Three-Equation New Keynesian Model

Based on Galí (2015), Chapter 3

Implementation in JAXecon/DEQN

---

1. Introduction

The three-equation New Keynesian (NK) model is the workhorse framework for monetary policy analysis. It combines optimizing behavior by households and firms with nominal rigidities (sticky prices) to generate a role for monetary policy.

The model consists of three key equations:

- Dynamic IS Equation (demand side)
- New Keynesian Phillips Curve (supply side)
- Monetary Policy Rule (Taylor rule)

Unlike Real Business Cycle models, the NK model is purely forward-looking with no endogenous state variables. The only state variables are exogenous shock processes.

Key Features:

- Forward-looking expectations (rational expectations)
- Nominal price rigidities (Calvo pricing)
- Monetary policy affects real variables in the short run
- Divine coincidence: stabilizing inflation also stabilizes output gap
- Steady state: zero inflation, zero output gap

2. The Three Equations

2.1 New Keynesian Phillips Curve (NKPC)

$$\pi_t = \beta\,\mathbb{E}_t\{\pi_{t+1}\} + \kappa\,\tilde{y}_t$$

(Galí Equation 21)

where:

- π_t = inflation rate (log deviation from steady state)
- $\tilde{y}_t$ = output gap (log deviation of output from natural level)
- β = household discount factor (0.99 quarterly)
- κ = slope of Phillips curve, depends on price stickiness

The NKPC relates current inflation to expected future inflation and the output gap. Higher output gap (demand pressure) leads to higher inflation.

2.2 Dynamic IS Equation (DIS)

$$\tilde{y}_t = \mathbb{E}_t\{\tilde{y}_{t+1}\} - \frac{1}{\sigma}(i_t - \mathbb{E}_t\{\pi_{t+1}\} - r_t^n)$$

(Galí Equation 22)

where:

- i_t = nominal interest rate
- r_t^n = natural rate of interest
- σ = inverse elasticity of intertemporal substitution (CRRA)

The DIS is derived from the household's Euler equation. Higher real interest rates (i_t - $\mathbb{E}_t\{\pi_{t+1}\}$) relative to the natural rate reduce current output gap.

2.3 Monetary Policy Rule (Taylor Rule)

$$i_t = \rho + \phi_\pi \pi_t + \phi_y \tilde{y}_t + v_t$$

(Galí Equation 25)

where:

- ρ = steady-state real interest rate (= -log(β))
- $\varphi\pi$ = response to inflation (typically 1.5, Taylor principle: $\varphi\pi > 1$)
- φy = response to output gap (typically 0.5/4 = 0.125 quarterly)
- v_t = monetary policy shock (deviation from rule)

The Taylor rule describes how the central bank sets interest rates in response to inflation and output gap deviations from target.

3. Natural Rate and Shock Processes

3.1 Natural Rate of Interest

$$r_t^n = \rho + \sigma\psi_{ya}^n \mathbb{E}_t\{\Delta a_{t+1}\}$$

(Galí Equation 23)

The natural rate is the real interest rate that would prevail under flexible prices.
For an AR(1) productivity process $a_t = \rho_a a_{t-1} + \varepsilon_t^a$:

$$\mathbb{E}_t\{\Delta a_{t+1}\} = \mathbb{E}_t\{a_{t+1} - a_t\} = (\rho_a - 1) a_t$$

So in deviations from steady state: $r_t^n - \rho = \sigma \psi^n ya (\rho_a - 1) a_t$

Since $\rho_a < 1$, a positive productivity shock lowers the natural rate.

3.2 Exogenous Shock Processes

The model has two exogenous AR(1) shock processes:

Productivity Shock (affects natural rate):
$a_t = \rho_a a_{t-1} + \sigma_a \varepsilon_t^a$, $\varepsilon_t^a \sim N(0,1)$

Monetary Policy Shock (deviation from Taylor rule):
$v_t = \rho_v v_{t-1} + \sigma_v \varepsilon_t^v$, $\varepsilon_t^v \sim N(0,1)$

Default calibration:
- $\rho_a = 0.9$ (persistent productivity)
- $\rho_v = 0.5$ (less persistent monetary shock)
- $\sigma_a = 0.01$ (1% std dev)
- $\sigma_v = 0.0025$ (25 basis points)

3.3 State Space Representation

State Variables (exogenous): $s_t = [a_t, v_t]$
Policy Variables (endogenous): $p_t = [\tilde{y}_t, \pi_t]$

The interest rate i_t is determined by the Taylor rule given the policy variables.
The model is solved by finding a policy function p(s) that satisfies the equilibrium conditions (Euler equations) at all points in the state space.

4. Implementation in JAXecon/DEQN

4.1 File Structure

```
DEQN/econ_models/NK/
├── __init__.py       # Module exports
├── model.py          # Model class with equilibrium conditions
├── train.py          # Training script
└── analysis.py       # Impulse response analysis
```

4.2 Key Methods in model.py

• __init__(): Initialize parameters (β, σ, κ, $\varphi\pi$, φy, ρ_a, ρ_v, etc.)

• step(state, policy, shock): Transition function
Returns next state: $s_{t+1} = [\rho_a\, a_t + \sigma_a\, \varepsilon_t^a,\ \rho_v\, v_t + \sigma_v\, \varepsilon_t^v]$

• expect_realization(state_next, policy_next):
Returns $[\tilde{y}_{t+1}, \pi_{t+1}]$ for computing expectations

• loss(state, expect, policy): Euler equation residuals
Computes residuals for DIS and NKPC:

DIS residual: $\tilde{y}_t - \mathbb{E}_t\{\tilde{y}_{t+1}\} + (1/\sigma)(i_t - \mathbb{E}_t\{\pi_{t+1}\} - r_t^n)$
NKPC residual: $\pi_t - \beta\, \mathbb{E}_t\{\pi_{t+1}\} - \kappa\, \tilde{y}_t$

where $i_t = \varphi\pi\, \pi_t + \varphi y\, \tilde{y}_t + v_t$ (Taylor rule)

• get_aggregates(): Compute all variables from states and policies

4.3 DEQN Algorithm

The Deep Equilibrium Network (DEQN) algorithm:

1. Neural network $\pi\theta(s)$ maps states to policies: $[a_t, v_t] \to [\tilde{y}_t, \pi_t]$

2. Simulate episodes using the policy network

3. Compute Euler equation residuals using Monte Carlo expectations

4. Minimize squared residuals via gradient descent (Adam optimizer)

5. Repeat until convergence (accuracy > 99%)

The trained network provides an approximate global solution to the model.

5. Calibration

Default calibration (quarterly frequency):

Parameter	Value	Description
β	0.99	Discount factor
σ	1.0	CRRA coefficient (log utility)
κ	0.1275	NKPC slope
$\varphi\pi$	1.5	Taylor rule: inflation response
φy	0.125	Taylor rule: output gap response (0.5/4)
ρ_a	0.9	Productivity shock persistence
ρ_v	0.5	Monetary shock persistence
σ_a	0.01	Productivity shock std dev
σ_v	0.0025	Monetary shock std dev
$\psi^n ya$	1.0	Natural output elasticity to productivity

6. Usage

Training the model:
```
    python -m DEQN.econ_models.NK.train
```

Running impulse response analysis:
```
    python -m DEQN.econ_models.NK.analysis
```

Using the model in Python:
```
    from DEQN.econ_models.NK.model import Model

    # Create model with default parameters
    model = Model()

    # Or with custom calibration
    model = Model(beta=0.99, kappa=0.15, phi_pi=2.0)
```

7. References

• Galí, J. (2015). Monetary Policy, Inflation, and the Business Cycle:
An Introduction to the New Keynesian Framework. Princeton University Press.
Chapter 3: The Basic New Keynesian Model.

• Azinovic, M., Gaegauf, L., & Scheidegger, S. (2022). Deep Equilibrium Nets.
International Economic Review, 63(4), 1471-1525.