Discuss this article at JournalTalk:
https://journaltalk.net/articles/n6225/

EJW
ECON JOURNAL WATCH
Scholarly Comments on Academic Economics

ECON JOURNAL WATCH 23(3)
September 2026: 343–361

Deep Learning as a Projection Method for Solving Economic Models[1]

Kenneth L. Judd[2] Karl Schmedders[3]

LINK TO ABSTRACT

The use of deep neural networks to solve dynamic economic models is increasingly described as a new paradigm, set against (and superseding) the classical projection methods of numerical economics. The most prominent and careful statement of this view is the survey of Fernández-Villaverde (2026),[4] which frames the development as a contest between deep learning and classical projection methods.[5] Neural networks, in the survey's words, "take a different approach: Instead of prescribing basis functions, they learn them," and the exposition is organized into "fixed bases (polynomials) and learned bases (neural networks)," the latter presented as overcoming a curse of dimensionality attributed to classical techniques.

[1] We are grateful to three anonymous referees for helpful feedback on earlier versions and to Dave Brooks for excellent editorial support.

[2] Hoover Institution, Stanford, CA 94305 USA. Email: judd@stanford.edu.

[3] International Institute for Management Development, Chemin de Bellerive 23, P.O. Box 915, CH-1001 Lausanne, Switzerland. Email: karl.schmedders@imd.org.

[4] A chronological clarification of the recent literature reviewed by the survey: Maliar et al. (2019) circulated as CEPR Discussion Paper 14024 and appeared as Maliar et al. (2021); Azinovic et al. (2019) appeared as Azinovic et al. (2022); and Fernández-Villaverde et al. (2020), dated September 2020, remains unpublished. The survey states, "Fernández-Villaverde et al. (2020), Maliar et al. (2021) and Azinovic et al. (2022)," combining the manuscript year of the first and publication years of the later two; by date of first circulation the two later papers come first, an ordering the 2020 manuscript itself reflects in citing them as prior work.

[5] Throughout, *projection methods* denotes the numerical-analysis sense of Judd (1992), the weighted-residual methods (Galerkin, collocation, least squares) for solving functional equations, rather than the data-science usage in which "projection" means dimensionality reduction such as principal component analysis. The survey itself cautions that neural networks "should not be confused with dimensionality-reduction techniques such as principal component analysis." Our claim concerns deep learning as used to solve dynamic models, where the goal is to drive an equilibrium residual to zero, and not to compress data.

We use *machine learning versus projection methods* as shorthand for this framing, not as a quotation. We take that survey as our running example because it is lucid and will be widely read; our argument, however, is with the framing it crystallizes and not with the survey as such. The exposition is careful and the enthusiasm understandable. But the organizing contrast is miscast, and the miscasting propagates into several claims about what neural networks do, why they work, and when they should be preferred.

Our purpose is not to dispute that neural networks are useful for some high-dimensional problems; they are. It is to state, as clearly as we can, three misunderstandings embedded in the "ML vs. projection" framing, and to locate the one place where a genuine difference lives. The argument is almost entirely definitional, and that is the point: once the methods are written down side by side, the dichotomy disappears, and with it the case for treating deep learning as a new paradigm rather than as a new approximating family inside an old one, namely the method of weighted residuals (Judd 1992; 1998).

A clarification before we begin, because our claim is deliberately narrow. The survey makes a genuine and valuable contribution, and it is a *practical* one. Deep-learning frameworks bring automatic differentiation, mature open-source libraries, and GPU parallelism to a community that has too often had to hand-build sparse tensor grids and bespoke quadrature rules; for many problems, being able to write down a residual and let an autodiff engine and a graphics card do the rest is a real gain in convenience and scale. Our target is not this toolkit, but the *conceptual* framing wrapped around it. That deep learning is a convenient new way to execute a method economists already possessed is correct and useful, but claiming that it is a "new paradigm" superseding projection is a misclassification. One can accept the first claim and reject the second, and we do. Framing the development as a contest between competing paradigms creates an artificial dichotomy that obscures practical trade-offs. The danger is that researchers may reflexively reach for the "new paradigm" under the impression that it universally supersedes the old, neglecting the predictable costs in precision, speed, and numerical verification that the adaptive parameterization entails in many standard economic regimes.

We proceed by stating the common program, then three misunderstandings of the new- paradigm view and a worked illustration on a canonical growth model, before drawing the constructive implication: deep learning is a single toolkit to be built and used well, and not a choice between paradigms.

The Common Program

As Fernández-Villaverde (2026) states, "When solving equilibrium models, we must approximate unknown functions." Let the unknown object of a model be

a function $f : \mathcal{X} \to \mathbb{R}^d$ on a state space $\mathcal{X} \subseteq \mathbb{R}^N$; it might be a value function, a policy function, a conditional expectation, or a pricing kernel. The model's equilibrium conditions define an operator $\mathcal{T}$ (a Bellman, Euler, or Hamilton–Jacobi–Bellman equation, possibly with complementarity conditions) whose solution $f^\star$ satisfies

$$\mathcal{T}[f^\star](X) = 0 \qquad \text{for all } X \in \mathcal{X}.$$

Write the residual $\mathcal{R}[g](X) := \mathcal{T}[g](X)$ of a candidate g. Choose a finite set of evaluation points $\{X_\ell\}_{\ell=1}^{L} \subset \mathcal{X}$ (grid nodes, collocation points, or simulated states) and a weighting $\{\nu_\ell\}$.

The general framework is the **method of weighted residuals**: pick a parametric family g_θ and impose finitely many restrictions on the residual $\mathcal{R}[g_\theta]$ against a set of test functionals (which often computationally reduces to restrictions at evaluation points). Its classical members are distinguished by the choice of test functional in $\langle \psi_j, \mathcal{R}[g_\theta] \rangle = 0$: *collocation* ($\psi_j = \delta_{X_j}$) sets the residual to zero at selected nodes; *Galerkin* ($\psi_j = \phi_j$) makes it orthogonal to a test space, and least squares ($\psi_j = \partial \mathcal{R} / \partial \theta_j$) provides the first-order conditions for minimizing a squared residual norm (Judd 1992). These are different members of one family, not different families.

The neural-network solvers emphasized in the survey use the *least-squares* member, typically with Monte Carlo evaluation points, chosen because the resulting average loss suits stochastic-gradient methods such as plain SGD or adaptive variants such as Adam (Kingma and Ba 2015; Goodfellow et al. 2016). Their numerical program is therefore

$$\theta^\star \in \arg\min_{\theta} \sum_{\ell=1}^{L} \nu_\ell \left\| \mathcal{R}[g_\theta](X_\ell) \right\|^2, \tag{1}$$

the least-squares projection program with the points $\{X_\ell\}$ sampled by Monte Carlo, a special case of the family that classical projection methods already inhabit, not a new genus.[6]

Once the residual criterion, the evaluation points, and the weighting are fixed, the remaining primitive is the approximating family. The two candidates are the following.

[6] We credit the origin of this construction as being outside economics: minimizing the squared residual of a differential or functional equation with a neural-network ansatz is the deep Galerkin method of Sirignano and Spiliopoulos (2018) and the physics-informed neural networks of Raissi et al. (2019), both of which precede its adoption for solving economic models. We cite them because the approach is theirs; our contribution is not the method, but its placement within the economist's projection toolkit and the comparison that follows.

Projection (linear basis). A linear combination of prescribed functions,

$$g_\theta(X) = \sum_{m=1}^{M} \theta_m \phi_m(X), \qquad \{\phi_m\} \text{ prescribed ex ante.} \tag{2}$$

Network (adaptive parameterization). A linear combination of trained functions,

$$g_\theta(X) = v_0 + \sum_{m=1}^{M} v_m \psi_m^\theta(X), \qquad \psi_m^\theta(X) := \varphi(w_m^\top X). \tag{3}$$

Here $\theta = (v_0, v, W)$ is the vector of parameters, and $w_m^\top$ denotes the transposed m-th column of the weight matrix W. We adopt the standard convention of augmenting the state X with a constant 1 so that $w_m \in \mathbb{R}^{N+1}$ absorbs the hidden-layer biases.

While we express a single-hidden-layer network here to map cleanly to classical approximation bounds, our argument applies identically to deep, multi-layer architectures where $g_\theta(X)$ is a nested composition of functions. Regardless of layer depth, the mapping $\theta \mapsto g_\theta$ remains a nonlinear parameterization, and its integration into Program (1) remains structurally unchanged.

Written as in (3), the network forms its approximation as an affine combination of the functions ψ_m^θ, that is, *the same algebra as* (2), with one structural difference: in (2) these functions are a linear basis, so $\partial g / \partial \theta_m = \phi_m(X)$ does not depend on θ and the approximation set is a flat M-dimensional linear subspace; in (3) the functions ψ_m^θ deform with the hidden weights W, so the approximation set is a nonlinear manifold.

Observation (Structural Identity)**.** *Fixing the residual $\mathcal{R}$, the least-squares criterion, the evaluation points $\{X_\ell\}$, and the weighting $\{v_\ell\}$, the least-squares projection solver and the neural-network solver are the* same *program* (1)*, restricted to the families* (2) *and* (3) *respectively. The economic content (the operator $\mathcal{R}$ and the endogenous choice of $\{X_\ell\}$) is common to both. The methods differ in exactly one primitive: whether $\theta \mapsto g_\theta$ is affine or nonlinear.*

This is not a theorem; it is a definition written out. Its force is that "machine learning *vs.* projection methods" compares a toolkit to a function class. The honest comparison is *linear approximation space vs. adaptive nonlinear parameterization, inside one paradigm.* Here, "linear" refers to the parameters, not to any particular basis: a projection approximation is linear in its unknown coefficients, and the basis functions are the modeler's to choose in advance, whether Chebyshev polynomials, splines, or finite elements; a network instead folds trainable weights into the basis functions, so its approximation is nonlinear in its parameters. This is the classical distinction

between linear and nonlinear approximation (DeVore 1998). Everything below follows from taking that seriously.

To be clear: the endogeneity of the evaluation points—namely, that the states the model visits depend on the solution itself—is a real feature of economic problems, and the survey is right to stress it. But it is a property of the residual and the sampling, shared identically by (2) and (3). It distinguishes economic problems from generic function approximation, but it does not distinguish neural networks from projection.

A deeper similarity underlies the comparison, and it is worth stating because the word "finite" can mislead. At a given size each method realizes a finite-dimensional set: an M-term projection spans an M-dimensional linear subspace, and a width-M network traces out a nonlinear manifold of dimension at most the number of weights. But what is finite is only the truncation. The projection method selects a finite section of a complete basis $\{\phi_m\}_{m\geq 0}$ of an infinite-dimensional function space, such as polynomials, splines, or a Fourier system; the network selects finitely many units from the continuum of candidate features $\{\varphi(w^\top X) : w \in \mathbb{R}^{N+1}\}$. Both approaches replace an infinite-dimensional functional equation with a finite-dimensional approximation. The difference is the geometry of the approximating family: a linear subspace versus a nonlinear parameterized family. The practical issue is therefore not whether neural networks belong outside projection methods, but when their adaptive geometry buys enough approximation efficiency to justify the loss of the well-conditioned, low-dimensional structure, transparent diagnostics, and standard approximation guarantees a linear basis provides.[7]

A linear basis also makes qualitative shape restrictions easy to enforce: monotonicity and concavity, properties common to value and policy functions in economics, become linear inequality constraints on the coefficients and can be imposed exactly (Judd 1998). A generic network carries no such guarantee; shape can be encouraged through penalty terms or built in through specialized architectures, but it cannot be enforced as cheaply or as surely. As the discussion of the curse of dimensionality below makes precise, the two reach a given accuracy at very different costs as the dimension grows. The linear-versus-adaptive distinction is therefore about the shape and efficiency of a finite-dimensional approximation drawn from a shared infinite-dimensional space, not about what lies within reach in the limit. It is one more respect in which deep learning is a variation within the projection family rather than an alternative to it.

[7] As the number of terms grows, prescribed bases approximate any continuous function on a compact domain arbitrarily well (the Weierstrass and Stone theorems), and so do neural networks (the universal-approximation theorems of Cybenko (1989) and Hornik et al. (1989)).

Three Misunderstandings

The misclassification from the previous section is not a quibble about words; it propagates into three confusions common to the new-paradigm view, concerning what neural networks do, why they work, and when they should be preferred. The survey states them clearly enough to serve as our examples. We take them in turn.

What is called "learning" is residual optimization

The survey describes the determination of θ as *learning* the function, and the procedure as *machine learning* because it is automated. We do not quarrel with the word in its plain sense: One certainly learns the solution by running the solver, just as one learns a sum of numbers by running the SUM command in an Excel spreadsheet. But that sense cannot be what distinguishes a neural-network solver from a linear-basis one, because in it the Chebyshev solver learns the policy function too, as does the spreadsheet that totals a column. The technical term "learning" names something narrower.[8]

In the standard statistical-learning interpretation, one fits a rule using data generated by an unknown or only partially observed process and asks how well it performs away from the sample used to fit it. That is why generalization, the bias–variance trade-off, held-out validation, and regularization are central concepts. None of this statistical inference apparatus is present in (1). The evaluation points are placed by the modeler, the equations defining the residual $\mathscr{R}$ are fully specified by the model, and there is no empirical label noise. The fundamental object minimized is a model-implied equilibrium residual, not statistical predictive risk under an unknown data-generating process.

In the statistical-learning setting, the target criterion is tied to an unknown population or data-generating process; in model solving, the residual criterion is fixed by the model and approximated numerically. What is called "learning" in these implementations is the use of gradient-based optimization, often stochastic gradient descent or Adam, to minimize (1). The adaptivity the survey treats as distinctive—namely, that the functions ψ_m^θ are themselves adjusted—is ordinary nonlinear approximation: free-knot splines and radial bases with trainable centers "learn" their basis in exactly this sense and predate the vocabulary by decades. The analysis of families that adapt their shape to the target is classical nonlinear approx-

[8] One could, in the same loose sense, call the iterations of Newton's method "learning," but the machine-learning literature usually means more by the word: induction from data, or agents adapting their behavior through repeated interaction. The deep-learning solvers at issue here involve neither; they minimize a deterministic residual. Where "learning" does carry the richer sense, as in the multi-agent reinforcement learning that Zheng et al. (2022) use to *design* tax policy, the enterprise is simulating an adaptive process rather than solving a given model. This lies outside our scope.

imation theory; see, for example, the landmark survey by DeVore (1998). A neural network introduces a specific algebraic composition to this long-standing tradition; it does not alter the fundamental optimization task defined by Program (1).

The terminology matters because it changes the story economists tell themselves about the computation. Calling the procedure "learning" suggests that the algorithm extracts structure from data generated by an unknown process, which is not what happens: the economist supplies the model, the equilibrium operator, the residual, the domain, and the sampling rule, and the computer adjusts parameters to reduce the chosen loss. Where the implementations are stochastic, the randomness comes from minibatching or simulation; that is, from the modeler's own integration choice, not from an unknown empirical data-generating process. The procedure is better described as residual optimization over a nonlinear parameterized family than as learning in the usual statistical sense.

The survey's phrase "learned basis" is best read as informal shorthand. A network supplies a trained dictionary of features that enters the final layer linearly; these features need not be a basis in the strict sense. The relevant point is not that neural networks literally provide a basis, but that they enter the same residual-based numerical program as other approximating families.

The relabeling is not harmless, because it imports a statistical apparatus that the setting does not support. Generalization, the bias–variance trade-off, held-out test sets, and implicit regularization that improves out-of-sample prediction are statements about prediction under sampling noise. Here there is no sampling noise; a held-out set of points is a *coverage check* on the state space (is the residual small everywhere, or only where we trained?), which is numerical analysis, not inference. Where the expectation is formed by Monte Carlo, independent draws likewise guard against integration error, but this is error control, not inference about an unknown population. The survey's own worked example makes the point for us: when it comes to certifying the solution, it abandons the held-out validation set and reports Euler-equation residuals against a fine-grid benchmark, that is, it reaches for the projection-method diagnostic.

The implicit-regularization argument deserves a precise statement rather than a statistical one. The claim, which the survey draws from Wilson (2025), is that among the many parameter vectors that drive the residual to zero, gradient descent is biased toward a vector of small (min-)norm. That min-norm characterization is established for linear models and special training regimes; it is not established for nonconvex residual minimization over (3), and so for the present setting it is better read as a heuristic than as a guarantee.

The source of the high-dimensional advantage

The survey's central theoretical support is Barron's (1993) theorem: for a class of target functions, a single-hidden-layer network achieves mean-squared approximation error $O(1/M)$ *independent of the dimension*, N, whereas a tensor basis requires $M \sim D^N$ terms to maintain a resolution of D basis functions per dimension. We grant the premise at once: for target functions with ridge or compositional structure, an adaptive nonlinear family can reach an accuracy that no full tensor basis can attain at feasible cost, and this is a genuine advantage in high dimensions. Our concern is with what follows from it, not whether it holds. We make three observations.

First, whatever advantage this describes is a fact about nonlinear adaptive approximation on a favorable function class, so it is an approximation-theoretic property of the family (3) and not about "machine learning." Attributing the advantage to learning-from-data misnames its source, and obscures that the relevant competitors are *other* adaptive schemes, not the full tensor grid.

Second, Barron's theorem is nonconstructive: it asserts that a good network exists, not that minimizing (1) by gradient descent *finds* it.[9] The cost of finding a near-Barron network is itself unbounded by the theorem and may be large. A linear basis, by contrast, reduces (1) to a low-dimensional, well-conditioned problem whose per-iteration linear algebra is polynomial in M (at worst $O(M^3)$, often less). The relevant yardstick is wall-clock time to a target accuracy, not the number of terms in an existence proof.

Third, the approximation theorem holds strictly on a specified compact domain, and the numerical loss constrains the network only at the chosen evaluation points. Neither provides functional guarantees outside these regions. For structural economics, this is more than a technical extrapolation problem: many of the counterfactuals of interest concern precisely regions of the state space that are rarely or never visited under the baseline policy. The questions that motivate global solutions, such as rare disasters, tail-risk policy shifts, or unprecedented counterfactual policies, live precisely in the regions where the network was never evaluated. Because the loss constrains the network primarily inside the sampled region, a flexible approximator may fit equilibrium conditions well there while behaving erratically outside it.

None of this denies that adaptive nonlinear families can break the tensor curse for suitable targets. It demonstrates instead that the asymptotic existence

[9] The survey itself supplies two of these qualifications. It notes that the Barron rate "does not extend to all target functions; even within the Barron class, the implicit constants in $O(1/M)$ can be large in economically relevant problems" (Fernández-Villaverde 2026), and it concedes that the second aspect of the curse, the scarcity of evaluation points in high dimensions, can be overcome by "no approximation architecture," neural networks included.

bounds offered by Barron's theorem do not translate into practical computational guarantees.

A final clarification belongs here, because the curse of dimensionality is often treated as a single obstacle when it is several. A full tensor grid does pay an exponentially growing cost, and the survey is right that this becomes infeasible past a handful of state variables. But the rate comparison it draws, the $O(M^{-2s/N})$ accuracy of a generic series expansion (with s the smoothness order of the target) against the dimension-free Barron rate, takes a tensor basis as its foil. Modern projection practice does not stop there. Sparse grids and Smolyak approximations (Judd et al. 2014; Brumm and Scheidegger 2017) and adaptive, ε-distinguishable domains (Maliar and Maliar 2015) are built precisely to blunt the tensor cost in moderately high dimensions, and they, not the full tensor grid, are the proper benchmark; the concluding section returns to them. Separating the aspects of the curse also clarifies what is at stake: an adaptive family can ease the *approximation* cost on targets with ridge or compositional structure, but no architecture, neural or classical, overcomes the *scarcity of information* once evaluation points cover a vanishing fraction of the state space, and forming the conditional expectation carries an *integration* curse of its own. Deep learning attacks the first of these and shares the other two with every method, a point the deep-learning literature itself records. Neural-network methods "alleviate" rather than remove the curse (Azinovic et al. 2022), and its proponents separate the grid cost that adaptive sampling avoids from the integration cost that it does not (Maliar et al. 2021). That the high-dimensional regime is, in any case, not uncharted territory for the classical methods is clear from the second *JEDC* computational suite (Den Haan et al. 2011), whose multi-country real business cycle model was built to stress-test solution methods as the state dimension grows, with projection, perturbation, and stochastic-simulation methods benchmarked on it head-to-head. The relevant competitor in high dimensions is this modern projection practice, not the full tensor grid.

The criterion is the weak link, and it is a choice

The fragility reported in the survey (sensitivity to initialization, architecture, and learning rate; convergence to poor approximations) is real. But it is attributable to two choices *within* the family (1), not to the network itself, and projection practice already knows the remedies.

The first choice is the objective norm. The survey minimizes a mean-squared (L^2) residual. An L^2 criterion averages errors and thus tolerates large violations at isolated points; for solving a functional equation, the demanding and informative metric is the sup-norm of unit-free residuals, which is why the projection literature reports maximum Euler-equation errors (Judd 1992; Aruoba et al. 2006). A small mean-squared training loss is consistent with a solution that is locally wrong

where it matters. From the viewpoint of projection methods, this is a consequential restriction: Galerkin and collocation impose different, more structured residual restrictions than a sampled mean-squared loss, while the latter is largely chosen because it is convenient for stochastic-gradient optimization. The prevalence of L^2-type losses is therefore not a consequence of neural networks as approximating families, nor of projection methods as such; it reflects the convenience of average objectives for stochastic-gradient optimization (Goodfellow et al. 2016; Bottou et al. 2018).

The second choice is how the conditional expectation is formed. The survey's simulation-based residual is the parameterized-expectations construction of Den Haan and Marcet (1990): replace the expectation by a Monte Carlo draw along a simulated path and fit by least squares. This injects integration noise into the very object being approximated. The remedy is known and published: accurate deterministic quadrature for the expectation, combined with stable fitting on a well-chosen domain, as in the generalized stochastic-simulation algorithm of Judd et al. (2011), the ε-distinguishable-set domains of Maliar and Maliar (2015), or the Smolyak construction of Judd et al. (2014). These methods attack exactly the weaknesses the survey ascribes to "projection," using projection.

The point is structural, not partisan: L^2 with Monte Carlo is a simple averaging criterion. It can pass while missing isolated violations; therefore, it must be backed by external sup-norm diagnostics. Nonetheless, it is the member the survey adopts. The approximating family is orthogonal to this choice; one can pair either family with either criterion.

A Controlled Illustration

We use a canonical benchmark: the continuous-time deterministic neoclassical growth model (also the survey's illustrative example, in Section 9), with standard parameters ($\gamma = 2$, $\rho = 0.04$, $A = 0.5$, $\alpha = 0.36$, $\delta = 0.05$, so $k_{ss} = 2.95$) on $k \in [0.1, 10]$. The optimality condition is the consumption Euler equation; we treat the consumption policy $c(k)$ as the unknown and seek to drive its Euler residual to zero across the domain. We hold the economic problem, the state domain, and the accuracy diagnostic fixed, and pair each approximating family with its standard solver: Chebyshev collocation, which enforces a zero residual at the collocation nodes, and least-squares minimization of the residual for the neural network.[10]

[10] Implementation notes: The policy derivative entering the residual is formed by central differences in k; the network represents the policy directly. Both methods are anchored at the analytically known steady state (the value c_{ss} and the saddle-path slope at k_{ss}, available in closed form from the local linearization), so neither is otherwise handed the solution. The benchmark, c^*, is computed to near machine precision by integrating the model's stable manifold, an ODE solve independent of both methods. All computations

- **(A) Linear basis.** The consumption policy $c(k)$ is a Chebyshev expansion (2) in log-capital; the coefficients are solved by collocation under degree continuation.
- **(B) Adaptive parameterization.** The consumption policy $c(k)$ is a feedforward network (3) with three hidden layers of 64 tanh units (8,513 parameters), the survey's architecture, trained by Adam with a cosine-annealed learning rate, gradient clipping, and minibatches resampled from the state space each epoch.

Accuracy is the unit-free consumption error $|c(k)/c^*(k) - 1|$ on a fine test grid. Table 1 reports the maximum and mean error, the wall-clock time, and the parameter count; Figure 1 plots the policy and the error across the domain.

TABLE 1: Same model and same diagnostic; approximating family and standard residual implementation differ. Errors are unit-free consumption errors against the stable-manifold benchmark.

Method	Parameters	Wall-clock	Max. error	Mean error
Chebyshev (8 coeff.)	8	~4 ms	1.6×10^{-5}	6.0×10^{-6}
Chebyshev (12 coeff.)	12	~6 ms	1.4×10^{-9}	4.1×10^{-10}
Chebyshev (16 coeff.)	16	~9 ms	3.6×10^{-14}	9.7×10^{-15}
Neural network	8,513	~58 s	2.7×10^{-3}	9.7×10^{-4}

In this benchmark the comparison is stark. Both methods reproduce the policy to the eye (Figure 1a). But the linear basis reaches a maximum error of 1.4×10^{-9} with twelve coefficients in a few milliseconds (and machine precision with sixteen) while the network, with roughly seven hundred times as many parameters and about four orders of magnitude more wall-clock time, attains a maximum error of 2.7×10^{-3} (Figure 1b). Longer training and further tuning may improve the network, but the comparison shows that, in this example, the adaptive parameterization starts from a large disadvantage in accuracy, speed, and verifiability.[11]

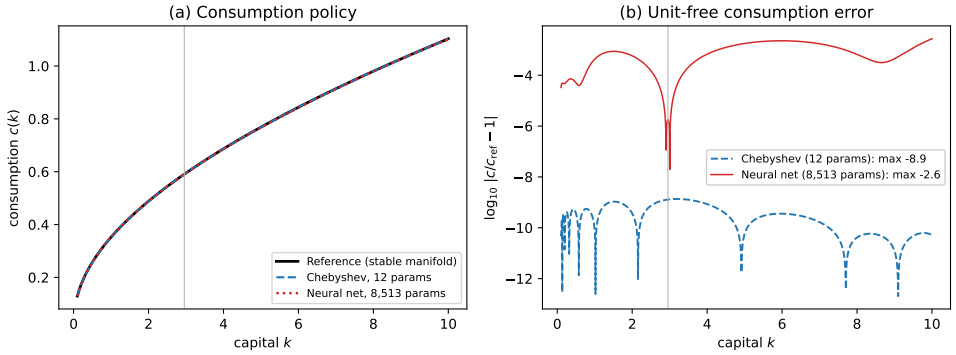
This is a one-dimensional problem, a setting where deep learning is not expected to excel, as the survey itself notes. We state the limits of the experiment plainly. The example does not test whether neural networks are useful in high-dimensional problems with favorable structure: they can be. It asks a narrower

were run on a MacBook Air (Apple M2, 8-core CPU, 16 GB unified memory) under macOS 26.5.1, in a Miniforge/`conda-forge` environment with Python 3.11.15 and native `arm64` builds of NumPy, SciPy, and Matplotlib; the network was trained on the CPU, with no GPU.

[11] Each family is used in its standard form. Matching the criteria would not change the ordering: the gap is driven by the approximating family, not the criterion, and solving the linear basis by least squares as well would only further support the results.

question: What does the adaptive parameterization buy, and what does it cost, in a benchmark where both approaches are feasible and accuracy can be measured against a near-exact solution? On that question, the answer is clear. In the benchmark, the neural network is slower, less accurate, and harder to verify. The experiment therefore should not be read as a refutation of neural networks in high dimensions. It should be read as a warning against using this low-dimensional teaching example to build intuition about their computational superiority. This is not a contrarian reading. The deep-learning literature itself motivates these methods by high dimensionality, where grid-based projection becomes infeasible (Azinovic et al. 2022), while granting that the canonical projection method is "remarkably fast for small problems" and grows costly only as the dimension rises (Maliar et al. 2021). Crucially, in moderate or high dimensions the proper benchmark is not the rigid tensor grid but the sparse and adaptive projection methods noted above, developed precisely to reduce the dimensional burden while preserving clearer diagnostics.

Figure 1: Solving the survey's continuous-time growth model. (a) The consumption policy: Reference (solid), Chebyshev with 12 coefficients (dashed), and the neural network (dotted) coincide to the eye. (b) Unit-free consumption error across the capital domain; the linear basis is six orders of magnitude more accurate than the adaptive parameterization.



One might object that our comparison minimizes the Euler residual, whereas the survey works with the value function and the Hamilton–Jacobi–Bellman (HJB) equation. The ordering is unchanged. On this very model, the survey's own neural-network solution attains Euler errors below 10^{-3} over most of the domain (its Figure 12). A standard linear-basis discretization of the same HJB equation, the finite-difference upwind scheme that is the workhorse for continuous-time models, attains a maximum consumption error of 1.0×10^{-4} on a ten-thousand-point grid

in about fifty milliseconds, and the spectral collocation above reaches 10^{-9}. Because this scheme recovers consumption from V' exactly as the network does, the comparison controls for the choice of unknown. The remaining gap reflects the difference between these standard numerical implementations, not the difference between policy and value function. These reported figures do not use a common metric, the survey reports a domain-averaged Euler residual and the finite-difference scheme a worst-case consumption error, so the value-function comparison is qualitative, drawn from each method's own reported diagnostic. The rigorous, single-metric comparison is that of Table 1. Even on the survey's most favorable reported figure for the network, the standard linear-basis discretization is at least as accurate at far lower cost, and spectral collocation is far more accurate.[12]

A Call to Action

Stripped of the paradigm framing, deep learning supplies an adaptive nonlinear parameterization for the method of weighted residuals. It can be valuable for structured high-dimensional targets, but it often gives up the well-conditioned, low-dimensional structure, transparent diagnostics, and reliable behavior away from the training region. The right question is therefore not which paradigm wins, but which member of the weighted-residual family to use for a given model. That family leaves three choices open: the approximating family, such as Chebyshev and other orthogonal polynomials, splines, finite elements, sparse and adaptive grids, radial bases, or neural networks; the residual criterion, such as collocation, Galerkin, or least squares; and the placement of evaluation points, such as tensor grids, sparse grids, quadrature nodes, or simulated paths. Different models call for different choices. A smooth, low-dimensional growth model is solved to near machine precision in milliseconds by a spectral basis; a model with kinks or occasionally binding constraints is better served by splines or finite elements; a heterogeneous-agent model with a high-dimensional state may need an adaptive sparse grid or a neural parameterization. None of these methods dominates the others everywhere; each is the right tool for some region of the modeling space.

The survey is right that ecosystems matter. It credits the "vibrant ecosystem" of graphics hardware and automatically differentiated libraries behind deep learning, and it invokes David (1985) for the lesson that a method often prevails less through intrinsic superiority than through the infrastructure that grows up around it. We agree, and draw the conclusion the survey does not. The most durable contribution of the deep-learning movement to economics is not a new approximating

[12] We also reproduced the survey's two-network formulation directly. Because we did not tune it to the survey's full training budget or use its exact automatic-differentiation derivatives, we cite the survey's own reported accuracy, which is the more favorable figure for the network.

family but a way of working: shared, well-engineered, hardware-aware, automatically differentiated libraries that let a researcher write down a residual and obtain a solution without rebuilding the machinery each time.

Economics is not without projection-based tools. The difficulty is not absence but *fragmentation*.[13] There are many separate projects, in different languages, with different interfaces, uneven maintenance, and small overlapping communities, not one coordinated, shared ecosystem in which a Chebyshev basis, a sparse grid, and a neural parameterization are interchangeable components behind a common interface.

The asymmetry with machine learning is real, but it is not, as is sometimes suggested, a matter of economists' hostility to software. Instead, the asymmetry is structural. Open-source scientific software earns little academic credit, effort is dispersed across MATLAB, Julia, and Python, and the deep-learning stack was built on industrial investment (Google's JAX, Meta's PyTorch) on a scale no field can match from research budgets alone. That the gap is nonetheless fillable is shown by Dynare itself, which became dominant and universally taught and has been maintained because it solved a real problem with a shared tool. The same can be done for the wider weighted-residual family.

Coordination is not unprecedented in this corner of economics. Following the comparison of solution methods for the stochastic growth model organized by Taylor and Uhlig (1990), the computational suites of Den Haan et al. (2010; 2011) assembled two common test models: an incomplete-markets economy with aggregate uncertainty and a multi-country real business cycle model, and had many groups solve each of them so that the accuracy and speed of competing methods could be compared on identical problems. The profession has thus already shown that it can agree on benchmark problems and evaluate methods against them in common. Such shared problems are also what a fair comparison between an adaptive parameterization and modern projection practice in high dimensions still lacks. Under controlled, common conditions that comparison has not in fact been run, and until it has, claims of across-the-board superiority, in either direction, remain asserted rather than established. What the profession has not yet done is carry its coordination from shared test models through to shared software.

It can, moreover, be built on the very infrastructure the deep-learning movement created. Automatic differentiation and GPU parallelism are general-purpose:

[13] Chebyshev and spline collocation have long been available in CompEcon (Miranda and Fackler 2004); QuantEcon (Sargent and Stachurski 2024) and the EconForge stack, including dolo (Winant 2019), provide interpolation and global-solution routines; SolveDSGE.jl (Dennis 2023) implements Chebyshev, Smolyak, and hyperbolic-cross projection; GDSGE (Cao et al. 2023) solves global models by policy-function iteration; adaptive sparse grids (Brumm and Scheidegger 2017) carry projection into higher dimensions; and Dynare (Adjemian et al. 2024), the profession's most widely used solver, has recently been augmented with sparse grids and high-dimensional model representation (Eftekhari et al. 2026).

The autodiff that trains a network also assembles the Jacobian of a collocation system, and the hardware that parallelizes minibatch gradients parallelizes the evaluation of a sparse-grid operator. There is no reason this tooling should be confined to the neural-network member of the family.

Deep learning is a useful tool with a known price, not a new paradigm. The contrast that organizes the survey, "machine learning versus projection methods," is a contrast between a function class and the family it belongs to. Stating this correctly does not diminish neural networks. It lets economists choose the approximating family on its merits, which is all a numerical method ever asks of us. We therefore read the recent enthusiasm for deep learning not as the arrival of a new paradigm but as an invitation to build a better toolkit: a numerical workbench in which approximating spaces and residual criteria are interchangeable components, a Chebyshev approximation space exchanged for a neural parameterization, or a least-squares criterion for collocation, without rewriting the model. Once deep learning is correctly placed inside the projection family, the task is no longer to choose between families but to make all of them easy to use, and to coordinate the profession's scattered efforts into one ecosystem rather than many.

Declaration on the Use of Artificial Intelligence

In preparing this manuscript, the authors used the large language model *Claude Opus 4.8* (Anthropic) to assist with preliminary literature discovery and to refine the clarity and flow of the text. The search parameters, core arguments, and final literature synthesis were entirely directed by the authors. All AI-assisted text was rigorously reviewed, verified against primary economic sources, and edited for scholarly accuracy. The authors assume full responsibility for the integrity and final content of this work.

References

Adjemian, Stéphane, Michel Juillard, Frédéric Karamé, Willi Mutschler, et al. 2024. Dynare: Reference Manual, Version 6. Dynare Working Paper 80. CEPREMAP. Link

Aruoba, S. Borağan, Jesús Fernández-Villaverde, and Juan F. Rubio-Ramírez. 2006. Comparing Solution Methods for Dynamic Equilibrium Economies. *Journal of Economic Dynamics and Control* 30(12): 2477–2508. Link

Azinovic, Marlon, Luca Gaegauf, and Simon Scheidegger. 2019. Deep Equilibrium Nets. Working paper, SSRN 3393482. Link

Azinovic, Marlon, Luca Gaegauf, and Simon Scheidegger. 2022. Deep Equilibrium Nets. *International Economic Review* 63(4): 1471–1525. Link

Barron, Andrew R. 1993. Universal Approximation Bounds for Superpositions of a Sigmoidal Function. *IEEE Transactions on Information Theory* 39(3): 930–945. Link

Bottou, Léon, Frank E. Curtis, and Jorge Nocedal. 2018. Optimization Methods for Large-Scale Machine Learning. *SIAM Review* 60(2): 223–311. Link

Brumm, Johannes and Simon Scheidegger. 2017. Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models. *Econometrica* 85(5): 1575–1612. Link

Cao, Dan, Wenlan Luo, and Guangyu Nie. 2023. Global DSGE Models. *Review of Economic Dynamics* 51: 199–225. Link

Cybenko, George. 1989. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals and Systems* 2(4): 303–314. Link

David, Paul A. 1985. Clio and the Economics of QWERTY. *American Economic Review* 75(2): 332–337. Link

Den Haan, Wouter J., Kenneth L. Judd, and Michel Juillard. 2010. Computational Suite of Models with Heterogeneous Agents: Incomplete Markets and Aggregate Uncertainty. *Journal of Economic Dynamics and Control* 34(1): 1–3. Link

Den Haan, Wouter J., Kenneth L. Judd, and Michel Juillard. 2011. Computational Suite of Models with Heterogeneous Agents II: Multi-Country Real Business Cycle Models. *Journal of Economic Dynamics and Control* 35(2): 175–177. Link

Den Haan, Wouter J. and Albert Marcet. 1990. Solving the Stochastic Growth Model by Parameterizing Expectations. *Journal of Business & Economic Statistics* 8(1): 31–34. Link

Dennis, Richard. 2023. SolveDSGE.jl: A Julia Package to Solve, Simulate, and Analyze Nonlinear DSGE Models. Software. Link

DeVore, Ronald A. 1998. Nonlinear Approximation. *Acta Numerica* 7: 51–150. Link

Eftekhari, Aryan, Michel Juillard, Normann Rion, and Simon Scheidegger. 2026. Scalable Global Solution Techniques for High-Dimensional Models in Dynare. *Journal of Economic Dynamics and Control* 182: 105225. Link

Fernández-Villaverde, Jesús. 2026. Deep Learning for Solving Economic Models. *Journal of Economic Literature* 64(3): 829–875. Link

Fernández-Villaverde, Jesús, Galo Nuño, George Sorg-Langhans, and Maximilian Vogler. 2020. Solving High-Dimensional Dynamic Programming Problems Using Deep Learning. Unpublished manuscript. Link

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. Cambridge, MA: MIT Press. Link

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks* 2(5): 359–366. Link

Judd, Kenneth L. 1992. Projection Methods for Solving Aggregate Growth Models. *Journal of Economic Theory* 58(2): 410–452. Link

Judd, Kenneth L. 1998. *Numerical Methods in Economics*. Cambridge, MA: MIT Press. Link

Judd, Kenneth L., Lilia Maliar, and Serguei Maliar. 2011. Numerically Stable and Accurate Stochastic Simulation Approaches for Solving Dynamic Economic Models. *Quantitative Economics* 2(2): 173–210. Link

Judd, Kenneth L., Lilia Maliar, Serguei Maliar, and Rafael Valero. 2014. Smolyak Method for Solving Dynamic Economic Models: Lagrange Interpolation, Anisotropic Grid and Adaptive Domain. *Journal of Economic Dynamics and Control* 44: 92–123. Link

Kingma, Diederik P. and Jimmy J. Ba. 2015. "Adam: A Method for Stochastic Optimization". In *3rd International Conference on Learning Representations (ICLR)*. Link

Maliar, Lilia and Serguei Maliar. 2015. Merging Simulation and Projection Approaches to Solve High-Dimensional Problems with an Application to a New Keynesian Model. *Quantitative Economics* 6(1): 1–47. Link

Maliar, Lilia, Serguei Maliar, and Pablo Winant. 2019. Will Artificial Intelligence Replace Computational Economists Any Time Soon? CEPR Discussion Paper 14024. Centre for Economic Policy Research. Link

Maliar, Lilia, Serguei Maliar, and Pablo Winant. 2021. Deep Learning for Solving Dynamic Economic Models. *Journal of Monetary Economics* 122: 76–101. Link

Miranda, Mario J. and Paul L. Fackler. 2004. *Applied Computational Economics and Finance*. Cambridge, MA: MIT Press. Link

Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis. 2019. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *Journal of Computational Physics* 378: 686–707. Link

Sargent, Thomas J. and John Stachurski. 2024. QuantEcon: Open Source Tools for Computational Economics. Software and lectures. Link

Sirignano, Justin and Konstantinos Spiliopoulos. 2018. DGM: A Deep Learning Algorithm for Solving Partial Differential Equations. *Journal of Computational Physics* 375: 1339–1364. Link

Taylor, John B. and Harald Uhlig. 1990. Solving Nonlinear Stochastic Growth Models: A Comparison of Solution Methods. *Journal of Business & Economic Statistics* 8(1): 1–17. Link

Wilson, Andrew Gordon. 2025. "Deep Learning Is Not So Mysterious or Different". In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*. Position Paper Track. Link

Winant, Pablo. 2019. Dolo: A Toolbox for Solving Economic Models. EconForge. Software. Link

Zheng, Stephan, Alexander Trott, Sunil Srinivasa, David C. Parkes, and Richard Socher. 2022. The AI Economist: Taxation Policy Design via Two-Level Deep Multiagent Reinforcement Learning. *Science Advances* 8(18): eabk2607. Link

About the Authors

Kenneth L. Judd is a Senior Fellow at the Hoover Institution at Stanford University. He is a leading scholar in computational economics, mathematical economics, public finance, and industrial organization. His research has developed computational methods for solving economic models and applied them to questions in taxation, antitrust, macroeconomics, and climate policy. He has served as coeditor of the *RAND Journal of Economics* and the *Journal of Economic Dynamics and Control*, and as associate editor of the *Journal of Public Economics*. His book *Numerical Methods in Economics*, published by MIT Press, has become a standard reference in computational economics. He holds undergraduate degrees in mathematics and computer sciences, an MA in mathematics, an MA in economics, and a PhD in economics, all from the University of Wisconsin. He is a Fellow of the Econometric Society. His email is judd@stanford.edu.

Karl Schmedders is Professor of Finance at IMD. Before joining IMD, Karl was Professor of Quantitative Business Administration at the University of Zurich and Associate Professor (with tenure) of Managerial Economics and Decision Sciences at the Kellogg School of Management, Northwestern University, where he remains a Visiting Professor of Executive MBA Education. He holds an MS and a PhD in Operations Research from Stanford University. He is a Fellow of the Game Theory Society and an SAET Economic Theory Fellow. His email is karl.schmedders@imd.org.

Go to archive of Comments section
Go to September 2026 issue