# 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]

May 22, 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**

- Most dynamic models require **economic assumptions** eliminating explosive solutions (transversality, no-bubble, no-ponzi schemes, Blanchard-Khan conditions, etc.):
  - These are variations on "boundary conditions" for **forward looking** behavior of agents.
  - Without those economic conditions, problems are not well-posed and have multiplicity.
  - Deterministic, stochastic, sequential, recursive formulations all require conditions in some form.
- Steady states/forward-looking boundary conditions are the key limitation on increasing dimensionality:
  - Otherwise, researchers routinely solve initial value problems with millions of equations.
  - Equivalently, conditions for recursive formulations manifest as requiring accurate solution on the entire domain, even though one may only care about the solution from a single initial condition.
- **Key trade-off:** Can we avoid precisely calculating a stationary distribution/steady-state/... –which is never reached– and still have accurate short/medium-run dynamics disciplined by transversality/etc.?

# This paper

- Show –**numerically**– that **deep learning** solutions to many dynamic, forward looking, models automatically fulfill the long run boundary conditions we need (transversality, no-bubble, etc.).
- 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**.
- Only empirical, but still provide theoretical intuition on why these results hold: the deep learning theory is not quite ready to formally prove everything in this environment.
- Suggests these methods may solve higher-dimensional problems while avoiding the key computational limitation—with the only trade-off being loss of precision for equilibria after "we are all dead."

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

Many theoretical models 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. asset price, investment choice, best-response, etc.).
- Domain $\mathcal{X}$ could be state (e.g. dividends, capital, opponents state) or time if sequential.
- The "model" is $\ell : \mathcal{F} \times \mathcal{X} \to \mathcal{R}$ (e.g., Euler and Bellman residuals, equilibrium FOCs).
- Normalize so that a solution is the "zero" of the residuals, i.e. $\mathbf{0} \in \mathcal{R}$, at each $x \in \mathcal{X}$.

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

# Example: one formulation of neoclassical growth

- Capital, $k$, consumption $c$, utility $u(c)$, discount rate $\beta$, depreciation $\delta$, production function $f(k)$.
- Domain: $x = \begin{bmatrix} k \end{bmatrix}$ and $\mathcal{X} = \mathbb{R}_+$.
- Solve for $k'(\cdot)$ and $c(\cdot)$: So $f : \mathbb{R} \to \mathbb{R}^2$ and $\mathcal{Y} = \mathbb{R}^2_+$.
- Skipping to lagrangian... residuals are the euler equation and feasibility, so $\mathcal{R} = \mathbb{R}^2$:

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

- Finally, a solution if the $f^*$ which has zero residuals on domain $\mathcal{X}$.

# How to solve globally?

- Find approximate $\hat{f}$ which only holds approximately on $\mathcal{X}$.
- Choose a class of approximate solutions which aligns with $\ell$ and $\mathcal{F}$ solutions.
- Potentially bound or emphasize precision in regions of economic interest in $\mathcal{X}$.

# Interpolation solutions for solving functional equations

Classic approach: use class of functions with finite parameters and interpolate a finite number of points

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 parameters $\Theta \subseteq \mathbb{R}^M$ (e.g., polynomials, splines).
3. **Fit** with nonlinear least-squares for a general $M \gtreqless N$

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

   - If $M = N$ and $\mathcal{H}(\Theta)$ functions span $\mathbb{R}^N$ can solve nonlinear system (e.g., Chebyshev collocation).
   - 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** $\mathcal{X}_{\text{train}}$.

4. **Hope** that $\hat{f}(x;\theta) \approx f^*(x)$ for $x \in \mathcal{X} \setminus \mathcal{X}_{\text{train}}$. i.e., has low **generalization error**:
   - For $M \geq N$ we usually interpolate exactly ( and hence $\hat{f}(x;\theta) \approx f^*(x)$ for $x \in \mathcal{X}_{\text{train}}$).
   - In practice, we tinker with $\mathcal{H}, \Theta$ and $\mathcal{X}_{\text{train}}$ until error no longer seems to be an issue.
   - More generally, our goal is to minimize $\|\hat{f}(\cdot;\theta) - f^*\|_s$.

**Deep learning** here just enables "pick, choose, fit, hope" with more flexibility using **economic insights**.

# "Modern" ML is massively overparameterized

**Deep learning** here is **highly-overparameterized** $\mathcal{H}$ (i.e. $M \gg N$) designed for good generalization:

- Complete flexibility in the choice of $\mathcal{H}$ from economic insights on problem structure $\ell$ and $\mathcal{F}$:
- Composing $\mathcal{H}$ from multiple functions (e.g., "deep"er) tends to generalize better in practice.
- For example, if $f : \mathbb{R}^Q \to \mathbb{R}$ could choose $\hat{f}(x;\theta) \equiv W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2$:
  - $W_1 \in \mathbb{R}^{P \times Q}$, $b_1 \in \mathbb{R}^P$, $W_2 \in \mathbb{R}^P$, and $b_2 \in \mathbb{R}$.
  - $\sigma(\cdot) = max(0, \cdot)$ element-wise (i.e. ReLU activation in CS literature) but many others.
  - $\theta \in \Theta \equiv \{b_1, W_1, b_2, W_2\}$ and $M = PQ + P + P + 1$.
  - Choose a big $P$...or add another "layer": $\hat{f}(x;\theta) \equiv W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2) + b_3$.
  - Let's generically call these $NN(\theta)$ and explain structure when it matters.
- Software (e.g., PyTorch) makes it easy to experiment with different $\mathcal{H}$ (i.e., neural networks), manage the $\theta$, and **get gradients required for optimization methods**.
  - But otherwise, the method and objective is the same as before, i.e. $\min_{\theta \in \Theta} \sum_{x \in \mathcal{X}_{\text{train}}} \ell(\hat{f}(\cdot;\theta), x)^2$.

# Deep learning optimizes in a space of functions

- Since $M \gg N$ has an enormous number of solutions (e.g., $\theta_1$ and $\theta_2$),
  1. Agree only on "data": $\hat{f}(x; \theta_1) \approx \hat{f}(x; \theta_2)$ for $x \in \mathcal{X}_{\text{train}}$.
  2. Agree everywhere: $\hat{f}(x; \theta_1) \approx \hat{f}(x; \theta_2)$ for $x \in \mathcal{X}$. Alternatively, $||\hat{f}(\cdot; \theta_1) - \hat{f}(\cdot; \theta_2)||_S \approx 0$.

- Since individual $\theta$ are irrelevant it is helpful to think of optimization directly within $\mathcal{H}$

$$\min_{\hat{f} \in \mathcal{H}} \sum_{x \in \mathcal{X}_{\text{train}}} \ell(\hat{f}, x)^2 \tag{1}$$

- Since $M \gg N$, $\hat{f}$ always interpolates and the objective value in (1) will always be $\approx 0$.

# 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 "data").
- **Mental model:** chooses min-norm interpolating solution for a (usually) unknown functional norm $S$

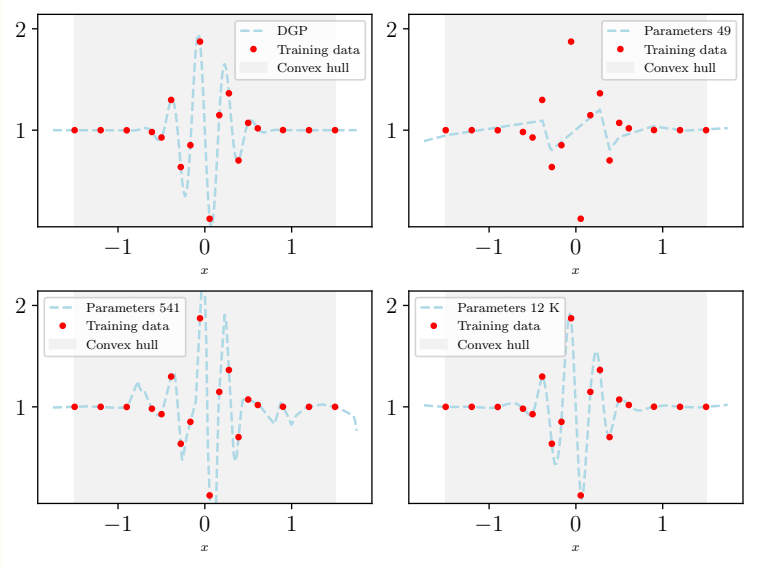
$$\min_{\hat{f}\in\mathcal{H}} ||\hat{f}||_S$$
$$\text{s.t. } \ell(\hat{f}, x) = 0, \qquad \text{for } x \in \mathcal{X}_{\text{train}}$$

  - CS and literature refers to this as the **inductive bias**: optimization process biased towards particular $\hat{f}$,
  - Characterizing $S$ (e.g., Sobolev?) is an active research area in CS at the heart of deep learning theory.
  - Intuition is that it may choose the interpolating solutions which are flattest and have smallest derivatives.
- Is $\|\hat{f} - f^*\|_S$ small (i.e., does the min-norm solution generalize well)? Depends on $G, \mathcal{H}, \mathcal{X}_{\text{train}}$.

**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/etc. in dynamic models—and hence are disciplined by long run boundary conditions.

# Smooth interpolation

Examples



# 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 with/without a no-bubble condition.
2. Sequential formulation of the neoclassical growth model with transversality condition.
3. Equivalent for a recursive formulation of the neoclassical growth model.
4. (In paper many more: e.g. what if non-stationary? BGPs? etc.).

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, $y_t$, take $y_0$ as given and follow 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 $\beta(1+g) < 1$

$$p_t^f \equiv \sum_{j=0}^{\infty} \beta^j y_{t+j} = G(I - \beta A)^{-1} x_t$$

# Recursive formulation

With standard transformation, all solutions $p_t^f$ fulfill the recursive equations

$$p_t = Gx_t + \beta p_{t+1} \tag{2}$$

$$x_{t+1} = Ax_t \tag{3}$$

$$0 = \lim_{T \to \infty} \beta^T p_T \tag{4}$$

$$x_0 \text{ given} \tag{5}$$

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

- The boundary condition (4) is an **assumption** necessary to be well-posed and have unique solutions
- It ensures that $p_t = p_t^f$ by imposing long run boundary on forward-looking behavior
- But without this assumption there can be "rational bubbles" with $p_t \neq p_t^f$, but fulfilling (2) and (3)
- 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., (5) not enough on its own)

# Solutions without a "no-bubble conditions"

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

$$p_t = p_t^f + \zeta\ \beta^{-t}. \tag{6}$$

- For any $\zeta \geq 0$. The $x_t$ initial condition determined the $p_t^f$ solution.
- The "no bubble condition" chooses the $\zeta = 0$ solution.

Lets analyze this with a "deep learning" solution, first by imposing the no-bubble condition.

# Interpolation formulation

Write $p_t$ as $p(t)$ to allow interpolation between sparse $t \in \mathcal{X}_{\text{train}}$ points and collect (2) to (4)

$$\min_{p \in \mathcal{H}} \quad \|p\|_S \tag{7}$$

$$\text{s.t.} \quad p(t) - Gx(t) - \beta p(t+1) = 0 \quad \text{for } t \in \mathcal{X}_{\text{train}} \tag{8}$$

$$0 = \lim_{T \to \infty} \beta^T p(T) \tag{9}$$

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 (3)

- Recall: generalization comes from design of $\mathcal{H}$ and optimizer; not only model, i.e., (8) and (9).
- The 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 (9) bind?
- In this case, we know the interpolating solutions to (8) without imposing (9)

$$p(t) = p^f(t) + \zeta\beta^{-t} \tag{10}$$

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

$$0 \leq \|p\|_S \leq \|p^f\|_S + \zeta \, \|\beta^{-t}\|_S \tag{11}$$

- Relative to classic methods the "deep learning" problem now has a $\|p\|_S$ objective!
  - From (11) 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 (9) is satisfied at the optima.
- What types of norms $\|p\|_S$ would $\mathcal{H}$ and optimization induce? CS theory suggests variations Sobolev

# Minimum norm formulation

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

$$\min_{p \in \mathcal{H}} \quad \|p\|_S \tag{12}$$

$$\text{s.t.} \quad p(t) - Gx(t) - \beta p(t+1) = 0 \quad \text{for } t \in \mathcal{X}_{\text{train}} \tag{13}$$

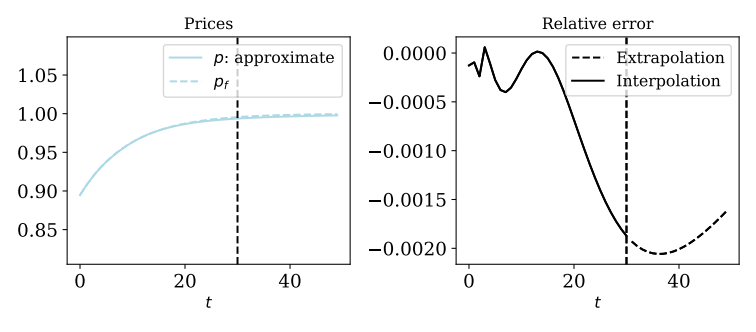
A reminder: in practice, given the $\mathcal{X}_{\text{train}}$, we directly implement this as $p(\cdot;\theta) \in \mathcal{H}(\Theta)$ and fit with

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{X}_{\text{train}}|} \sum_{t \in \mathcal{X}_{\text{train}}} \left[p(t;\theta) - Gx(t) - \beta p(t+1;\theta)\right]^2 \tag{14}$$

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

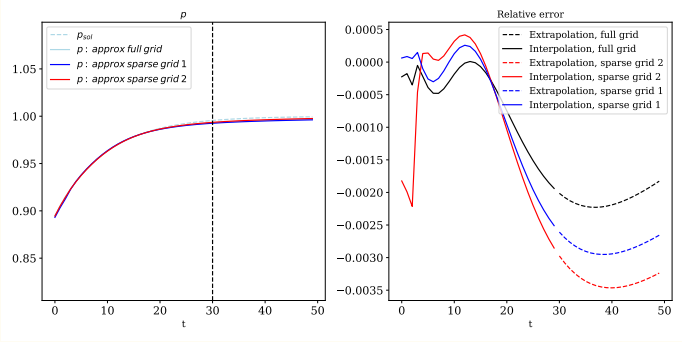
- The $\mathcal{X}_{\text{train}}$ does not need to have contiguous $t$ and $|\mathcal{X}_{\text{train}}|$ may be relatively small
- Most important: no steady state calculated, nor large $T \in \mathcal{X}_{\text{train}}$ required

# Results



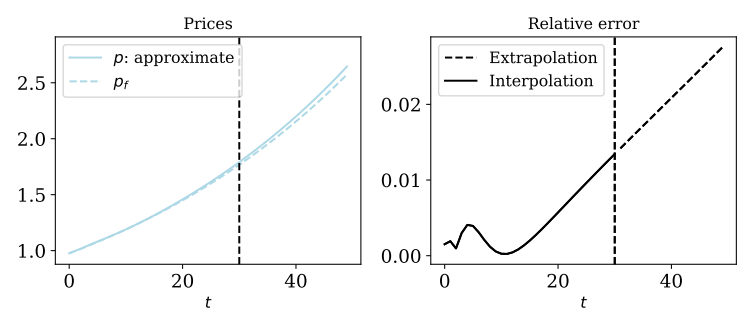

1. **Pick** $\mathcal{X}_{\text{train}} = [0, 1, 2, .., 30]$ and $t > 30$ is "extrapolation" where $c = 0.01$, $g = -0.1$, and $y_0 = 0.8$
2. **Choose** $p(t;\theta) = NN(t;\theta)$ where "NN" has 4 hidden layers of 128 nodes. $|\theta| = 49.9K$ parameters.
3. **Fit** using L-BFGS and PyTorch in just a **few seconds**. Could use Adam/SGD/etc.
4. **Pray**'ers were answered, even without imposing no-bubble condition. Compare to analytic $p^f(t)$
   - Relative error $\equiv \frac{p(t)-p^f(t)}{p^f(t)}$ ranging from 0.0007% for $t = 0$ to 0.02% when extrapolating.
   - These long run errors don't affect the short-run accuracy (still small, even after we are all dead)

# Contiguous vs. dense grid




- $\mathcal{X}_{\text{train}}(\text{Grid 1}) = [0, 1, 2, 4, 6, 8, 12, 16, 20, 24, 30]$ and $\mathcal{X}_{\text{train}}(\text{Grid 2}) = [0, 1, 4, 8, 12, 18, 24, 30]$
- Small errors even with 8 data points (and $\approx$ 40K parameters). Even $< 0.035\%$ after we all are dead
- Contrary to popular wisdom about deep learning only being appropriate in high "data" environments
  - Can use **less "data"** relative to alternatives

# 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** $p(t;\theta) = e^{\phi t} NN(t;\theta_{NN})$ where $\theta \equiv \{\phi, \theta_{NN}\} \in \Theta$ is the parameter vector
  - 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!

# Neoclassical growth in sequence space

# Sequential formulation (with a possible BGP)

$$
\begin{aligned}
\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} \quad & \sum_{t=0}^{\infty} \beta^t u(c_t) \\
\text{s.t.} \quad & k_{t+1} = z_t^{1-\alpha} f(k_t) + (1-\delta) k_t - c_t \\
& z_{t+1} = (1+g) z_t \\
& k_t \geq 0 \\
& 0 = \lim_{T \to \infty} \beta^T u'(c_T) k_{T+1} \\
& k_0, z_0 \text{ given}
\end{aligned}
$$

- 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 $z_t$
- 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 formulation

$$\min_{q \equiv [\,k\ c\,] \in \mathcal{H}} \quad \|q\|_S \tag{15}$$

$$\text{s.t.} \quad u'(c(t)) = \beta u'(c(t+1))\left[z(t+1)^{1-\alpha} f'(k(t+1)) + 1 - \delta\right] \quad \text{for } t \in \mathcal{X}_{\text{train}} \tag{16}$$

$$k(t+1) = z(t)^{1-\alpha} f(k(t)) + (1-\delta)k(t) - c(t) \quad \text{for } t \in \mathcal{X}_{\text{train}} \tag{17}$$

$$k(0) = k_0 \tag{18}$$

$$0 = \lim_{T \to \infty} \beta^T u'(c(T))k(T+1) \tag{19}$$

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

- **Choose** now requires both $k$ and $c$ or one function $q : \mathbb{R} \to \mathbb{R}^2$ where $q(t) \equiv [\,k(t)\ c(t)\,]$
  - Easiest is $q(t;\theta) = NN(t;\theta)$ where $q : \mathbb{R} \to \mathbb{R}^2$. But PyTorch makes separate $k, c$ easy as well
  - Also, $k(t) \geq 0$ and $c(t) \geq 0$ built directly into $\mathcal{H}$.
- **Fit** Minimize the residuals on $\mathcal{X}_{\text{train}}$ for sum of (16) to (18)
  - Is the transversality condition (19) needed? **Severe multiplicity** previously without (15)

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

Sketch of the proof:

- Let $q(t) = \{k(t), c(t)\}$ be the optimal solution.
- Let $\tilde{q}(t) = \{\tilde{k}(t), \tilde{c}(t)\}$ be a solution that satisfies all the equations **except** transversality condition (19).

There are two possible cases:

1. $\tilde{k}(t+1) \to \infty$, and $k \to k_{ss} = \left(\frac{\beta^{-1}+\delta-1}{\alpha}\right)^{\frac{1}{\alpha-1}}$. Therefore any norm that measures curvature or level of a function

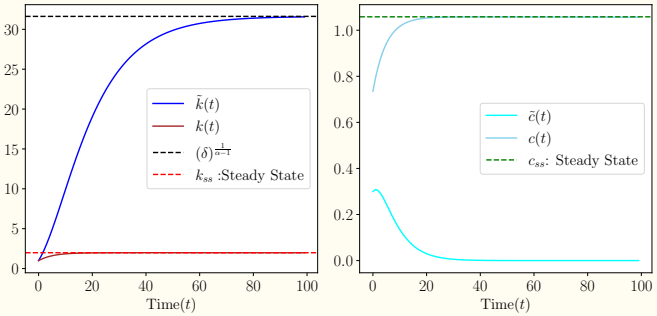
$$0 \leq \|k\|_S \leq \|\tilde{k}\|_S$$

2. $\tilde{c}(t)$ approaches zero.
   - $\tilde{k}(t)$ approaches $(\delta)^{\frac{1}{\alpha-1}} \gg k_{ss} = \left(\frac{\beta^{-1}+\delta-1}{\alpha}\right)^{\frac{1}{\alpha-1}}$
   - Both $k(t)$, and $\tilde{k}(t)$ are monotone. Therefore any norm that measures curvature or level of a function

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

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

Example: the violation of the transversality condition:




- The solution that violate the transversality are associate with **"big"** $k(t)$
- Make sure **explosive/big variables** are included in $q(\cdot : \theta)$
- If explosive/big variables are **not** included, the solutions violate the transversality condition.

# Minimum norm formulation

Any solution that violates the transversality condition has a **bigger norm** than the optimal solution. Therefore, the transversality condition becomes **redundant**.

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

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

# Minimum norm formulation

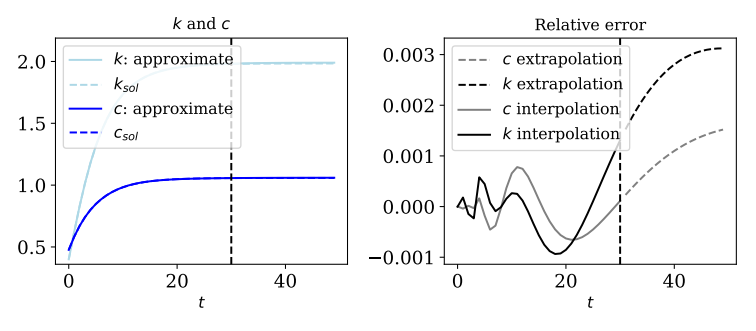
In practice, given the $\mathcal{X}_{\text{train}}$, we directly implement this as $q(\cdot;\theta) \in \mathcal{H}(\Theta)$ and fit with

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

Given $z(0)$, $z(t)$ for $t \in \mathcal{X}_{\text{train}}$ is generated by recurrence $z(t+1) = (1+g)z(t)$

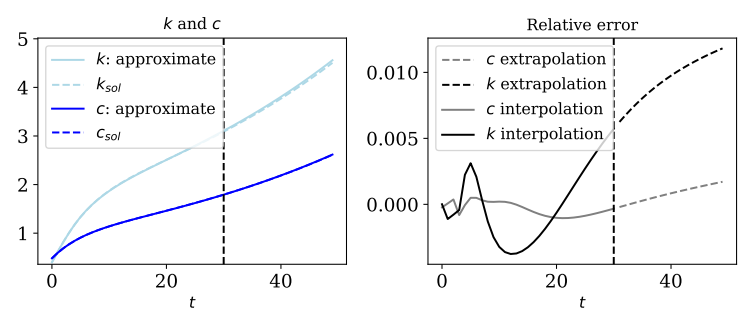
- The $\mathcal{X}_{\text{train}}$ does not need to have contiguous $t$ and $|\mathcal{X}_{\text{train}}|$ may be relatively small ⏩ Sparse
- Most important: no steady state calculated, nor large $T \in \mathcal{X}_{\text{train}}$ required

# 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** $q(t; \theta) = NN(t; \theta)$ where "NN" has 4 hidden layers of 128 nodes. $|\theta| = 49.9K$ parameters.
3. **Fit** using L-BFGS and PyTorch in just a **few seconds**.
4. **Pray**'ers were answered, 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** $q(t;\theta) = e^{\phi t} NN(t;\theta_{NN})$ where $\theta \equiv \{\phi, \theta_{NN}\} \in \Theta$ is the parameter 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!

# Recursive version of the neoclassical growth model

# The neoclassical growth model (with a possible BGP)

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

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

- 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 formulation

$$\min_{k' \in \mathcal{H}} \quad \|k'\|_S \tag{20}$$

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

$$\big[((1+g)z)^{1-\alpha} f'(k'(k,z)) + 1 - \delta\big] \quad \text{for } (k,z) \in \mathcal{X}_{\text{train}} \tag{21}$$

$$0 = \lim_{T\to\infty} \beta^T u'(c(T))k(T+1) \quad \text{for all } (k_0,z_0) \in \mathcal{X}_{\text{train}} \tag{22}$$

where

$$c(k,z;k') \equiv z^{1-\alpha} f(k) + (1-\delta)k - k'(k,z) \tag{23}$$

- **Choose** now $k' : \mathbb{R}^2 \to \mathbb{R}$ , $k'(k,z;\theta) = NN(k,z;\theta)$ , $k'(k,z) \geq 0$ built direclty into $\mathcal{H}$
- **Fit** Minimize the residuals on $\mathcal{X}_{\text{train}}$ for sum of (21)
  - Is the transversality condition (22) needed? **multiplicity** happens without it.

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

Still working on the proof, however the idea at the moment is

- For a fixed period of time $T$, and fixed capital $k_0$ any function the violates the transversality has to have **larger derivatives** to back up the growth to $\left(\delta\right)^{\frac{1}{1-\alpha}}$ and consequently larger norm.
- Minimizing norms like Sobolev that measures big derivatives should get rid of the solutions that violate the transversality condition.

# Minimum norm formulation

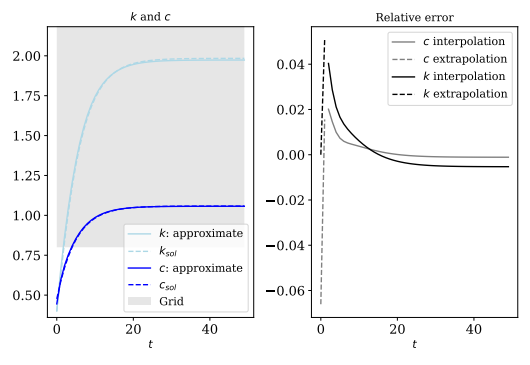
$$\begin{aligned} \min_{k' \in \mathcal{H}} \quad & \|k'\|_S \\ \text{s.t.} \quad & u'\Big(c\big(k, z; k'\big)\Big) = \beta u'\Big(c\big(k'(k,z), (1+g)z; k'\big)\Big) \times \\ & \Big[((1+g)z)^{1-\alpha} f'(k'(k,z)) + 1 - \delta\Big] \quad \text{for } (k,z) \in \mathcal{X}_{\text{train}} \end{aligned}$$

Where $c(\cdot)$ is defined via equation (23).

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

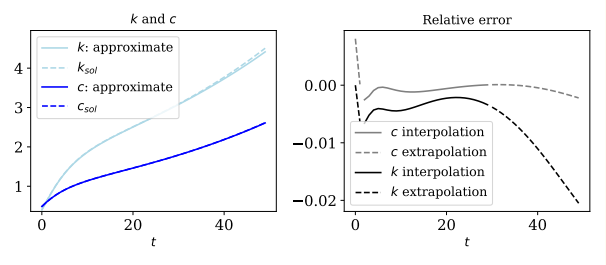
$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{X}_{\text{train}}|} \sum_{(k,z) \in \mathcal{X}_{\text{train}}} \Bigg[ - u'\Big(c\big(k, z; k'(.;\theta)\big)\Big) + \beta u'\Big(c\big(k'(k,z;\theta), (1+g)z; k'(.;\theta)\big)\Big) \times \Big[((1+g)z)^{1-\alpha} f'(k'(k,z);\theta) + 1 - \delta\Big] \Bigg]^2$$

# Results




1. **Pick** $\mathcal{X}_{\text{train}} = [0.8, 2.5] \times \{1\}$ and $k_0 = 0.4 \notin \mathcal{X}_{\text{train}}$ is "extrapolation" $\alpha = \frac{1}{3}$, $\sigma = 1$, $\beta = 0.9$, $g = 0.0$
2. **Choose** $k'(k, z; \theta) = NN(k, z; \theta)$ where "NN" has 4 hidden layers of 128 nodes. $|\theta| = 49.9K$ parameters.
3. **Fit** using L-BFGS and PyTorch in just a few seconds.

# 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.
- **Extraordinary** extrapolation from both sides the grid for capital. robustness

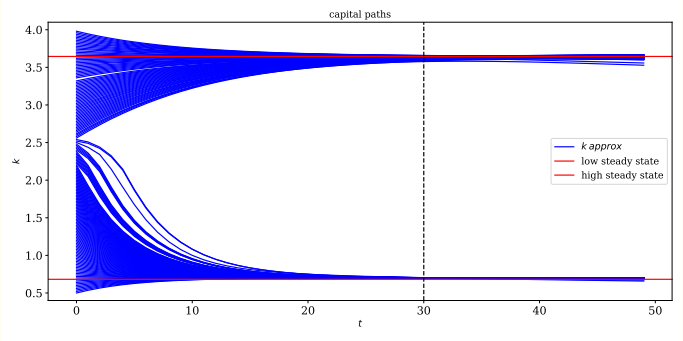
# The neoclassical growth model with multiple steady states

# 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.

# Results




1. **Pick** $\mathcal{X}_{\text{train}} = [0, 1, 2, .., 30]$ and $t > 30$ is "extrapolation" $\alpha = \frac{1}{3}$, $\sigma = 1$, $\beta = 0.9$, $g = 0.0$, $a = 0.5$, $b_1 = 3.0$, and $b_2 = 2.5$, for 100 different initial conditions in $[0.5, 4.0]$.
2. **Choose** $q(t; \theta) = NN(t; \theta)$ where "NN" has 4 hidden layers of 128 nodes. $|\theta| = 49.9K$ parameters.
3. **Fit** using L-BFGS and PyTorch, each in just a few seconds.
4. **Pray**'ers were answered, even without imposing the transversality condition.

# Conclusion

# Conclusion

- Solving functional equations with deep learning is an extension of collocation/interpolation methods
- With **massive overparameterization** optimizers tend to choose those interpolating functions which are not explosive and with smaller gradients (i.e., **inductive bias**)
- In practice, those solutions **automatically** fulfill **forward-looking** assumptions (e.g. transversality)
  - e.g. in growth models, deep learning loves to take the "turnpike" even if fuzzy on when to exit
- If we solve models with deep-learning without (directly) imposing long run boundary conditions
  - Can use very few "grid" points and **avoids calculating steady-states** and recursive equivalents
  - 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 if **non-stationarity** and **steady-state multiplicity**
- Exploiting key trade-off: give up accuracy globally and at steady state for better transition dynamics
  - Gives hope for solving high-dimensional models still disciplined by forward looking economic assumptions
  - With ML frameworks (e.g., PyTorch) these methods are robust & easier to implement than alternatives

# Appendix

# Sobolev semi-norms

back

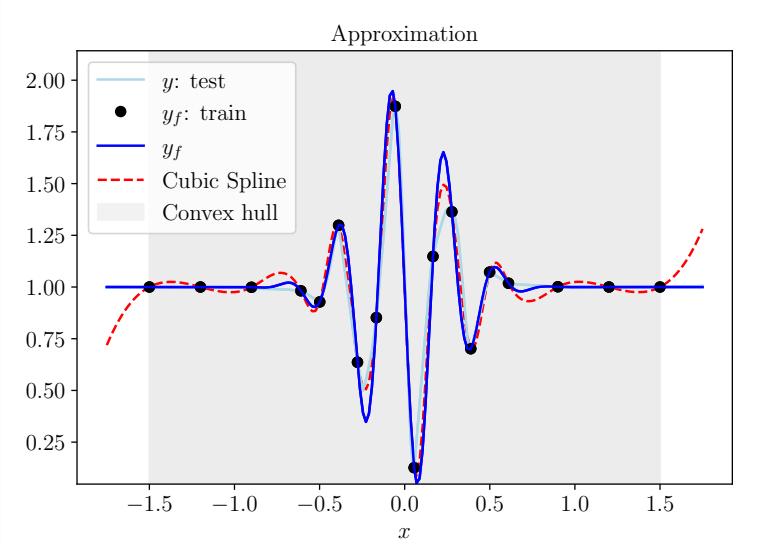
Let $f : \mathcal{X} \to \mathbb{R}$

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

- Recently shown the optimizers penalize Sobolev norm: Ma, C., Ying, L. (2021)
- See you in the econometrics lunch

# 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:

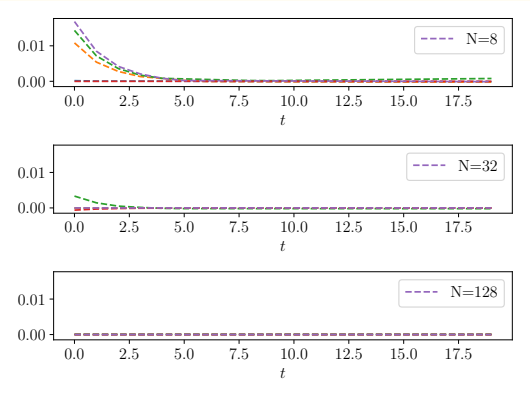
$$\begin{aligned} \min_{K \in \mathcal{H}} \quad & \|K\|_S \\ \text{s.t.} \quad & K(t+1) - \eta K(t) = 0 \quad \text{for } t = t_1, \ldots, t_N \end{aligned}$$

  is $K(t) = 0$.

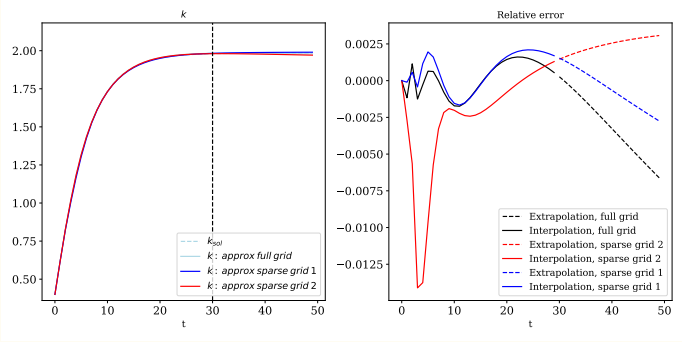
# Smooth interpolation: A simple dynamical system results

back

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



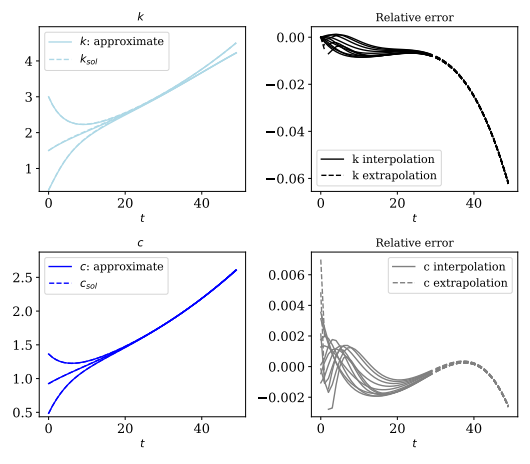
# Sparse vs. dense grid




Left panel: Capital for three different $\mathcal{X}_{\text{train}}$. Right panel: Relative errors, dashed line relative errors in the extrapolation region.

- Full grid : $\mathcal{X}_{\text{train}} = [0, 1, 2, ..., 30]$(Black).
- Grid 1: $\mathcal{X}_{\text{train}} = [0, 1, 2, 4, 6, 8, 12, 16, 20, 24, 30]$(Blue).
- Grid 2: $\mathcal{X}_{\text{train}} = [0, 1, 4, 8, 12, 18, 24, 30]$(Red).

# Sparse vs. dense grid




10 different $k_0 \in [0.4, 3.5]$. Left panel: trajectories for 3 different initial conditions. Right panel: relative errors for all 10 different trajectories. back