Spooky Boundaries at a Distance: Exploring Transversality and Stationarity with Deep Learning

Mahdi Ebrahimi Kahou[1] Jesús Fernández-Villaverde[2] Sebastián Gómez-Cardona[1] Jesse Perla[1] Jan Rosa[1]

September 26, 2022

[1]University of British Columbia, Vancouver School of Economics

[2]University of Pennsylvania

Motivation

Motivation

In the long run we are all dead; J.M. Keynes

- Dynamic models usually require **economic assumptions** eliminating explosive solutions (e.g., transversality, or no-bubble.)
  - These are variations on "boundary conditions" for **forward looking** behavior of agents.
  - Deterministic, stochastic, sequential, recursive formulations all require conditions in some form.

- These forward-looking boundary conditions are the key limitation on increasing dimensionality:
  - Otherwise, in sequential setups, we can easily solve high-dimensional initial value problems.

- **Question:** Can we avoid precisely calculating steady-state(or stationary distribution), which is never reached, and still have accurate short/medium-run dynamics disciplined by these boundary conditions?

Contribution

- Show that **deep learning** solutions to many dynamic forward looking models automatically fulfill the long run boundary conditions we need (transversality and no-bubble).
  - We show how to design the approximation using economic insight.
  - This is not reinforcement learning. This is just classical function approximation.
- Solve classic models with known solutions (asset pricing and neoclassical growth) and show excellent short/medium term dynamics –even when **non-stationary** or with **steady-state multiplicity**.
- Suggests these methods may solve high-dimensional problems while avoiding the key computational limitation—with the only trade-off being loss of precision for equilibria after "we are all dead."
  - We have to understand low-dimensional problems first.
- **Intuition**: The solutions that violate the long run boundary conditions are explosive and deep learning solutions eliminate them.

But first, we need to be very precise on what deep learning solutions mean in this context.

Background: Deep learning for functional equations

Models as functional equations

Equilibrium conditions in economics can be written as functional equations:

- Take some function(s) $f \in \mathcal{F}$ where $f : \mathcal{X} \to \mathcal{Y}$ (e.g. optimal policy and consumption function in neoclassical growth model).
- Domain $\mathcal{X}$ could be state (e.g. capital) or time if sequential.
- The "model" is $\ell : \mathcal{F} \times \mathcal{X} \to \mathcal{R}$ (e.g. Euler residuals and feasibility condition).
- The solution is the "zero" (root) of the model (residuals operator), i.e. $0 \in \mathcal{R}$, at each $x \in \mathcal{X}$ (e.g. optimal policy is the root of the Euler and feasibility condition residuals over the space of capital).

Then a **solution** is an $f^* \in \mathcal{F}$ where $\ell(f^*, x) = 0$ for all $x \in \mathcal{X}$.

Example: neoclassical growth model

Bellman formulation of the neoclassical growth problem:

$$
\begin{aligned}
v(k) = \max_{c,k'} \{u(c) + \beta v(k')\} \\
\text{s.t.} \quad c + k' = f(k) + (1-\delta)k \\
k' \geq 0.
\end{aligned}
$$

- Along with the transversality condition.
- Capital, k, consumption c, utility $u(c)$, discount rate β, depreciation δ, production function $f(k)$.
- The Euler equation:

$$u'(c) = \beta u'(c')\left(f'(k') + 1 - \delta\right).$$

Example: one formulation of neoclassical growth

- Domain: $x = \begin{bmatrix} k \end{bmatrix}$ and $\mathcal{X} = \mathbb{R}_+$.
- Solve for the optimal policy $k'(\cdot)$ and consumption function $c(\cdot)$: So $f : \mathbb{R} \to \mathbb{R}^2$ and $\mathcal{Y} = \mathbb{R}^2_+$.
- Residuals are the Euler equation and feasibility condition, so $\mathcal{R} = \mathbb{R}^2$:

$$\ell(\underbrace{\begin{bmatrix} k'(\cdot) & c(\cdot) \end{bmatrix}}_{\equiv f}, \underbrace{k}_{\equiv x}) = \underbrace{\begin{bmatrix} u'(c(k)) - \beta u'(c(k'(k)))\left(f'(k'(k)) + 1 - \delta\right) \\ f(k) - c(k) - k'(k) + (1-\delta)k \end{bmatrix}}_{\text{model}}$$

- Finally, $f^* = [k'(\cdot), c(\cdot)]$ is a solution if it has zero residuals on domain $\mathcal{X}$.

Classical solution method for functional equations

1. **Pick** finite set of N points $\mathcal{X}_{\text{train}} \subset \mathcal{X}$ (e.g., a grid).
2. **Choose** approximation $\hat{f}(\cdot;\theta) \in \mathcal{H}(\Theta)$ with coefficients $\Theta \subseteq \mathbb{R}^M$ (e.g., Chebyshev polynomials, splines).
3. **Fit** with nonlinear least-squares

$$\min_{\theta\in\Theta} \sum_{x\in\mathcal{X}_{\text{train}}} \ell(\hat{f}(\cdot;\theta), x)^2$$

If $\theta \in \Theta$ is such that $\ell(\hat{f}(\cdot;\theta), x) = 0$ for all $x \in \mathcal{X}_{\text{train}}$ we say it **interpolates** $\mathcal{X}_{\text{train}}$.

Good generalization is the goal

- If we have more coefficients than grid points ($M \geq N$) we usually interpolate exactly:
  - $\hat{f}(x;\theta) \approx f^*(x)$ for $x \in \mathcal{X}_{\text{train}}$.
- The goal is to have good **generalization**:
  - The approximate function is close to the solution outside of $\mathcal{X}_{\text{train}}$.
  - That is $\hat{f}(x;\theta) \approx f^*(x)$ for $x \notin \mathcal{X}_{\text{train}}$.
- Usually tinkering with "pick", "choose", and "fit" steps to attain low generalization error.

A deep learning approach

Recall we need $\hat{f}(\cdot;\theta) \in \mathcal{H}$:

- **Deep neural networks** are **highly-overparameterized** functions designed for good generalization.
  - Number of coefficients much larger than the grid points ($M \gg N$).

How to construct these functions:

- Example: one layer neural network, $\hat{f} : \mathbb{R}^Q \to \mathbb{R}$:

$$\hat{f}(x;\theta) = W_2 \cdot \sigma\left(W_1 \cdot x + b_1\right) + b_2 \tag{1}$$

- $W_1 \in \mathbb{R}^{P\times Q}$, $b_1 \in \mathbb{R}^{P\times 1}$, $W_2 \in \mathbb{R}^{1\times P}$, and $b_2 \in \mathbb{R}$.
- $\sigma(\cdot)$ is a nonlinear function applied element-wise (e.g., $\max\{\cdot, 0\}$).
- $\Theta \equiv \{b_1, W_1, b_2, W_2\}$ are the coefficients, in this example $M = PQ + P + P + 1$.
- Architecture of the neural networks ($\mathcal{H}$) can be flexibly informed by the economic insight and theory.

Empirically deeper neural networks generalize better

- Making it "deeper" by adding another "layer":

$$\hat{f}(x;\theta) \equiv W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2) + b_3. \tag{2}$$

- Let's generically call them $NN(\cdot;\theta)$ and explain the structure if it is needed.
- Software (e.g., PyTorch) makes it easy to experiment with different $\mathcal{H}$ (i.e., neural networks), manage the Θ.
- $\hat{f}(x;\theta)$ and the gradients can be used to solve $\min_{\theta\in\Theta} \sum_{x\in\mathcal{X}_{\text{train}}} \ell(\hat{f}(\cdot;\theta), x)^2$.

Deep learning optimizes in a space of functions: which $\hat{f}$?

- Since $M \gg N$, it is possible for $\hat{f}$ to interpolate and the objective value will be ≈ 0.
- Since $M \gg N$ has an enormous number of solutions (e.g., θ_1 and θ_2),
  - Agree on the grid points: $\hat{f}(x; \theta_1) \approx \hat{f}(x; \theta_2)$ for $x \in \mathcal{X}_{\text{train}}$.
- Since individual θ are irrelevant it is helpful to think of optimization directly within $\mathcal{H}$

$$\min_{\hat{f} \in \mathcal{H}} \sum_{x \in \mathcal{X}_{\text{train}}} \ell(\hat{f}, x)^2$$

Which $\hat{f}$?

Deep learning and interpolation

- **Counterintuitively:** for M large enough, optimizers **tend to** converge towards something **unique** $\hat{f}$ in equivalence class from some $\|\cdot\|_S$ define on $x \in \mathcal{X}$ (i.e., not just at interpolated grid points).
- **Mental model:** chooses min-norm interpolating solution for a (usually) unknown functional norm S

$$\min_{\hat{f} \in \mathcal{H}} ||\hat{f}||_S$$
$$\text{s.t. } \ell(\hat{f}, x) = 0, \qquad \text{for } x \in \mathcal{X}_{\text{train}}$$

  - CS and literature refers to this as the **inductive bias**: optimization process biased towards particular $\hat{f}$.
  - Intuition is that it may choose the interpolating solutions which are flattest and have smallest derivatives.
  - Characterizing S (e.g., Sobolev?) is an active research area in CS at the heart of deep learning theory.
    - Belkin, M. (2021) and Ma, C., & Ying, L. (2021).

Examples

Deep learning and interpolation in practice

Reminder: in practice we solve

$$\min_{\theta \in \Theta} \sum_{x \in \mathcal{X}_{\text{train}}} \ell \left(\hat{f}(\cdot; \theta), x \right)^2$$

- The min-norm interpolation happens **implicitly** through the optimizers.
- No explicit norm minimization or penalization is required.

In this paper: we describe how the $\min_{\hat{f} \in \mathcal{H}} ||\hat{f}||_S$ solutions are also the ones which automatically fulfill transversality and no-bubble conditions in many dynamic models.

- They are disciplined by long run boundary conditions. Therefore, we can obtain accurate short/medium-run dynamics.

Agenda

To explore how we can ignore events after "we are all dead", we show deep learning solutions to

1. Classic linear-asset pricing model.
2. Sequential formulation of the neoclassical growth model.
3. Sequential neoclassical growth model with multiple steady states.
4. Recursive formulation of the neoclassical growth model.
5. Non-stationarity, such as balanced growth path.

We can show numerical solutions while explaining theory from the economics, but there are current limitations on theory from CS.

Linear asset pricing

Sequential formulation

- Dividends, $y(t)$, y_0 as given, and follows the process:

$$y(t+1) = c + (1+g)y(t)$$

- Writing as a linear state-space model with $x(t+1) = Ax(t)$ and $y(t) = Gx(t)$ and

$$x(t) \equiv \begin{bmatrix} 1 & y(t) \end{bmatrix}^\top, A \equiv \begin{bmatrix} 1 & 0 \\ c & 1+g \end{bmatrix}, G \equiv \begin{bmatrix} 0 & 1 \end{bmatrix}$$

- "Fundamental" price given $x(t)$ is PDV with $\beta \in (0,1)$ and $\beta(1+g) < 1$

$$p_f(t) \equiv \sum_{j=0}^{\infty} \beta^j y(t+j) = G(I - \beta A)^{-1} x(t).$$

Recursive formulation

With standard transformation, all solutions $p_f(t)$ fulfill the recursive equations

$$p(t) = Gx(t) + \beta p(t+1) \tag{3}$$

$$x(t+1) = Ax(t) \tag{4}$$

$$0 = \lim_{T \to \infty} \beta^T p(T) \tag{5}$$

$$x_0 \text{ given} \tag{6}$$

That is, a system of two difference equations with one boundary and one initial condition.

- The boundary condition (5) is an **assumption** necessary for the problem to be well-posed and have a unique solution.
- It ensures that $p(t) = p_f(t)$ by imposing long run boundary on forward-looking behavior.
- But without this assumption there can be "rational bubbles" with $p(t) \neq p_f(t)$, only fulfilling (3) and (4).
- Intuition: system of $(p(t), x(t))$ difference (or differential) equations requires total of two boundaries or initial values to have a unique solution (i.e., (6) not enough on its own)

Solutions without no-bubble condition

- Rational bubble solutions in this deterministic asset pricing model are of the form:

$$p(t) = p_f(t) + \zeta\ \beta^{-t}. \tag{7}$$

- For any $\zeta \geq 0$. The x_t initial condition determined the $p_f(t)$ solution.
- Without the no-bubble condition there are infinitely many solutions.
- The no-bubble condition chooses the $\zeta = 0$ solution.

Interpolation problem: without no-bubble condition

- A set of points in time $\mathcal{X}_{\text{train}} = \{t_1, \ldots, t_{\text{max}}\}$.
- An over-parameterized family of approximating functions $p(\cdot; \theta) \in \mathcal{H}(\Theta)$ (i.e. deep neural networks).
- Generate $x(t)$ using the law of motion and $x(0)$, equation (4).
  (In practice) we minimize the residuals of the recursive form for the price (equation (3)):

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{X}_{\text{train}}|} \sum_{t \in \mathcal{X}_{\text{train}}} [p(t; \theta) - Gx(t) - \beta p(t+1; \theta)]^2 \tag{8}$$

- This minimization **does not contain** no-bubble condition. It has infinitely many minima.
- Does the min-norm inductive bias of over-parameterized interpolation weed out the bubbles? **Yes**.
- **Intuition**: bubble solutions are explosive (big functions with big derivatives).

Let's analyze this more rigorously.

Interpolation formulation: min-norm mental model

The min-norm mental model can be written as:

$$\min_{p \in \mathcal{H}} \quad \|p\|_S \tag{9}$$

$$\text{s.t.} \quad p(t) - Gx(t) - \beta p(t+1) = 0 \quad \text{for } t \in \mathcal{X}_{\text{train}} \tag{10}$$

$$0 = \lim_{T \to \infty} \beta^T p(T) \tag{11}$$

Where $x(t)$ for $t \in \mathcal{X}_{\text{train}}$ is defined by $x(0)$ initial condition and recurrence $x(t+1) = Ax(t)$ in (4)

- The minimization of norm $\|p\|_S$ has "inductive bias" towards particular solutions for $t \in [0, \infty] \setminus \mathcal{X}_{\text{train}}$.

Is the no-bubble condition still necessary?

- To analyze, drop the no-bubble condition and examine the class of solutions. Does it bind?
- In this case, we know the interpolating solutions to (10) without imposing (11)

$$p(t) = p_f(t) + \zeta\beta^{-t} \tag{12}$$

- Take some norm $\|.\|_S$ of both sides and apply triangle inequality

$$\|p_f\|_S \leq \|p\|_S \leq \|p_f\|_S + \zeta \, \|\beta^{-t}\|_S \tag{13}$$

- Relative to classic methods the "deep learning" problem now has a $\|p\|_S$ objective!
  - From (13) for a large class of S, the norm minimizing $\|p\|_S$ will be one where $\zeta = 0$
  - That is, $p(t) = p_f(t)$, the solution fulfills the no-bubble condition, and (11) is satisfied at the optima.
- The new objective of minimizing the norm, makes the no-bubble condition **redundant**.

Min-norm norm formulation: redundancy of no-bubble condition

Given the no-bubble condition is automatically fulfilled, could solve the following given some $\mathcal{H}$ and compare to $p_f(t)$

$$\min_{p \in \mathcal{H}} \quad \|p\|_S \tag{14}$$

$$\text{s.t.} \quad p(t) - Gx(t) - \beta p(t+1) = 0 \quad \text{for } t \in \mathcal{X}_{\text{train}} \tag{15}$$

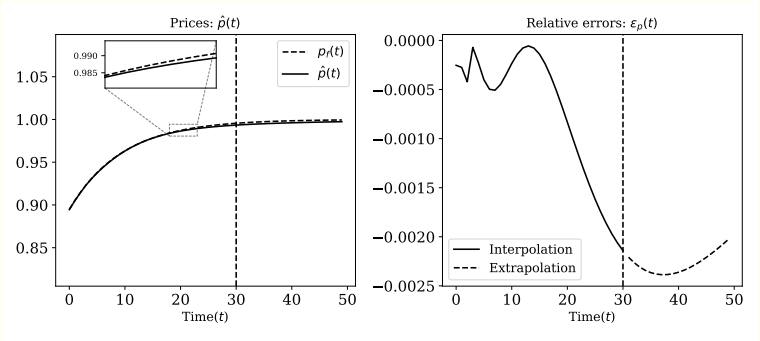
A reminder: in practice, given the $\mathcal{X}_{\text{train}}$, we directly implement this as $p(\cdot;\theta) \in \mathcal{H}(\Theta)$ and fit with

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{X}_{\text{train}}|} \sum_{t \in \mathcal{X}_{\text{train}}} \left[p(t;\theta) - Gx(t) - \beta p(t+1;\theta)\right]^2 \tag{16}$$

Since law of motion is deterministic, given $x(0)$ we generate $x(t)$ with $x(t+1) = Ax(t)$ for $t \in \mathcal{X}_{\text{train}}$

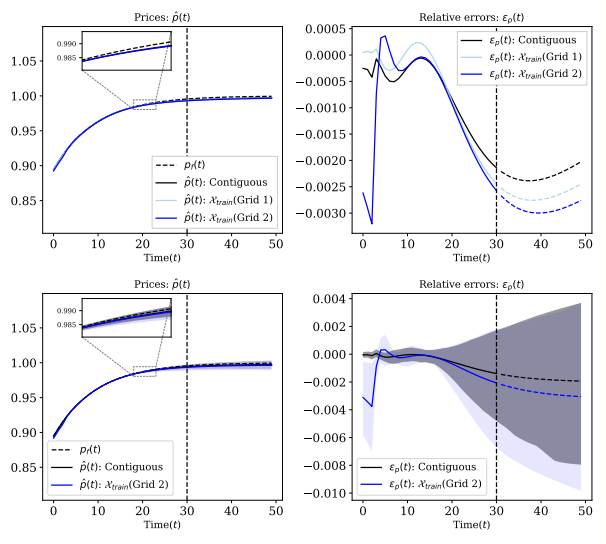
- The $\mathcal{X}_{\text{train}}$ does not need to be contiguous and $|\mathcal{X}_{\text{train}}|$ may be relatively small.
- Most important: no steady state calculated, nor large $T \in \mathcal{X}_{\text{train}}$ required.

Results



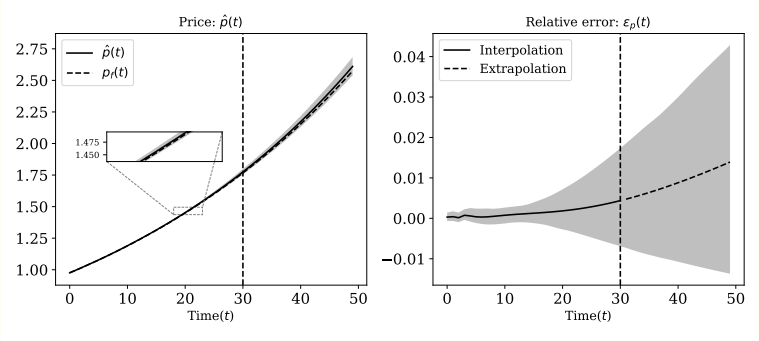
1. **Pick** $\mathcal{X}_{\text{train}} = [0, 1, 2, ..., 29]$ and $t > 29$ is "extrapolation" where $c = 0.01$, $g = -0.1$, and $y_0 = 0.8$.
2. **Choose** $p(t; \theta) = NN(t; \theta)$ where "NN" has 4 hidden layers of 128 nodes. $|\Theta| = 49.9K$ coefficients.
3. **Fit** using L-BFGS and PyTorch in just a **few seconds**. Could use Adam/SGD/etc.
4. Low generalization errors, even without imposing no-bubble condition. Compare to analytic $p_f(t)$.

Contiguous vs. sparse grid



- $\mathcal{X}_{\text{train}}(\text{Grid } 1) = [0, 1, 2, 4, 6, 8, 12, 16, 20, 24, 29]$ and $\mathcal{X}_{\text{train}}(\text{Grid } 2) = [0, 1, 4, 8, 12, 18, 24, 29]$.
- Contrary to popular wisdom that deep learning only being appropriate in high "data" environments
  - Can use **less grid points** relative to alternatives.
- The same neural network with $|\Theta| = 49.9K$.
- Shaded regions: 10th and 90th percentiles over 100 seeds:
  - The solutions are very close in the extrapolation region (unseen grid points).

Growing dividends



- **Pick** same $\mathcal{X}_{\text{train}}$ but now $c = 0.0$, $g = 0.02$, and $y_0 = 0.8$ ($y(t)$ grows at rate g).
- **Choose** $p(t;\theta) = e^{\phi t} NN(t;\theta_1)$ where $\theta \equiv \{\phi, \theta_1\} \in \Theta$ are the coefficients.
  - Here we used economic intuition of problem to design the $\mathcal{H}$ to generalize better.
- Non-stationary but can figure out the growth. Short term errors are very small, long run manageable.
- Bonus: learns the growth rate: $\phi \approx \ln(1+g)$ and even extrapolates well! ⏩ Examples

Neoclassical growth in sequence space

Sequential formulation (with a possible BGP)

$$\max_{\{c(t),k(t+1)\}_{t=0}^{\infty}} \quad \sum_{t=0}^{\infty} \beta^t u\left(c(t)\right) \tag{17}$$

$$\text{s.t.} \quad k(t+1) = z(t)^{1-\alpha} f\left(k(t)\right) + (1-\delta)k(t) - c(t) \tag{18}$$

$$z(t+1) = (1+g)z(t) \tag{19}$$

$$k(t) \geq 0 \tag{20}$$

$$0 = \lim_{T\to\infty} \beta^T u'\left(c(T)\right) k(T+1) \tag{21}$$

$$k_0, z_0 \text{ given} \tag{22}$$

- Preferences: $u(c) = \frac{c^{1-\sigma}-1}{1-\sigma}$, $\sigma > 0$, $\lim_{c\to 0} u'(c) = \infty$, and $\beta \in (0,1)$.
- Cobb-Douglas production function: $f(k) = k^{\alpha}$, $\alpha \in (0,1)$ before scaling by TFP z_t.
- Skip standard steps... Euler equation: $u'(c(t)) = \beta u'(c(t+1))\left[z(t+1)^{1-\alpha} f'(k(t+1)) + 1 - \delta\right]$.

Interpolation problem: without transversality condition

- A set of points in time $\mathcal{X}_{\text{train}} = \{t_1, \dots, t_{\text{max}}\}$.
- An over-parameterized family of approximating functions $k(\cdot;\theta) \in \mathcal{H}(\Theta)$ (i.e. deep neural networks).
- Generate $z(t)$ using the law of motion and $z(0)$, equations (19).
- Use the feasibility condition and define $c(t;k) \equiv z(t)^{1-\alpha} f\big(k(t)\big) + (1-\delta)k(t) - k(t+1)$.

(In practice) we minimize the Euler and initial conditions residuals:

$$\min_{\theta\in\Theta} \frac{1}{|\mathcal{X}_{\text{train}}|} \sum_{t\in\mathcal{X}_{\text{train}}} \lambda_1 \Bigg[\underbrace{\beta u'\big(c(t+1;k(\cdot;\theta))\big)\big[z(t+1)^{1-\alpha} f'(k(t+1;\theta)) + 1 - \delta\big]}_{\text{Euler residuals}} - \underbrace{u'\big(c(t;k(\cdot,\theta))\big)}_{} \Bigg]^2 + \lambda_2 \Bigg[\underbrace{k(0;\theta) - k_0}_{\text{Initial condition residuals}} \Bigg]^2$$

- λ_1 and λ_2 positive weights.

Interpolation problem: without transversality condition

- This minimization **does not contain** the transversality condition.
  - Without the transversality condition it has infinitely many minima.
- **No explicit** norm minimization.
- Does the min-norm inductive bias weed out the solutions that violate the transversality condition? **Yes**.
- **Intuition**: The solutions that violate the transversality condition are big functions with big derivatives.

  Let's analyze this more rigorously.

Interpolation formulation: min-norm mental model

$$\min_{k \in \mathcal{H}} \quad \|k\|_S \tag{23}$$

$$\text{s.t.} \quad u'(c(t;k)) = \beta u'(c(t+1;k))\left[z(t+1)^{1-\alpha} f'(k(t+1)) + 1 - \delta\right] \quad \text{for } t \in \mathcal{X}_{\text{train}} \tag{24}$$

$$k(0) = k_0 \tag{25}$$

$$0 = \lim_{T \to \infty} \beta^T u'(c(T;k)) k(T+1) \tag{26}$$

$$c(t;k) \equiv z(t)^{1-\alpha} f\big(k(t)\big) + (1-\delta)k(t) - k(t+1) \tag{27}$$

Where $z(t)$ for $t \in \mathcal{X}_{\text{train}}$ is defined by $z(0)$ initial condition and recurrence $z(t+1) = (1+g)z(t)$.

Is the transversality condition still necessary? Case of $g = 0$, $z = 1$

Sketch of the proof:

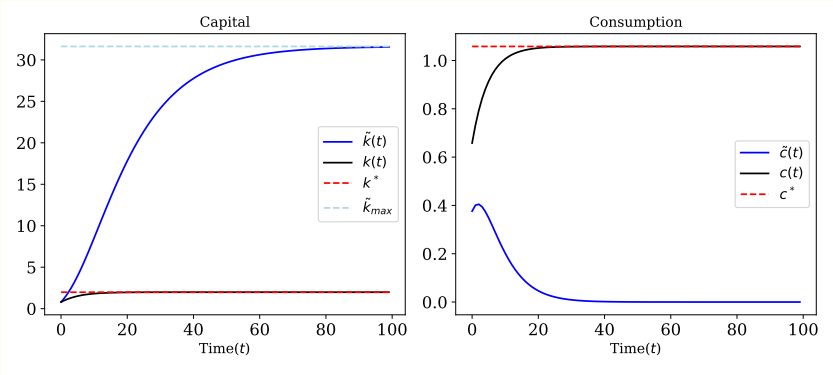
- Let $\{k(t), c(t)\}$ be the sequence of optimal solution.
- Let $\{\tilde{k}(t), \tilde{c}(t)\}$ be a sequence of solution that satisfy all the equations **except** transversality condition (26).

1. $\tilde{c}(t)$ approaches zero.
2. $\tilde{k}(t)$ approaches $\tilde{k}_{\max} \equiv \delta^{\frac{1}{\alpha-1}}$, and $k(t)$ approaches $k^* \equiv \left(\frac{\beta^{-1}+\delta-1}{\alpha}\right)^{\frac{1}{\alpha-1}}$.
3. Both $\tilde{k}(t)$ and $k(t)$ are monotone. $\tilde{k}_{\max} \gg k^*$. Therefore, for any norm $\|\cdot\|_S$, that measures level or derivatives of a function

$$0 \leq \|k\|_S \leq \|\tilde{k}\|_S.$$

Is the transversality condition still necessary? Case of g = 0, z = 1

Example: the violation of the transversality condition.



- The solution that violate the transversality are associate with **"big"** capital path.
- The new objective of minimizing the norm, makes the transversality condition **redundant**.

Min-norm formulation: redundancy of transversality condition

Given the transversality condition is automatically fulfilled, one could solve

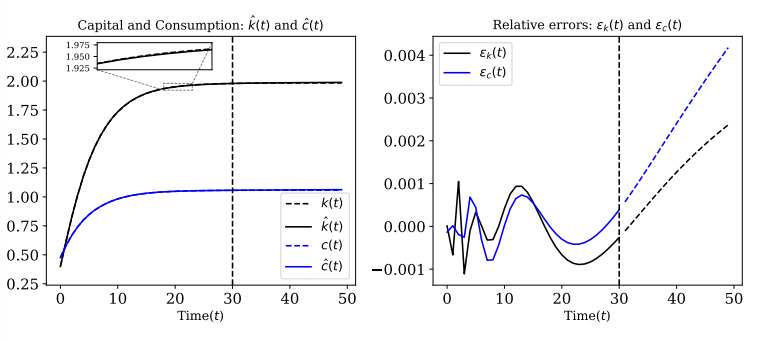
$$\begin{aligned} \min_{k \in \mathcal{H}} \quad & \|k\|_S \\ \text{s.t.} \quad & u'(c(t;k)) = \beta u'(c(t+1;k))\left[z(t+1)^{1-\alpha} f'(k(t+1)) + 1 - \delta\right] \quad \text{for } t \in \mathcal{X}_{\text{train}} \\ & k(0) = k_0 \end{aligned}$$

Reminder: in practice we solve

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{X}_{\text{train}}|} \sum_{t \in \mathcal{X}_{\text{train}}} \lambda_1 \Big[\beta u'\big(c(t+1;k(\cdot;\theta))\big)\left[z(t+1)^{1-\alpha} f'(k(t+1;\theta)) + 1 - \delta\right] - u'\big(c(t;k(\cdot,\theta))\big) \Big]^2 + \lambda_2 \Big[k(0;\theta) - k_0 \Big]^2$$

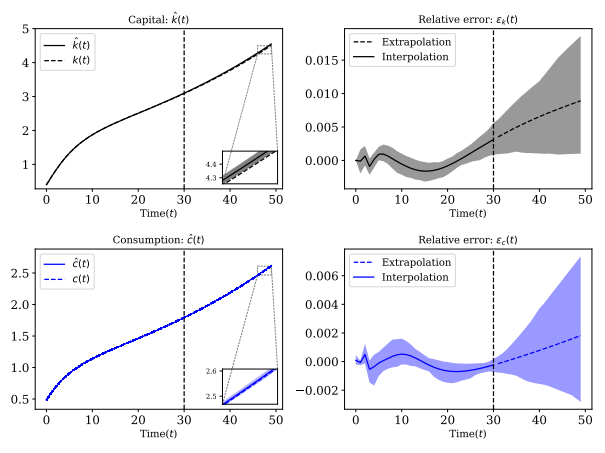
- The $\mathcal{X}_{\text{train}}$ does not need to be contiguous and $|\mathcal{X}_{\text{train}}|$ may be relatively small. Sparse
- Most important: no steady state calculated, nor large $T \in \mathcal{X}_{\text{train}}$ required.

Results



1. **Pick** $\mathcal{X}_{\text{train}} = [0, 1, 2, ..., 30]$ and $t > 30$ is "extrapolation" $\alpha = \frac{1}{3}$, $\sigma = 1$, $\beta = 0.9$, $g = 0.0$, and $k_0 = 0.4$
2. **Choose** $k(t;\theta) = NN(t;\theta)$ where "NN" has 4 hidden layers of 128 nodes. $|\Theta| = 49.9K$ coefficients.
3. **Fit** using L-BFGS in just a **few seconds**. Comparing with value function iteration solution.
4. Low generalization errors, even without imposing the transversality condition.

Growing TFP



- **Pick** same $\mathcal{X}_{\text{train}}$ but now $\alpha = \frac{1}{3}$, $\sigma = 1$, $\beta = 0.9$, $g = 0.02$, $z_0 = 1.0$ and $k_0 = 0.4$.
- **Choose** $k(t;\theta) = e^{\phi t} NN(t;\theta_{NN})$ where $\theta \equiv \{\phi, \theta_{NN}\} \in \Theta$ is the coefficient vector
  - Here we used economic intuition of problem to design the $\mathcal{H}$ to generalize better.
- Non-stationary but can figure out the BGP. Short term errors are very small.
- Bonus: learns the growth rate: $\phi \approx \ln(1+g)$ and even extrapolates well!
- Shaded regions: 10th and 90th percentiles over 100 seeds.

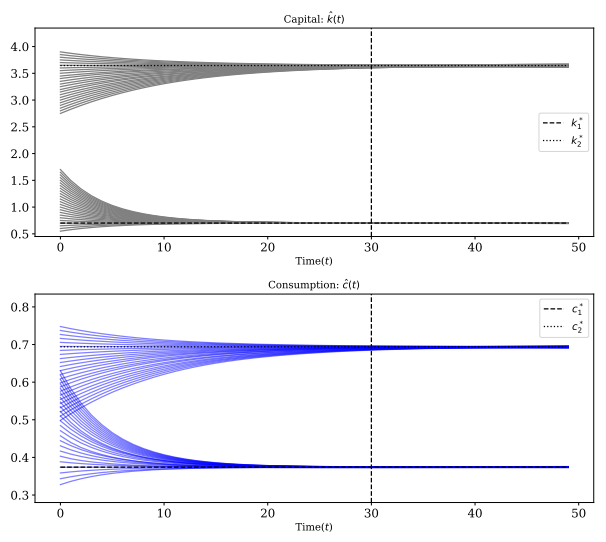
The neoclassical growth model with multiple steady states

Sequential formulation

$$
\begin{aligned}
\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} \quad & \sum_{t=0}^{\infty} \beta^t u(c_t) \\
\text{s.t.} \quad & k_{t+1} = f(k_t) + (1-\delta)k_t - c_t \\
& k_t \geq 0 \\
& 0 = \lim_{T \to \infty} \beta^T u'(c_T) k_{T+1} \\
& k_0 \text{ given.}
\end{aligned}
$$

1. Preferences: $u(c) = \frac{c^{1-\sigma}-1}{1-\sigma}$, $\sigma > 0$, $\lim_{c \to 0} u'(c) = \infty$, and $\beta \in (0,1)$.
2. **"Butterfly production function"**: $f(k) = a \max\{k^\alpha, b_1 k^\alpha - b_2\}$, $\alpha \in (0,1)$:
   - There is a kink in the production function at $k^* \equiv \left(\frac{b_2}{b_1 - 1}\right)^{\frac{1}{\alpha}}$.
   - This problem has **two** steady states, k_1^* and k_2^* and their corresponding consumption levels c_1^* and c_2^*.

Results



- **Pick** $\mathcal{X}_{\text{train}} = [0, 1, 2, ..., 29]$ and $t > 29$ is "extrapolation" $\alpha = \frac{1}{3}$, $\sigma = 1$, $\beta = 0.9$, $g = 0.0$, $a = 0.5$, $b_1 = 3.0$, and $b_2 = 2.5$, for 100 different initial conditions in $[0.5, 4.0]$.
- **Choose** $k(t;\theta) = NN(t;\theta)$ where "NN" has 4 hidden layers of 128 nodes. $|\Theta| = 49.9K$ coefficients.
- **Fit** using L-BFGS and PyTorch, each in just a few seconds.
- Low generalization errors, even without imposing the transversality condition.
- This is not intended to sharply distinguish the basins of attraction. ⏩ Details

Recursive version of the neoclassical growth model here

Recursive formulation (with a possible BGP)

Skipping the Bellman formulation and going to the first order conditions in the state space , i.e. (k, z)

$$
\begin{aligned}
u'(c(k,z)) &= \beta u'(c(k'(k,z),z'))\left[z'^{1-\alpha} f'(k'(k,z)) + 1 - \delta\right] \\
k'(k,z) &= z^{1-\alpha} f(k) + (1-\delta)k - c(k,z) \\
z' &= (1+g)z \\
k' &\geq 0 \\
0 &= \lim_{T\to\infty} \beta^T u'(c_T) k_{T+1} \quad \forall (k_0, z_0) \in \mathcal{X}
\end{aligned}
$$

- Preferences: $u(c) = \frac{c^{1-\sigma}-1}{1-\sigma}$, $\sigma > 0$, $\lim_{c\to 0} u'(c) = \infty$, and $\beta \in (0,1)$.
- Cobb-Douglas production function: $f(k) = k^{\alpha}$, $\alpha \in (0,1)$ before scaling by TFP z.

Interpolation problem: without transversality condition

- A set of points $\mathcal{X}_{\text{train}} = \{k_1, \ldots, k_{\max}\} \times \{z_1, \ldots, z_{\max}\}$.
- An over-parameterized family of approximating functions $k'(\cdot, \cdot; \theta) \in \mathcal{H}(\Theta)$ (i.e. deep neural networks).
- Use the feasibility condition and define $c(k, z; k') \equiv z^{1-\alpha} f(k) + (1-\delta)k - k'(k, z)$.

(In practice) we minimize the Euler residuals:

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{X}_{\text{train}}|} \sum_{(k,z) \in \mathcal{X}_{\text{train}}} \Bigg[\underbrace{-u'\Big(c\big(k, z; k'(.;\theta)\big)\Big) + \beta u'\Big(c\big(k'(k, z; \theta), (1+g)z; k'(.;\theta)\big)\Big)}_{\text{Euler residuals}} \times \underbrace{\Big[((1+g)z)^{1-\alpha} f'(k'(k, z; \theta)) + 1 - \delta\Big]}_{} \Bigg]^2$$

Interpolation problem: without transversality condition

- This minimization **does not contain** the transversality condition.
  - Without the transversality condition it has more than one minima.
- **No explicit** norm minimization.
- Does the min-norm inductive bias weed out the solutions that violate the transversality condition? **Yes**
- **Intuition**: The solutions that violate the transversality condition are "bigger" than those don not violate it.

  Let's analyze this more rigorously.

Interpolation formulation: min-norm mental model

$$\min_{k' \in \mathcal{H}} \quad \|k'\|_S \tag{28}$$

$$\text{s.t.} \quad u'\Big(c\big(k, z; k'\big)\Big) = \beta u'\Big(c\big(k'(k,z), (1+g)z; k'\big)\Big) \times$$

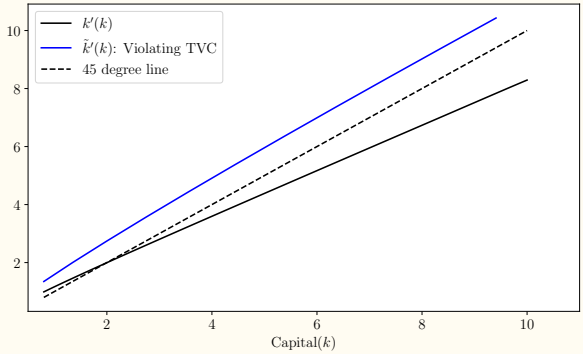
$$\big[((1+g)z)^{1-\alpha} f'(k'(k,z)) + 1 - \delta\big] \quad \text{for } (k,z) \in \mathcal{X}_{\text{train}} \tag{29}$$

$$0 = \lim_{T \to \infty} \beta^T u'(c(T)) k(T+1) \quad \text{for all } (k_0, z_0) \in \mathcal{X} \tag{30}$$

where

$$c(k, z; k') \equiv z^{1-\alpha} f(k) + (1-\delta)k - k'(k,z)$$

Is the transversality condition necessary? Case of g = 0, z = 1



- The solutions that violate the transversality condition are above the one that do not.
- They have bigger derivatives. Therefore, they have bigger norms:

$$0 \leq \|k'\|_S < \|\tilde{k}'\|_S. \tag{31}$$

Min-norm formulation: redundancy of transversality condition

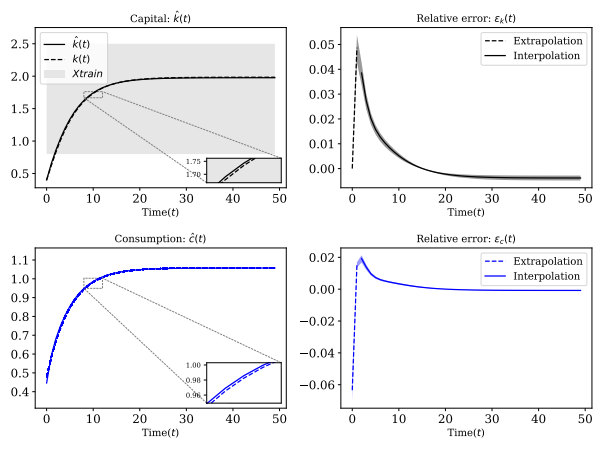
We can drop the transversality condition:

$$\begin{aligned} \min_{k' \in \mathcal{H}} \quad & \|k'\|_S \\ \text{s.t.} \quad & u'\Big(c\big(k,z;k'\big)\Big) = \beta u'\Big(c\big(k'(k,z),(1+g)z;k'\big)\Big) \times \\ & \Big[((1+g)z)^{1-\alpha} f'(k'(k,z)) + 1 - \delta\Big] \quad \text{for } (k,z) \in \mathcal{X}_{\text{train}} \end{aligned}$$

In practice, given $\mathcal{X}_{\text{train}}$, we directly implement this as $k'(\cdot,\cdot;\theta) \in \mathcal{H}(\Theta)$ and fit with

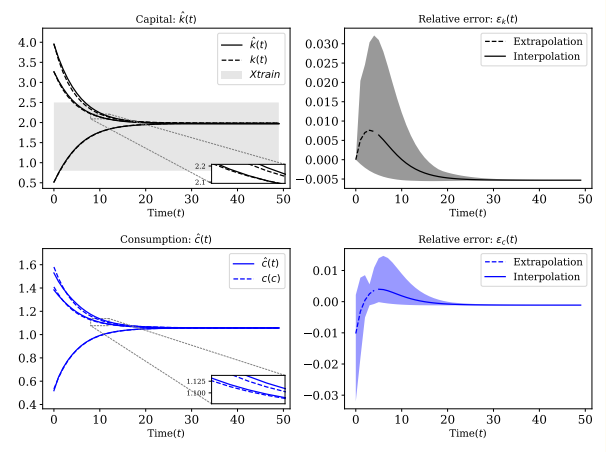
$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{X}_{\text{train}}|} \sum_{(k,z) \in \mathcal{X}_{\text{train}}} \Bigg[-u'\Big(c\big(k,z;k'(.;\theta)\big)\Big) + \beta u'\Big(c\big(k'(k,z;\theta),(1+g)z;k'(.;\theta)\big)\Big) \times \Big[((1+g)z)^{1-\alpha} f'(k'(k,z);\theta) + 1 - \delta\Big] \Bigg]^2$$

Results: one initial condition



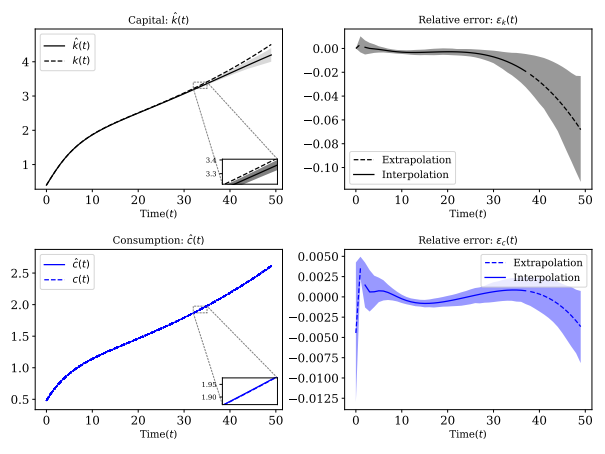
- **Pick** $\mathcal{X}_{\text{train}} = [0.8, 2.5] \times \{1\}$ and $k_0 = 0.4 \notin \mathcal{X}_{\text{train}}$ is "extrapolation" $\alpha = \frac{1}{3}$, $\sigma = 1$, $\beta = 0.9$, $g = 0.0$.
- **Choose** $k'(k, z; \theta) = NN(k, z; \theta)$ where "NN" has 4 hidden layers of 128 nodes. $|\Theta| = 49.9K$ coefficients.
- **Fit** using L-BFGS and PyTorch in just a few seconds.
- Low generalization errors, even without imposing transversality condition.
- Shaded regions: 10th and 90th percentiles over 100 seeds.

Results: initial conditions over the state space



- The solution has to satisfy the transversality condition for all points in $\mathcal{X}$
  - $\lim_{T\to\infty} \beta^T u'(c(T))k(T+1) = 0 \quad \forall\ k_0 \in \mathcal{X}$
- Left: Three different initial condition for capital, two of them outside $\mathcal{X}$.
- Shaded regions: error range in capital and consumption for 70 different initial condition in $[0.5, 4.0]$.

Growing TFP



- **Pick** $\mathcal{X}_{\text{train}} = [0.8, 3.5] \times [0.8, 1.8]$ but now $\alpha = \frac{1}{3}$, $\sigma = 1$, $\beta = 0.9$, $g = 0.02$, $z_0 = 1$, and $k_0 = 0.4 \notin \mathcal{X}_{\text{train}}$.
- **Choose** $k'(k, z; \theta) = zNN(k, \frac{k}{z}; \theta)$, same as before $|\Theta| = 49.9K$
  - Here we used economic intuition of problem to design the $\mathcal{H}$ to generalize better.
- Relative errors are very small inside the grid.
- Extrapolation works from both sides the grid for capital.
- Shaded regions: 10th and 90th percentiles over 100 seeds.

Conclusion

Conclusion

- Solving functional equations with deep learning is an extension of collocation/interpolation methods.
- With **massive over-parameterization** optimizers tend to choose those interpolating functions which are not explosive and with smaller gradients (i.e., **inductive bias**).
- Over-parameterized solutions **automatically** fulfill **forward-looking** boundary conditions:
  - Shedding light on the convergence of deep learning based solutions in dynamic problems in macroeconomics.
- If we solve models with deep-learning without (directly) imposing long run boundary conditions,
  - Short/medium run errors are small, and long run errors after **"we are all dead"** are even manageable.
  - Long run errors do not affect transition dynamics even in the presence of **non-stationarity** and **steady-state multiplicity**.
  - Gives hope for solving high-dimensional models still disciplined by forward looking economic assumptions.

Appendix

Sobolev semi-norms

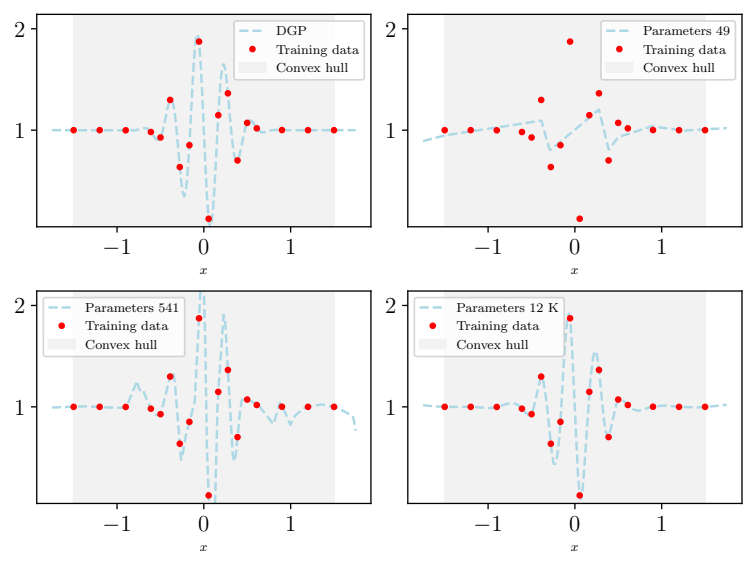
back

Let $f : \mathcal{X} \to \mathbb{R}$

$$\|f\|_{k,p} = \left(\sum_{i=0}^{k} \int_{\mathcal{X}} \left| \frac{d^i f}{dx^i}(x) \right|^p dx \right)^{\frac{1}{p}}$$

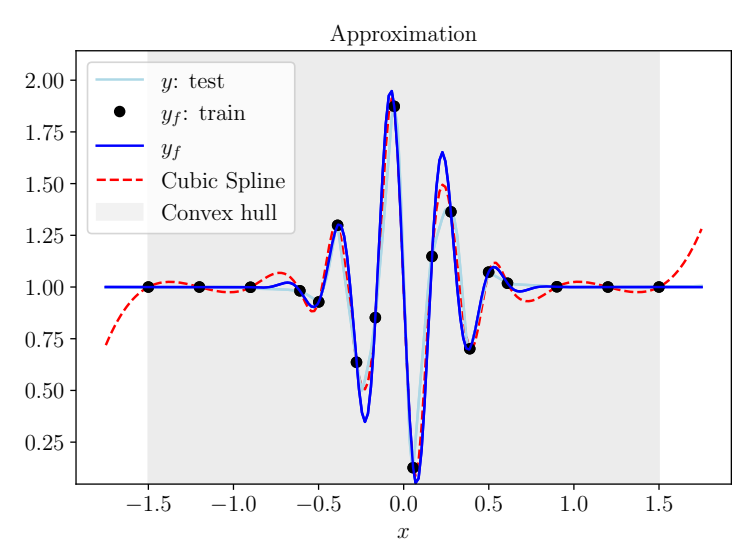
- Recently shown the optimizers penalize Sobolev norm: Ma, C., Ying, L. (2021)

Smooth interpolation



Smooth interpolation: Comparison with cubic splines

back



Smooth interpolation: A simple dynamical system

Consider the following system

$$K_{t+1} = \eta K_t.$$

This system have the following solutions

$$K(t) = K_0 \eta^t.$$

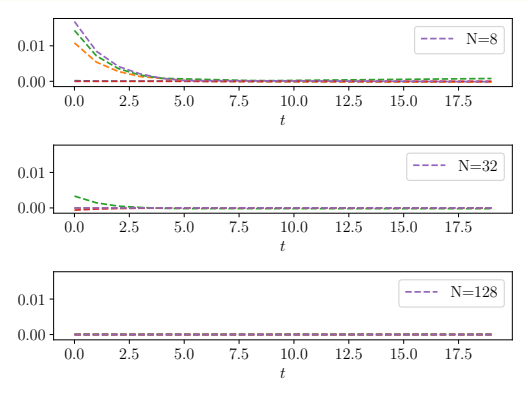
- Without specifying the initial condition, K_0, this is an ill-defined problem, i.e. there are infinity many solutions.
- The solution to:

$$\begin{aligned} \min_{K \in \mathcal{H}} \quad & \|K\|_S \\ \text{s.t.} \quad & K(t+1) - \eta K(t) = 0 \quad \text{for } t = t_1, \ldots, t_N \end{aligned}$$

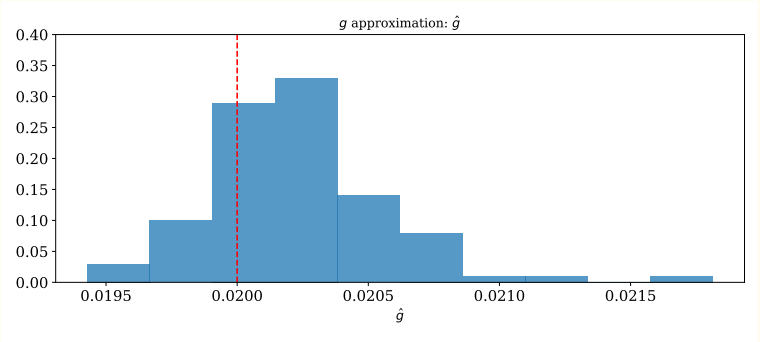
  is $K(t) = 0$.

Smooth interpolation: A simple dynamical system results

Three layers deep neural network, for $N = 8$, 32, and 128. Each trajectory corresponds to different random initialization of the optimization procedure (seed).

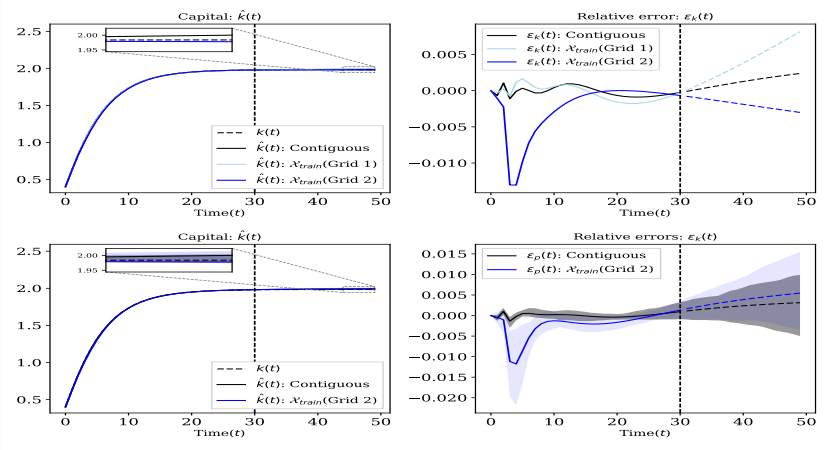


Learning the growth rate



The histogram for approximate growth rate over 100 seeds. back

Contiguous vs. dense grid



- $\mathcal{X}_{\text{train}}(\text{Grid 1}) = [0, 1, 2, 4, 6, 8, 12, 16, 20, 24, 29]$, $\mathcal{X}_{\text{train}}(\text{Grid 2}) = [0, 1, 4, 8, 12, 18, 24, 29]$.
- Contiguous grid : $\mathcal{X}_{\text{train}} = [0, 1, 2, ..., 29]$. back

Neoclassical growth with multiple steady-states: where things fail