Week 5 — Deep Equilibrium Networks II

Stochastic Models & Gauss–Hermite Quadrature

Dr Juan Zurita

Deep Learning for Macroeconomics
The University of Edinburgh · School of Economics

Adding Uncertainty

Real economies face shocks: $y_t = e^{z_t} k_t^{\alpha}$, where z_t follows an AR(1).

The Euler equation now involves an expectation:

$$u'(c_t) = \beta \, \mathbb{E}_t[u'(c_{t+1}) \, (1 + r_{t+1})]$$

How do we compute $\mathbb{E}_t[\cdot]$ inside a neural-network training loop?

Gauss–Hermite Quadrature

Approximate the expectation with a weighted sum:

$$\mathbb{E}[g(\varepsilon)] \approx \sum_{j=1}^{J} \omega_j \, g(\varepsilon_j)$$

- ε_j: quadrature nodes (roots of Hermite polynomials)
- ω_j: quadrature weights
- With $J = 5$–7 nodes, accuracy is excellent for smooth integrands

This integrates seamlessly with PyTorch — each node is a forward pass through the network.

Stochastic Brock–Mirman

State: (k_t, z_t). Policy: $c_t = \mathcal{N}_{\boldsymbol{\theta}}(k_t, z_t)$.

Residual with quadrature:

$$R(k, z; \boldsymbol{\theta}) = 1 - \beta \sum_j \omega_j \frac{\alpha(k')^{\alpha-1} e^{z'_j}}{c(k', z'_j; \boldsymbol{\theta}) / c(k, z; \boldsymbol{\theta})}$$

Now the network takes **two** inputs — this is where the grid-free advantage starts to matter.

Loss Function Variants

Loss	Formula	Properties
MSE	R^2	Standard, sensitive to outliers
Huber	smooth L_1	Robust to large residuals
Log-cosh	$\ln(\cosh R)$	Smooth Huber approximation

In practice, **MSE** works well for most economic models. Switch to Huber if training is unstable.

Summary

- Stochastic models require numerical integration of expectations
- Gauss–Hermite quadrature is accurate and differentiable
- The DEQN approach extends naturally to stochastic settings
- With two state variables, we already outperform naive grids

Next week: occasionally binding constraints — the Fischer–Burmeister trick.