Week 1 — From Numerical Methods to Neural Networks

Why Traditional Methods Struggle & Neural Networks as Function Approximators

Dr Juan Zurita

Deep Learning for Macroeconomics

The University of Edinburgh · School of Economics

The Challenge

Traditional numerical methods break down in high dimensions.

The Curse of Dimensionality

A grid with n points per dimension in d dimensions has n^d nodes.

$n = 100$, $d = 5 \Rightarrow 10^{10}$ grid points — infeasible.

Deep learning offers a **grid-free** alternative.

From PNM to Deep Learning

PNM	This course
Root-finding	Equilibrium as loss functions
BFGS / Nelder–Mead	Gradient descent, Adam
Polynomials / splines	Neural networks
Value-function iteration	Deep Equilibrium Networks
Grid-based methods	Grid-free solutions

The key shift: parameterise value and policy functions as neural networks.

Universal Approximation Theorem

Cybenko (1989), Hornik et al. (1989)

A feedforward network with one hidden layer and a non-linear activation can approximate any continuous function on a compact set to arbitrary accuracy.

This is the theoretical foundation for using neural networks as function approximators in economics.

Practical question: **how many** neurons do we need?

A Single Neuron

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

- **w**: weights (parameters to learn)
- b: bias
- σ: activation function (ReLU, sigmoid, tanh)

A **layer** stacks many neurons; a **network** stacks many layers.

Your First Network in PyTorch

Key idea

Define the architecture, choose a loss, run gradient descent.

In the notebook you will:

1. Approximate $\sin(x)$ with a small network
2. Compare with polynomial and Chebyshev approximation from PNM
3. See how adding layers and neurons improves accuracy

Next week: the full deep learning toolkit — backpropagation, regularisation, and training.