Analysis Methodology for the RBC Production Network Model

DEQN Framework

November 30, 2025

1 Introduction

This document describes the analysis methodology applied to the RBC Production Network model (`RbcProdNet_nonlinear`). The model features 37 sectors with CES production functions, intermediate input linkages, investment flows, and correlated productivity shocks. The analysis framework evaluates trained neural network solutions across the following dimensions: ergodic simulations, welfare costs, stochastic steady states, and generalized impulse responses.

2 Model Overview

2.1 State Variables

The model has $2N$ state variables where $N = 37$ is the number of sectors:

- K_i (states 0 to $N-1$): Capital stock in sector i (in logs). Capital evolves deterministically according to the law of motion with investment.
- a_i (states N to $2N-1$): Productivity level in sector i (in logs, deviation from steady state). This is the only source of uncertainty in the model, following an AR(1) process with correlated innovations:

$$a_{i,t+1} = \rho \cdot a_{i,t} + \varepsilon_{i,t+1}, \quad \varepsilon_t \sim \mathcal{N}(0, \Sigma_A) \tag{1}$$

2.2 Key Parameters

- α: Capital share in production
- β: Discount factor
- δ: Depreciation rate
- ρ: Persistence of productivity shocks
- ϕ: Investment adjustment cost parameter
- $\sigma_c, \sigma_m, \sigma_q, \sigma_y, \sigma_I, \sigma_l$: Elasticities of substitution
- Γ_M, Γ_I: Input-output matrices for intermediates and investment
- Σ_A: Covariance matrix of productivity shocks

2.3 Policy Variables

The neural network outputs $11N + 5$ policy variables:

1. C_i: Sectoral consumption
2. L_i: Sectoral labor
3. P_i^k: Capital good prices
4. P_i^m: Intermediate input prices
5. M_i: Intermediate inputs used
6. M_i^{out}: Intermediate outputs sold
7. I_i: Investment
8. I_i^{out}: Investment goods sold
9. P_i: Output prices
10. Q_i: Gross output
11. Y_i: Value added
12. $C^{agg}, L^{agg}, Y^{agg}, I^{agg}, M^{agg}$: Aggregate variables

3 Analysis Components

3.1 Ergodic Simulation Analysis

The simulation analysis generates long trajectories from the trained model to characterize the ergodic distribution of economic outcomes.

3.1.1 Configuration

- **Episode length**: 64,000 periods (configurable via `periods_per_epis`)
- **Burn-in**: 3,200 periods discarded to ensure convergence to ergodic distribution
- **Number of seeds**: 16 independent simulations for robustness
- **Volatility scale**: Adjustable shock magnitude for comparative analysis

3.1.2 Analysis Variables

For each simulation, we compute the following aggregate variables as log deviations from the deterministic steady state:

$$\hat{C}_t^{agg} = \log(C_t^{agg}) - \log(C_{ss}^{agg}) \tag{2}$$
$$\hat{L}_t^{agg} = \log(L_t^{agg}) - \log(L_{ss}^{agg}) \tag{3}$$
$$\hat{K}_t^{agg} = \log(K_t^{agg}) - \log(K_{ss}^{agg}) \tag{4}$$
$$\hat{Y}_t^{agg} = \log(Y_t^{agg}) - \log(Y_{ss}^{agg}) \tag{5}$$
$$\hat{M}_t^{agg} = \log(M_t^{agg}) - \log(M_{ss}^{agg}) \tag{6}$$
$$\hat{I}_t^{agg} = \log(I_t^{agg}) - \log(I_{ss}^{agg}) \tag{7}$$

Aggregate variables are computed using price weights from the average simulation prices:

$$K_t^{agg} = \sum_{i=1}^{N} K_{it} \cdot \bar{P}_i^k \tag{8}$$

3.1.3 Descriptive Statistics

For each analysis variable, we report:

- Mean (as percentage deviation from steady state)
- Standard deviation (as percentage)
- Skewness
- Excess kurtosis

3.2 Welfare Analysis

3.2.1 Utility Function

The representative household's period utility is:

$$U_t = \frac{1}{1-\varepsilon_c^{-1}} \left(C_t^{agg} - \theta \frac{1}{1+\varepsilon_l^{-1}} (L_t^{agg})^{1+\varepsilon_l^{-1}} \right)^{1-\varepsilon_c^{-1}} \tag{9}$$

3.2.2 Welfare Cost Calculation

The welfare cost measures the consumption-equivalent loss from business cycle fluctuations:

1. Compute steady-state welfare:
$$W_{ss} = \frac{U_{ss}}{1-\beta} \tag{10}$$

2. Estimate stochastic welfare from simulation trajectories using multiple random starting points:
$$W = \mathbb{E}\left[\sum_{t=0}^{T} \beta^t U_t\right] \tag{11}$$

3. Welfare loss (in percentage):

$$\text{Welfare Loss} = \left(1 - \frac{W}{W_{ss}}\right) \times 100 \tag{12}$$

3.2.3 Configuration

- **Number of trajectories**: 16,000
- **Trajectory length**: 100 periods
- **Random seed**: Configurable for reproducibility

3.3 Stochastic Steady State

The stochastic steady state represents the resting point of the system when the shocks are zero.

3.3.1 Methodology

1. Sample n points from the ergodic distribution (default: 2,000 draws)
2. For each sampled state, simulate forward for T periods (default: 500) with zero shocks
3. Average the final states and policies across all draws
4. Verify convergence by checking that the standard deviation across draws is small

3.3.2 Convergence Check

If $\max(\sigma_{\text{stoch\_ss}}) > 0.001$, an error is raised indicating insufficient convergence.

3.4 Generalized Impulse Responses (GIRs)

GIRs measure the dynamic response of the economy to a shock, accounting for the nonlinearity of the model.

3.4.1 Methodology

The GIR procedure follows these steps:

1. **Sample from ergodic distribution**: Draw n initial states from the simulation (default: 1,000 draws)
2. **Generate shock trajectories**: For each initial state s_0, generate a sequence of future shocks $\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_T\}$
3. **Simulate baseline trajectory**: Starting from s_0, simulate forward using the shock sequence
4. **Create counterfactual state**: Apply an impulse to the productivity of sector i:

$$s_0^{cf} = s_0 + \Delta \quad \text{where} \tag{13}$$

$$\Delta_j = \begin{cases} \log(1 \mp \text{shock\_size}) & \text{if } j = N + i \text{ (productivity of sector } i\text{)} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

5. **Simulate counterfactual trajectory**: Starting from s_0^{cf}, simulate forward using the *same* shock sequence
6. **Compute impulse response**: The GIR is the difference between counterfactual and baseline paths:
$$\text{GIR}_t = x_t^{cf} - x_t^{baseline} \tag{15}$$
7. **Average across draws**: The final GIR averages over all sampled initial conditions

3.4.2 Shock Configuration

The model features productivity (TFP) shocks only. Capital evolves deterministically according to the investment decision. For the GIR analysis, we shock the productivity states:

- **Productivity shocks**: State index $= N + i$ for sector i (i.e., states 37 to 73 for $N = 37$ sectors)
- **Shock sizes**: 5%, 10%, 20% (configurable)
- **Shock signs**: Both positive and negative
- **Trajectory length**: 100 periods (configurable)

3.4.3 Shock Implementation

The shock is applied multiplicatively to TFP in log space. For a negative x% shock:
$$A^{cf} = A^{current} \times (1 - x/100) \quad \Rightarrow \quad \log(A^{cf}) = \log(A^{current}) + \log(1 - x/100) \tag{16}$$

Specifically:

- 5% negative shock: $\log(A^{cf}) = \log(A^{current}) + \log(0.95) \approx \log(A^{current}) - 0.051$
- 10% negative shock: $\log(A^{cf}) = \log(A^{current}) + \log(0.90) \approx \log(A^{current}) - 0.105$
- 20% negative shock: $\log(A^{cf}) = \log(A^{current}) + \log(0.80) \approx \log(A^{current}) - 0.223$

For positive shocks, the signs are reversed (e.g., $+\log(1.20) \approx +0.182$ for a 20% positive shock).

3.4.4 Comparison with Log-Linear IRs

When available, GIRs are compared with impulse responses from log-linearized MATLAB/Dynare solutions (perfect foresight paths). This comparison:

- Validates the nonlinear solution against standard benchmarks
- Quantifies the importance of nonlinearities for different shock sizes
- Identifies asymmetries between positive and negative productivity shocks

4 MATLAB Benchmark Calculations

The MATLAB/Dynare code provides benchmark impulse responses for comparison with the neural network solution. This section documents the calibration, steady state computation, and IR calculation procedures.

4.1 Calibration Procedure

The model is calibrated using U.S. sectoral data from the BEA. The calibration proceeds in stages:

4.1.1 Data Sources

- **Consumption shares** (ξ): Average sectoral consumption expenditure shares from BEA
- **Capital shares** (α): Average capital expenditure shares by sector (floored at 5%)
- **Value-added shares** (μ): Average ratio of value added to gross output
- **Input-output matrix** (Γ_M): Average IO flows, normalized with entries floored at 1%
- **Investment matrix** (Γ_I): Average investment flows across sectors
- **Depreciation rates** (δ): Sector-specific rates from BEA fixed assets
- **TFP process**: Estimated AR(1) process with covariance matrix Σ_A

4.1.2 Steady State Computation

The steady state is solved iteratively, starting from a Cobb-Douglas specification and gradually lowering elasticities to target values:

1. **Stage 1**: Solve with all $\sigma = 0.99$ (near Cobb-Douglas)
2. **Stage 2**: Lower σ_l to target value while matching labor reallocation
3. **Stage 3**: Lower σ_m, σ_q while matching value-added and IO shares
4. **Stage 4**: Lower $\sigma_c, \sigma_y, \sigma_I$ while matching all expenditure shares

The steady state solver (`ProdNetRbc_SS.m`) jointly determines:

- Sectoral quantities: $C_i, L_i, M_i, I_i, Q_i, Y_i$
- Sectoral prices: P_i, P_i^k, P_i^m
- Aggregates: C^{agg}, L^{agg}
- Preference parameter: θ (calibrated to normalize marginal utility)

4.2 Log-Linear Solution (Dynare)

The log-linearized model is solved using Dynare (`stoch_simul.mod`):

1. Variables are expressed as log deviations from steady state
2. First-order perturbation around the deterministic steady state
3. Shocks are specified with the estimated covariance matrix Σ_A
4. Policy functions are linear: $x_t = C \cdot s_{t-1} + D \cdot \varepsilon_t$

The solution yields state-space matrices (A, B, C, D) where:

$$s_t = A \cdot s_{t-1} + B \cdot \varepsilon_t \quad \text{(state evolution)} \tag{17}$$

$$x_t = C \cdot s_{t-1} + D \cdot \varepsilon_t \quad \text{(policy functions)} \tag{18}$$

4.3 Impulse Response Computation

Two types of impulse responses are computed for each sector:

4.3.1 Log-Linear IRs

Starting from steady state, apply a TFP shock to sector i and simulate using the linear policy functions:

1. Set initial state: $a_i(0) = \text{shock\_size}$ (e.g., $\log(0.95)$ for -5%)
2. Simulate forward with zero future shocks: $\varepsilon_t = 0$ for $t > 0$
3. Record deviations from steady state for all variables

4.3.2 Perfect Foresight (Deterministic) IRs

Solve the fully nonlinear model under perfect foresight (`determ_irs.mod`):

1. Set initial TFP: $a_i(0) = \text{shock\_size}$
2. Solve for the perfect foresight transition path back to steady state
3. No future uncertainty: agents know the exact path of TFP recovery

This captures nonlinearities in the transition dynamics but ignores precautionary behavior.

4.4 Units and Normalization

Critical for comparison: All impulse responses are stored in **log deviations from steady state** (not percentages):

- **MATLAB output** (`ProcessIRs.m`): Variables are computed as

$$\hat{x}_t = \log(x_t) - \log(x_{ss}) \tag{19}$$

- **JAX GIR output**: Also in log deviations from steady state
- **Plotting**: Both are multiplied by 100 to display as percentages

4.4.1 MATLAB IR Variable Mapping

The `ProcessIRs.m` function outputs a matrix with rows corresponding to:

Important: Row 0 (TFP) is in **levels**, not log deviations. For a -20% shock, $A_i(0) = 0.8$. All other rows are in log deviations from steady state.

Note: The Python loader (`matlab_irs.py`) contains a variable index mapping that should be verified against the actual saved `.mat` files using the `inspect_matlab_ir_structure()` function.

Row	Variable	Description	Units
0	A_i	TFP of shocked sector	**Levels** (not log dev)
1	$\hat{C}^{agg}$	Aggregate consumption	Log deviation
2	$\hat{L}^{agg}$	Aggregate labor	Log deviation
3	V_c	Continuation value	Level
4	$\hat{C}_i$	Sectoral consumption	Log deviation
5	$\hat{P}_i$	Output price	Log deviation
6	$\hat{I}_i^{out}$	Investment outputs sold	Log deviation
7	$\hat{M}_i^{out}$	Intermediate outputs sold	Log deviation
8	$\hat{L}_i$	Sectoral labor	Log deviation
9	$\hat{I}_i$	Sectoral investment	Log deviation
10	$\hat{M}_i$	Intermediate inputs	Log deviation
11	$\hat{Y}_i$	Sectoral value added	Log deviation
12	$\hat{Q}_i$	Gross output	Log deviation
13–22	Client	Client sector variables	(same ordering)
23	$\hat{K}_i$	Capital of shocked sector	Log deviation
24	$\hat{Y}^{agg}$	Aggregate output	Log deviation
25	$\hat{P}^m_{client}$	Intermediate price of client	Log deviation
26	$\hat{\gamma}_{ij}$	Expenditure share change	Log deviation

Table 1: MATLAB IR output variable ordering from `ProcessIRs.m`

4.4.2 Shock Sign Convention

In the MATLAB code, shocks are defined with the following convention:

- `params.IRshock = -0.0513`: This is $\log(0.95) \approx -0.0513$
- For a **positive** 5% shock (TFP increases to 105%): set $a_i(0) = -\text{IRshock} = +0.0513$
- For a **negative** 5% shock (TFP decreases to 95%): set $a_i(0) = +\text{IRshock} = -0.0513$

This matches the JAX convention where a negative shock adds $\log(1 - \text{shock\_size})$ to the log level.

4.5 Amplification Metrics

The MATLAB code computes several diagnostic metrics:

- **Peak values**: Maximum absolute response for each variable
- **Peak periods**: Time at which peak response occurs
- **Half-lives**: Time for response to decay to half of peak
- **Amplification**: Difference between perfect foresight and log-linear peaks

$$\text{Amplification}_i = \text{peak}_i^{determ} - \text{peak}_i^{loglin} \tag{20}$$

Positive amplification indicates that nonlinearities magnify the response relative to the log-linear approximation.

5 Comparative Analysis

The framework supports analyzing multiple trained models (experiments) simultaneously, enabling:

5.1 Volatility Comparative

Compare solutions trained under different shock volatilities:

- Baseline volatility
- Scaled volatility (e.g., $0.1\times$, $0.5\times$, $1.5\times$ baseline)

This reveals how the optimal policy functions change with the magnitude of uncertainty.

5.2 Architecture Comparative

Compare solutions from different neural network architectures or training configurations.

6 Output Files

The analysis generates the following outputs in the analysis directory:

6.1 Configuration

- `config.json`: Full analysis configuration for reproducibility

6.2 Tables (LaTeX)

- `descriptive_stats_table.tex`: Descriptive statistics for each experiment
- `descriptive_stats_comparative.tex`: Side-by-side comparison across experiments
- `welfare_table.tex`: Welfare costs for each experiment
- `stochastic_ss_table.tex`: Stochastic steady state values

6.3 Figures

- `simulation/`: Ergodic histograms and sectoral plots
- `IRs/`: Impulse response comparisons (JAX GIRs vs MATLAB)

7 Upstreamness Measures

The model also computes sectoral upstreamness measures following the production network literature:

7.1 Intermediate Input Upstreamness (U^M)

$$U^M = (I - \Delta^M)^{-1}\mathbf{1} \tag{21}$$

where Δ^M captures the intermediate input linkages weighted by prices and quantities.

7.2 Investment Flow Upstreamness (U^I)

$$U^I = (I - \Delta^I)^{-1}\mathbf{1} \tag{22}$$

where Δ^I captures the investment good linkages.

7.3 Simple Upstreamness

$$U_i^{simple} = \frac{M_i^{out}}{Q_i} \tag{23}$$

The ratio of intermediate outputs to gross output for each sector.

8 Implementation Notes

8.1 Normalization

All state and policy variables are internally normalized by their steady-state standard deviations for numerical stability during neural network training and evaluation.

8.2 Precision

The analysis supports both single (float32) and double (float64) precision. Double precision is recommended for welfare and GIR calculations.

8.3 JIT Compilation

All analysis functions are JIT-compiled using JAX for efficient execution on CPU, GPU, or TPU.