tests/                  unit tests of the shared library
analytical/             Section 3.1, Online Appendix B
rankzlb_ti/, rankzlb/   Section 3.2, Online Appendix C
hankmp/                 Section 3.3, Online Appendix D
hankquant/              Section 3.4, Online Appendix E
hankzlb/                Section 4, Online Appendix F
```

Each model folder has the same layout and a `README.md` with its files, commands and flags. `main_<model>*.py` are the programs. A configuration block at the top holds the settings; command-line flags override it. `<model>_model.py` holds the model, the networks and the training. `<model>_analysis.py` holds the simulations. `<model>_plots.py` holds the plots. `outputs/` holds the results: `figures/`, `tables/` (LaTeX and CSV), `logs/`, `results/`, `networks/` and `pf/`. `--smoke` runs write to `outputs/smoke/` only.

- `analytical/main_analytical.py`: policy network, comparison with the closed form, particle filter, likelihood slices.
- `rankzlb_ti/main_rankzlb_ti_posteriors.py`: time-iteration solution and bootstrap particle-filter estimation; writes the reference CSV files that `rankzlb/` reads.
- `rankzlb/main_rankzlb.py`: RANK networks without and with the ZLB, synthetic data, particle filter, RWMH.
- `hankmp/main_hankmp_ssj.py`: SSJ reference. `hankmp/main_hankmp.py`: neural-network solution and the comparison.
- `hankquant/main_hankquant.py`: solution and Figure 8. `main_hankquant_pf_dataset.py`: likelihood dataset. `main_hankquant_estimation.py`: surrogate, RWMH, Table 2. `main_hankquant_predictive.py`:

prior predictive check.
- `hankzlb/main_hankzlb.py`: solution in three phases. `main_hankzlb_estimation.py`: likelihood dataset, surrogate, RWMH, counterfactual, moments table.

License for Code

MIT. See LICENSE.txt.

Instructions to Replicators

1. Install the software (see "Software Requirements").
2. Unzip the package. Activate the environment. In the top folder, run:

```
./run_all.sh            # 1 min: every exhibit from the networks, estimation results and
simulated data in outputs/
```

3. Compare the regenerated `*/outputs/**/figures/*.pdf` and `tables/*.tex` with the files in the zip. The files in the zip are the ones in the paper.

Notes:

- This default run loads the trained networks and the estimation results (the `load_*` flags in each configuration block are `True`) and redoes the evaluation, the simulations and the plots. Figure 8 and Figure 17 come from cached simulation sweeps in `hankquant/outputs/baseline/results/`. Each cache file is headed by the SHA-256 hash of its checkpoint, which the program checks. `main_hankquant.py --no-load-errors` and `main_hankquant_predictive.py --no-load` rerun the sweeps.
- `./run_all.sh --smoke` trains tiny networks on the CPU and checks that the code runs. Its numbers are not those of the paper. It loads no trained networks or estimation results, with three exceptions: `hankmp/main_hankmp.py` reads the SSJ reference in `hankmp/outputs/reference/`, `rankzlb/main_rankzlb.py` reads the time-iteration reference in `rankzlb_ti/outputs/reference/`, and the three quantitative HANK estimation programs read the solution checkpoint `hankquant/outputs/baseline/networks/full.pt`.
- `./run_all.sh --no-load` retrains and re-estimates everything (about 5 days on the RTX 5090). It runs the programs in dependency order. It reuses the 20 parts of the quantitative HANK likelihood dataset in `hankquant/outputs/baseline/pf/datasets/` (about 49 GPU hours to regenerate). The RWMH chains of `hankzlb` and `hankquant` are stored as summaries in `pf/rwmh_results.pt`. A sampling run also writes the full draws to `pf/rwmh_draws.pt`, which is not part of the package.
- Pass `--device cpu` to run a program on the CPU. Every model folder runs on its own with `../estimating_hank_nn/` and the reference folders.
- A rerun writes PNG and JPG copies next to the PDF figures. The paper typesets the LaTeX tables in its own layout.

List of tables and programs

The code reproduces all numbers in the paper and in the Online Appendix except Figure 1 (an illustration of a neural network) and the data Gini coefficient in Table 6, which is typed into the paper's table. Paths are relative to the model folder. "OA" is the number in the stand-alone Online Appendix.

Exhibit (paper)	OA	Program (line)	Output file
Figure 1		n.a.	
Table 1		analytical/main_analytical.py (167)	analytical/outputs/tables/parameters.tex (.csv)
Figure 2		analytical/main_analytical.py (280)	analytical/outputs/figures/extended_pf_X.pdf
Figure 3		analytical/main_analytical.py (403)	analytical/outputs/figures/NN_likelihood_sigma.pdf
Figure 4		rankzlb/main_rankzlb.py (369)	rankzlb/outputs/figures/NN_posterior_all.pdf
Figure 5		hankmp/main_hankmp.py (397)	hankmp/outputs/baseline/figures/dss_asset_distribution.pdf
Figure 6		hankmp/main_hankmp.py (375)	hankmp/outputs/baseline/figures/dss_hh_policy.pdf
Figure 7		hankmp/main_hankmp.py (452)	hankmp/outputs/baseline/figures/irf_nn_3col_ssj_linear.pdf
Figure 8		hankquant/main_hankquant.py (451)	hankquant/outputs/baseline/figures/full_euler_errors_param_draws.pdf
Table 2		hankquant/main_hankquant_estimation.py (358)	hankquant/outputs/baseline/tables/posterior_summary.tex (.csv)
Table 3		hankzlb/main_hankzlb_estimation.py (346)	hankzlb/outputs/baseline/tables/posterior_summary.tex (.csv)
Figure 9		hankzlb/main_hankzlb_estimation.py (394)	hankzlb/outputs/baseline/figures/volatility_zlb_frequency.pdf
Figure 10	Fig. 1	analytical/main_analytical.py (251)	analytical/outputs/figures/model_loss.pdf
Figure 11	Fig. 2	analytical/main_analytical.py (267)	analytical/outputs/figures/extended_pf_Pi.pdf
Figure 12	Fig. 3	analytical/main_analytical.py (438)	analytical/outputs/figures/NN_likelihood_all.pdf
Figure 13	Fig. 4	analytical/main_analytical.py (378)	analytical/outputs/figures/pf_loss.pdf
Table 4	Tab. 1	rankzlb/main_rankzlb.py (478-480)	rankzlb/outputs/tables/TableRANK.tex (.csv)
Figure 14	Fig. 5	rankzlb/main_rankzlb.py (253)	rankzlb/outputs/figures/model_zlb_loss.pdf
Figure 15	Fig. 6	hankmp/main_hankmp.py (334)	hankmp/outputs/baseline/figures/full_loss_3col.pdf
Figure 16	Fig. 7	hankmp/main_hankmp.py (424)	hankmp/outputs/baseline/figures/full_euler_errors.pdf
Figure 17	Fig. 8	hankquant/main_hankquant_predictive.py (219)	hankquant/outputs/baseline/figures/prior_predictive_pairwise_hist2d_std_uniform.pdf
Figure 18	Fig. 9	hankquant/main_hankquant_estimation.py (269)	hankquant/outputs/baseline/figures/pf_loss.pdf
Table 5	Tab. 2	hankzlb/main_hankzlb.py (231)	hankzlb/outputs/baseline/tables/TableHANKZLB.tex (.csv)
Figure 19	Fig. 10	hankzlb/main_hankzlb.py (379)	hankzlb/outputs/baseline/figures/dss_avg_loss.pdf

Exhibit (paper)	OA	Program (line)	Output file
Figure 20	Fig. 11	`hankzlb/main_hankzlb.py` (384)	`hankzlb/outputs/baseline/figures/zlb_avg_loss.pdf`
Table 6	Tab. 3	`hankzlb/main_hankzlb_estimation.py` (423)	`hankzlb/outputs/baseline/tables/moments_comparison.tex` (`.csv`)

Notes:

- Exhibits that are simulated: the CPU and the GPU random generators give different draws for the same seed, so the simulated data behind these exhibits come from the run that made the package, and a run that loads the trained networks and estimation results reads them; `--no-load` and `--smoke` simulate and write them. The files: Figures 3 and 12, the synthetic dataset `analytical/outputs/pf/synthetic_data.csv` and the particle-filter points `pf_points_sigma.csv` and `pf_points_all.csv`; Figures 5, 6, 7 and 16, `hankmp/outputs/baseline/results/*.csv`; Figure 9 and the model moments of Tables 5 and 6, `hankzlb/outputs/baseline/results/*.csv` and `tables/moments_comparison.csv`; the synthetic dataset of the quantitative HANK estimation, `hankquant/outputs/baseline/pf/synthetic_data.csv`.
- Tables 2 and 3: the standard deviations of the shocks are in percent (x100), as in the paper; the `Param` column names them `100*sigma_...`.
- The CSV files use a period as the decimal separator and a comma as the field separator.
- Table 4: `TableRANK.tex` is the lower panel. The upper panel lists the calibration in the configuration block of `rankzlb/main_rankzlb.py`.
- Table 5: the "Target/Source" column of the paper is text. The calibration targets (ZLB frequency, share of borrowers) are in `hankzlb/outputs/baseline/results/moments_calibration.csv` (line 401 of `hankzlb/main_hankzlb.py`).
- Table 6: the code computes the model column. The data Gini is typed into the paper's table.
- Figure 4 overlays `rankzlb_ti/outputs/reference/ti_posterior_*.csv`. Figures 5-7 overlay `hankmp/outputs/reference/`. Figures 8 and 17 are drawn from the caches in `hankquant/outputs/baseline/results/`. Figure 9 is drawn from `hankzlb/outputs/baseline/results/volatility_zlb_frequency.csv`.

Numbers in the text: the calibration of the one-asset HANK model (Online Appendix D) is in `hankmp/outputs/baseline/tables/TableHANKMP.csv`, and that of the quantitative HANK model in `hankquant/outputs/baseline/tables/TableHANKQUANT.csv`. The posterior medians of Section 4 are the `Post_Median` column of `hankzlb/outputs/baseline/tables/posterior_summary.csv`. The training settings of Online Appendix G are the configuration blocks of the programs. The execution times quoted in the paper are the "From scratch" times in the runtime table above.

References

Auclert, A., B. Bardoczy, M. Rognlie and L. Straub (2021). "Using the Sequence-Space Jacobian to Solve and Estimate Heterogeneous-Agent Models." *Econometrica* 89(5), 2375-2408. Software: `sequence-jacobian` 1.0.0, https://github.com/shade-econ/sequence-jacobian.

Bayer, C., B. Born and R. Luetticke (2024). "Shocks, Frictions, and Inequality in US Business Cycles." *American Economic Review* 114(5), 1211-1247.

Bilbiie, F. O., G. E. Primiceri and A. Tambalotti (2023). "Inequality and Business Cycles." NBER Working Paper 31729.

Board of Governors of the Federal Reserve System (US). Effective Federal Funds Rate [FEDFUNDS]. Retrieved from FRED, Federal Reserve Bank of St. Louis, https://fred.stlouisfed.org/series/FEDFUNDS.

Federal Reserve Bank of Atlanta. Wu-Xia Shadow Federal Funds Rate. Retrieved from https://www.atlantafed.org/cqer/research/wu-xia-shadow-federal-funds-rate.

U.S. Bureau of Economic Analysis. Real Gross Domestic Product [GDPC1]; Gross Domestic Product: Implicit Price Deflator [GDPDEF]; Gross Domestic Product: Chain-type Price Index [GDPCTPI]; Real Personal Consumption Expenditures [PCECC96]; Real Gross Private Domestic Investment [GPDIC1]; Gross Private Domestic Investment [GPDI]; Personal Consumption Expenditures: Nondurable Goods [PCND], Durable Goods [PCDG], Services [PCESV]; Government Consumption Expenditures and Gross Investment [GCE]. Retrieved from FRED / ALFRED, Federal Reserve Bank of St. Louis, https://fred.stlouisfed.org/ (ALFRED vintage 2023-04-01 for GDPC1, GDPDEF and CNP16OV in `hankzlb/data/`).

U.S. Bureau of Labor Statistics. Population Level [CNP16OV]; Nonfarm Business Sector: Hourly Compensation for All Workers [COMPNFB]; Nonfarm Business Sector: Hours Worked for All Workers [HOANBS]. Retrieved from FRED / ALFRED, Federal Reserve Bank of St. Louis, https://fred.stlouisfed.org/.

Wu, J. C. and F. D. Xia (2016). "Measuring the Macroeconomic Impact of Monetary Policy at the Zero Lower Bound." *Journal of Money, Credit and Banking* 48(2-3), 253-291.