SPHERICALDEQ: AN ITERATION-FREE DEEP-EQUILIBRIUM LAYER

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep equilibrium models (DEQs) define neural layers implicitly as fixed points, enabling infinite-depth computation with constant activation memory. Their practical use, however, is hindered by the need to solve a nonlinear fixed-point problem at every forward pass and an implicit system during backpropagation. We introduce SphericalDEQ, an equilibrium layer that eliminates these iterative computations. Its key ingredient is a homogeneous spherical normalization that reduces the nonlinear fixed-point problem to a single linear solve followed by normalization. We establish conditions for the existence and uniqueness of this equilibrium and derive closed-form parameter derivatives, enabling exact backpropagation without iterative implicit differentiation, series approximations, or phantom gradients. This yields an equilibrium layer whose inference cost is comparable to that of an explicit layer once the inverse operator is cached. Experiments across image classification and neural PDE simulation show that SphericalDEQ substantially reduces the computational and memory overhead of conventional DEQs while matching or improving their predictive performance and, scales to ImageNet-1K classification and large-scale autoregressive physical simulation.

1 INTRODUCTION

Deep Equilibrium Models (DEQs) (Bai et al., 2019) replace the explicit computation of a conventional neural layer, $x_{\text{out}} = F_\theta(x_{\text{in}})$, with an implicitly defined equilibrium. Specifically, a DEQ layer outputs a solution z^* of $z^* = F_\theta(z^*, x_{\text{in}})$. This formulation admits an infinite-depth interpretation: the forward iteration $z_{k+1} = F_\theta(z_k, x_{\text{in}})$ corresponds to repeatedly applying the same weight-tied layer, with its limiting state defining the equilibrium.

This iterative viewpoint is appealing for problems in which repeated computation can progressively refine a representation or solution. In language modeling, where DEQs were originally introduced (Bai et al., 2019), repeated application of a shared transformation provides a form of adaptive computation reminiscent of test-time scaling (Snell et al., 2024). Similarly, many neural simulators for partial differential equations (PDEs) rely on iterative procedures that progressively improve an approximate solution (Marwah et al., 2023).

The implicit formulation, however, introduces a substantial computational burden. Evaluating a DEQ layer requires solving a high-dimensional nonlinear fixed-point problem, typically using methods such as Anderson acceleration (Anderson, 1965) or quasi-Newton schemes such as Broyden's method (Broyden, 1965). Moreover, despite their infinite-depth interpretation, practical architectures may benefit from composing multiple equilibrium blocks to build richer hierarchical representations (Bai et al., 2020; Winston & Kolter, 2020; Gabor et al., 2024). In this setting, repeatedly invoking a nonlinear root-finding procedure can become prohibitively expensive, limiting the scalability of DEQ architectures.

A second challenge is well-posedness. For a generic DEQ layer, the existence or uniqueness of an equilibrium and the convergence of the numerical solver used to compute it need not be guaranteed. As we illustrate in section 2, loss of convergence can occur during training and lead to severe instability. Existing approaches address this problem either by regularizing the equilibrium dynamics (Bai et al., 2021) or by imposing structural conditions that guarantee existence, uniqueness, and convergence (Gabor et al., 2024; Sittoni & Tudisco, 2024; Winston & Kolter, 2020). Regularization

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

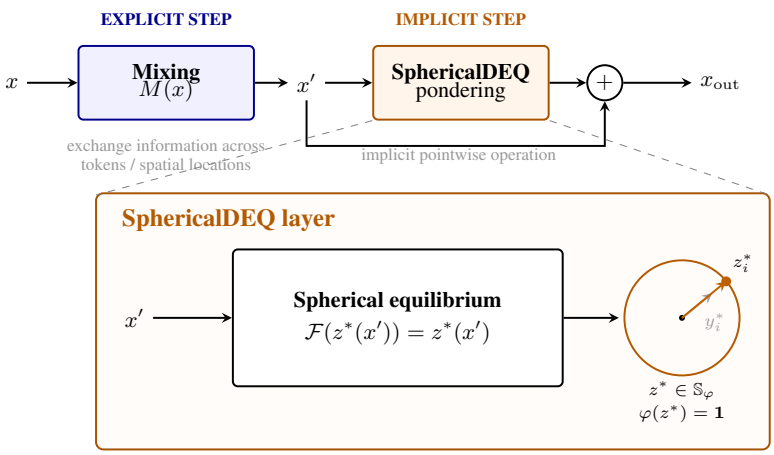


Figure 1: Schematic of an SphericalDEQ layer within an implicit–explicit architecture.

introduces additional hyperparameters and does not itself guarantee convergence, while theoretically well-posed constructions can require restrictive assumptions on the equilibrium map. Such restrictions may limit architectural flexibility and, potentially, scalability to more challenging tasks.

Contributions. In this work, we introduce SphericalDEQ, a family of nonlinear equilibrium layers that overcomes the computational and well-posedness limitations of standard DEQs. The key ingredient is a nonlinearity based on a homogeneous spherical normalization that gives the equilibrium problem sufficient structure to admit an explicit characterization, while retaining a nonlinear, implicitly defined layer.

Our main contributions are:

- We introduce SphericalDEQ and establish existence and uniqueness of an equilibrium for Lebesgue-almost every set of parameters.
- We show that the equilibrium can be computed through a linear solve followed by normalization, eliminating nonlinear fixed-point iterations, and derive closed-form parameter derivatives for exact backpropagation. We derive closed-form parameter derivatives, enabling exact backpropagation without iterative implicit differentiation, Neumann approximations, or phantom gradients. We exploit direct linear-algebra methods and operator caching to obtain an inference cost comparable to that of an explicit layer.
- We evaluate SphericalDEQ on image classification and neural PDE simulation, including ImageNet-1K and large-scale autoregressive physical simulation. Across these settings, SphericalDEQ matches or improves the predictive performance of conventional DEQs while substantially reducing computational and memory costs, with order-of-magnitude inference speedups over competing DEQ baselines.

2 Motivation: Stability and Computational Cost in DEQs

To illustrate how loss of fixed-point convergence can affect DEQ training, we consider a simple image classification experiment on MNIST (LeCun et al., 1998). The model consists of a single DEQ-MLP block

$$F(z) = W_2\,\mathrm{GELU}(W_1 z + Ux),$$

where x is the input image, U is an input projection layer, and W_1, W_2 are learnable linear maps. The equilibrium state z^* is then passed to a linear classifier.

To monitor stability during training, we track the following upper bound on the Lipschitz constant of the DEQ map: $\mathrm{Lip}(F) \approx \sigma(W_2)\,\max|\mathrm{GELU}'|\,\sigma(W_1)$, where $\sigma(\cdot)$ denotes the largest singular value (i.e., the spectral norm).

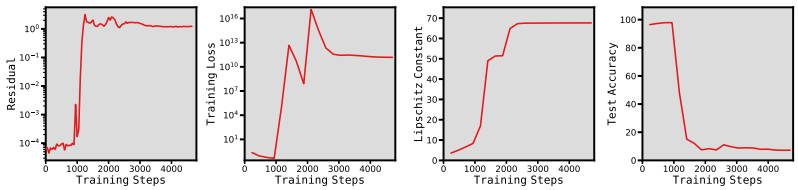


Figure 2: **Training instability in a simple DEQ model on MNIST.** We report (from left to right) the residual at the last step of the fixed-point method ($\|z_{k-1} - z_k\|_2$), the Lipschitz constant of the implicit layer estimated on the training set, the training loss, and the test accuracy throughout training. As the Lipschitz constant grows, the fixed-point solver ceases to converge, coinciding with a sharp deterioration in predictive performance.

Figure 2 illustrates the resulting training instability. Early in training, the fixed-point iteration converges and the residual remains small. As training progresses, however, the Lipschitz estimate increases and the solver eventually fails to converge. This loss of convergence coincides with a sharp increase in training loss and a collapse in test accuracy to approximately 7%. More details on the experimental setting can be found in Appendix A.

This illustrates that failures may be highly significant and motivate the need to control and stabilize DEQ learning. Fixed-point computation also introduces a non-negligible computational overhead. In this small-scale experiment, a forward pass through the DEQ layer takes 3.14 ms during training, compared with 0.9 ms for the corresponding explicit layer under the same experimental conditions. This overhead arises from the repeated evaluations of F required by the fixed-point solver and therefore compounds when equilibrium layers are composed or larger problems are considered.

Together, these observations highlight the two properties we seek in an equilibrium layer: well-posed and stable equilibrium computation, without the cost of iterative nonlinear fixed-point solving. In the following section, we introduce SphericalDEQ, which is designed to provide both.

3 Related Work

Deep Equilibrium Models (DEQs) DEQs, introduced by (Bai et al., 2019), define the output of a layer implicitly as the solution of a fixed-point problem, which is computed with a fixed-point method such as the classic forward iteration $z_k = f(z_{k-1})$ or a more sophisticated scheme like Anderson acceleration (Anderson, 1965). Several works have studied DEQs theoretically: (Ling et al., 2023; Gao et al., 2022) analyzed the convergence of overparameterized DEQs; (Wu et al., 2024) studied their expressivity, showing potential advantages in learning certain high-frequency components; and (Sun & Shi, 2024) examined their latent representations from a neural collapse perspective. A particularly relevant line of work develops DEQ architectures for which well-posedness can be guaranteed by construction. Positive and subhomogeneous DEQs use tools from nonlinear Perron–Frobenius theory to establish existence and uniqueness of equilibria (Gabor et al., 2024; Sittoni & Tudisco, 2024), while Monotone DEQs obtain related guarantees through monotone operator theory (Winston & Kolter, 2020; Ryu & Boyd, 2016). These approaches require an iterative procedure to compute the equilibrium, whereas SphericalDEQ exploits homogeneous normalization to reduce the equilibrium computation to a linear solve followed by normalization.

Complementary work has addressed the numerical difficulties of DEQ training. Jacobian regularization can improve the stability of equilibrium computation (Bai et al., 2021), while alternative backward-pass approximations have been proposed to reduce the cost of implicit differentiation (Geng et al., 2021; Fung et al., 2022). Other approaches directly target the computational cost of equilibrium solving. (Lin et al., 2026) use distillation to reduce the number of solver iterations, while (Bai et al., 2022b) train neural networks to approximate the equilibrium and thereby bypass expensive iterative solvers.

DEQs in practice DEQs were originally introduced for language modeling (Bai et al., 2019) and were subsequently extended to computer vision tasks, including image classification, semantic segmentation (Bai et al., 2020), optical flow (Bai et al., 2022a), and object detection (Wang et al., 2023). Equilibrium architectures have since been applied to inverse problems (Gilton et al., 2021), graph-structured data (Gu et al., 2020; Liu et al., 2022a), PDE solving (Marwah et al., 2023), molecular dynamics (Burger et al., 2025), algorithmic reasoning (Georgiev et al., 2024), and quantum machine

learning (Schleich et al., 2024). Equilibrium formulations have also been explored in generative modeling, including diffusion sampling (Pokle et al., 2022) and the distillation of diffusion inference into a single equilibrium layer (Geng et al., 2023).

4 SPHERICALDEQ

We now introduce SphericalDEQ, a family of nonlinear equilibrium layers whose fixed points can be computed without iterative solvers. The key idea is to combine a linear transformation with a homogeneous row-wise normalization. We begin by introducing the necessary notation and definitions.

4.1 MATHEMATICAL SETTING

We use the following row-wise notation. For $\lambda \in \mathbb{R}^n$ and $x \in \mathbb{R}^{n\times d}$, we write $\lambda \odot x \in \mathbb{R}^{n\times d}$ to denote the *row-wise product* $(\lambda \odot x)_{ij} = \lambda_i x_{ij}$. Similarly, we let $x \oslash \lambda \in \mathbb{R}^{n\times d}$ denote the *row-wise quotient* $(x \oslash \lambda)_{ij} = x_{ij}/\lambda_i$.

Next, we introduce the class of row-wise homogeneous functions.

Definition 4.1 (*Row-wise homogeneous function*). *A function $\varphi : \mathbb{R}^{n\times d} \to \mathbb{R}^n$ is* homogeneous*, denoted $\varphi \in \mathcal{H}(\mathbb{R}^{n\times d}, \mathbb{R}^n)$, if*

(i) $\varphi(x) \geq 0$ (entrywise) for all $x \in \mathbb{R}^{n\times d}$;

(ii) $\varphi(\lambda \odot x) = |\lambda| \odot \varphi(x)$ for all $x \in \mathbb{R}^{n\times d}$ and $\lambda \in \mathbb{R}^n$, where $|\lambda| \in \mathbb{R}^n$ denotes the entrywise absolute value.

Definition 4.2 (φ*-sphere*). *Given $\varphi \in \mathcal{H}(\mathbb{R}^{n\times d}, \mathbb{R}^n)$, the* φ-sphere *is the set*

$$\mathbb{S}_\varphi = \{x \in \mathbb{R}^{n\times d} \mid \varphi(x) = \mathbf{1}\},$$

where $\mathbf{1}$ is the vector of all ones.

Example 4.3 (Row-wise p-norm). *Let $p > 1$, and for $x \in \mathbb{R}^{n\times d}$ with rows $x_i = [x_{i1}, \ldots, x_{id}]$, define*

$$\varphi_i(x_i) = \Big(\sum_{j=1}^{d} |x_{ij}|^p\Big)^{1/p}, \qquad i = 1, \ldots, n.$$

Then $\varphi(x) = [\varphi_1(x_1), \ldots, \varphi_n(x_n)]$ belongs to $\mathcal{H}(\mathbb{R}^{n\times d}, \mathbb{R}^n)$. Thus, applying an ℓ_p norm independently to each row defines a row-wise homogeneous function, whose associated φ-sphere consists of matrices with unit ℓ_p-norm rows.

4.2 THE SPHERICALDEQ UNIT

Using this row-wise normalization, we now define the nonlinear equilibrium map underlying SphericalDEQ and show that its fixed point can be characterized through a linear solve followed by normalization.

Let $W : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ be a linear operator, let $b \in \mathbb{R}^{n\times d}$, let $\varphi \in \mathcal{H}(\mathbb{R}^{n\times d}, \mathbb{R}^n)$, and let $\alpha \in \mathbb{R}^n$ be entrywise positive. A SphericalDEQ unit is defined by a fixed point of the mapping

$$\mathcal{F}_\alpha \colon \mathbb{S}_\varphi \to \mathbb{S}_\varphi, \qquad \mathcal{F}_\alpha(z) = (Wz + \alpha \odot b) \oslash \varphi(Wz + \alpha \odot b).$$

In the remainder of this subsection, we show that this particular structure allows its fixed points to be characterized through constrained linear systems. All proofs are deferred to Appendix B.

Lemma 4.4 (Constrained linear formulation). *$z = \mathcal{F}_\alpha(z)$ is a fixed point for $\mathcal{F}_\alpha \colon \mathbb{S}_\varphi \to \mathbb{S}_\varphi$ if and only if it is a solution to the constrained linear problem*

$$Wz + \alpha \odot b = z \qquad \textit{subject to} \qquad \varphi(z) = \mathbf{1}.$$

For linear operators W with a suitable Kronecker structure, we can further establish existence and uniqueness of the equilibrium and obtain an explicit characterization in terms of a linear solve and normalization.

Theorem 4.5 (Existence and uniqueness). *Let $W : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ be a linear operator of the form $W = \sum_{i=1}^{r} W_i^\top \otimes \Lambda_i$ where Λ_i is a diagonal $n \times n$ matrix, $W_i \in \mathbb{R}^{d\times d}$, and $\otimes$ denotes the Kronecker product. If 1 is not an eigenvalue of W, then there exists an entrywise positive $\alpha^* \in \mathbb{R}^n$ such that $\mathcal{F}_{\alpha^*}$ admits a unique fixed point $z^* \in \mathbb{S}_\varphi$.*

We note that the condition $1 \notin \mathrm{Eigenvalues}(W)$ is generic and not restrictive, as the set of matrices having 1 as an eigenvalue has Lebesgue measure zero.

Theorem 4.6 (Linear-solve formulation). *Under the assumptions of Theorem 4.5, let*

$$y = (I - W)^{-1}b, \qquad z^* = y \oslash \varphi(y).$$

Then z^* is the unique fixed point of $\mathcal{F}_{\alpha^*}$, where α^* is the entrywise positive scaling from Theorem 4.5.

Theorem 4.6 provides the key computational property of SphericalDEQ: its nonlinear equilibrium can be obtained by solving a linear system and normalizing the result row-wise. Thus, given W and b, no nonlinear fixed-point iteration is required. Moreover, the corresponding scaling α^* is determined implicitly by the network parameters; see Appendix B for details.

4.3 Efficient forward and backward implementation

While Theorems 4.5 and 4.6 apply to operators of arbitrary Kronecker rank r, in our implementations we typically use the simpler rank-one structure

$$W = A^\top \otimes I_n,$$

where $A \in \mathbb{R}^{d\times d}$ is a learnable parameter matrix, which may itself be structured, for example to represent a row-wise convolution. This choice is particularly convenient computationally, since W acts independently on each row $Wz = (A^\top \otimes I_n)z = zA$. Consequently, the linear system in Theorem 4.6 reduces to $y(A - I) = -b$, so that all n rows share the same $d \times d$ coefficient matrix. Computing the SphericalDEQ equilibrium therefore requires solving a single $d \times d$ system with n right-hand sides, rather than operating on the full $nd \times nd$ Kronecker matrix $I - W$.

In our setting, n is typically much larger than d: for sequences, $n = bt$ for batch size b and sequence length t, while for feature maps $n = bhw$. We considered both factor-and-solve methods and explicitly forming $(A - I)^{-1}$. These approaches have the same leading-order cost, $\mathcal{O}(nd^2)$, when applied to all n rows, with an additional $\mathcal{O}(d^3)$ preprocessing cost that is negligible in our regime. Empirically, explicit inversion was consistently faster: it reduces the application to a single highly optimized matrix multiplication, avoiding triangular solves and benefiting from the high throughput of modern accelerators. For numerical robustness, we compute the small $d \times d$ inverse in at least FP32 precision and cast it to the model dtype only for its application. At inference time, the inverse is cached, so equilibrium computation reduces to a single matrix multiplication followed by normalization, with a cost comparable to that of an explicit pointwise layer.

The same structure also simplifies the backward pass. Standard DEQs require solving an implicit linear system during backpropagation or approximating the implicit gradient. In SphericalDEQ, both the linear solve and the normalization can instead be differentiated in closed form, yielding exact parameter gradients.

Theorem 4.7 (Exact derivative computation). *Assume that φ is differentiable and define*

$$y = (I - W)^{-1}b, \qquad z^* = y \oslash \varphi(y).$$

Let $L(W, b) = \mathcal{L}(z^*)$ and $g = \nabla\mathcal{L}$. Then, for every W such that $I - W$ is invertible, it holds

$$\begin{aligned}
\nabla_y L &= g \oslash \varphi(y) - J_\varphi^\top \left(g \odot y \oslash \varphi(y)^2\right), \\
\nabla_b L &= (I - W)^{-\top} \nabla_y L, \\
\nabla_W L &= \nabla_b L \, (y)^\top.
\end{aligned}$$

When SphericalDEQ is used within a larger network, the forcing term may depend on the preceding representation, for example $b(x) = b + x$. In this case, Theorem 4.7 extends straightforwardly and we have $\nabla_x L = \nabla_b L$.

Thus, the backward pass requires only matrix operations involving the same linear operator used in the forward pass, without differentiating through solver iterations or resorting to Neumann approximations or phantom gradients. Algorithm 1 summarizes the resulting forward and backward computations using PyTorch-style pseudocode.

Algorithm 1 Minimal SphericalDEQ layer pseudocode with exact backpropagation.

```
class SphericalDEQ(Function):

    ...

    def forward(x,W,b,scale):
        Ainv = inverse(I-W)    # cached during inference
        xb =  x + b
        y = xb @ Ainv
        r = rms(y)
        z = y/r
        save_for_backward(y,z,r,Ainv,scale)
        return scale*z

    def backward(G):
        y,z,r,Ainv,scale = saved_for_backward()
        g = scale * G
        gy = (g - z*(z*g).mean(-1,keepdim=True))/r
        V  = gy @ Ainv
        return V, V.T@y, V.sum(0)          # returns ∇_x L, ∇_W L, ∇_b L
```

4.4 Injecting SphericalDEQ layer inside modern architectures

Many modern neural architectures alternate between two types of operations: a *mixing* operation that exchanges information across tokens or spatial locations, and a *pointwise* transformation that processes the resulting features independently at each location. For example, a standard Transformer block (Vaswani et al., 2017) takes the form

$$x' = \mathrm{MHA}(x) + x, \qquad x_{\mathrm{out}} = \mathrm{MLP}(x') + x'.$$

where multi-head self-attention performs the mixing operation and the MLP acts independently on each token. A similar decomposition appears in convolutional architectures such as ConvNeXt (Liu et al., 2022b), where spatial convolution provides the mixing operation, and in Fourier neural operators (Li et al., 2020; Tran et al.), where mixing is performed by a Fourier integral operator.

The row-wise structure of SphericalDEQ makes it a natural replacement for the pointwise transformation in such architectures. Rather than representing the entire network by a single equilibrium block, we interleave explicit mixing operations with SphericalDEQ layers. This preserves the architecture-specific mechanism for exchanging information across tokens or spatial locations while introducing an implicit nonlinear transformation of the resulting features. In this respect, the construction is also reminiscent of Mixture-of-Experts architectures (Shazeer et al., 2017; Fedus et al., 2022), where the pointwise transformation is replaced while the mixing mechanism is left unchanged.

This organization can be interpreted through the lens of implicit–explicit (IMEX) methods (Ascher et al., 1995), which split a computation into explicitly and implicitly treated components. In our setting, the mixing operation forms the explicit component, while the pointwise SphericalDEQ layer forms the implicit component. A generic block therefore takes the form

$$\begin{aligned} x' &= M(x), && \textit{(mixing phase or explicit step)} \\ x_{\mathrm{out}} &= \mathrm{SphericalDEQ}(x') + x', && \textit{(pondering phase or implicit step)} \end{aligned} \tag{1}$$

where M denotes the architecture-specific mixing operator. This construction is agnostic to the choice of M: it can be instantiated using, for example, self-attention, convolutional operators, or Fourier-parameterized integral operators. This formulation allows multiple SphericalDEQ blocks to be stacked within a deep architecture, rather than relying on a single weight-tied equilibrium layer to represent the full computation, as it is standard practice in architectures implementing DEQ layers (Bai et al., 2020; Gabor et al., 2024; Marwah et al., 2023). In our construction, each block computes its equilibrium independently, while the explicit mixing operations propagate information across tokens or spatial locations between successive equilibrium layers.

5 EXPERIMENTS

We evaluate SphericalDEQ in terms of computational scalability, predictive performance relative to existing well-posed implicit models, and applicability to large-scale image classification and physical simulation. We first compare training time and memory use with existing DEQ and Neural ODE architectures (Section 5.1), and then compare accuracy and inference cost with well-posed implicit baselines on standard image-classification benchmarks (Section 5.2). Finally, we evaluate SphericalDEQ at larger scale on ImageNet-1K and on autoregressive neural PDE simulation (Section 5.4).

Following the construction of Section 4.4, we instantiate each SphericalDEQ block by combining an explicit mixing operator with a pointwise implicit transformation. For image classification, the mixing operator is convolutional, while for physical simulation we use a Fourier Neural Operator (FNO) layer (Li et al., 2020). The implicit component uses RMS norm as activation function (Zhang & Sennrich, 2019), which is widely used in modern neural architectures (Yang et al., 2024; Groeneveld et al., 2024; DeepSeek-AI et al., 2024). The RMSNorm layer is defined as $T(x) = x \oslash \mathrm{RMS}(x) \odot \gamma$, where $\gamma \in \mathbb{R}^d$ is a learnable scaling vector applied channel-wise, and with $\mathrm{RMS}(x) = [\mathrm{RMS}(x_1), \ldots, \mathrm{RMS}(x_n)]$, $\mathrm{RMS}(x_i) = \sqrt{\frac{1}{d}\sum_{j=1}^{d} x_{ij}^2}$. Complete architectural and training details are provided in Appendix E.

5.1 COMPUTATIONAL SCALABILITY

We evaluate whether eliminating nonlinear fixed-point iterations translates into improved computational efficiency in practice. We measure training throughput and peak memory consumption for both image classification and neural PDE simulation.

Well-posed implicit models. Figure 3 compares SphericalDEQ with three well-posed DEQ architectures—MonDEQ (Winston & Kolter, 2020), pcDEQ (Gabor et al., 2024), and SubDEQ (Sittoni & Tudisco, 2024)—and two Neural ODE baselines, NODE (Chen et al., 2018) and ANODE (Dupont et al., 2019). Across MNIST, SVHN, and CIFAR-10, SphericalDEQ achieves the highest training speed. Its memory consumption also scales favorably with problem size: pcDEQ has a marginally smaller peak memory footprint on MNIST and SVHN, whereas on CIFAR-10 SphericalDEQ requires substantially less memory than all competing methods.

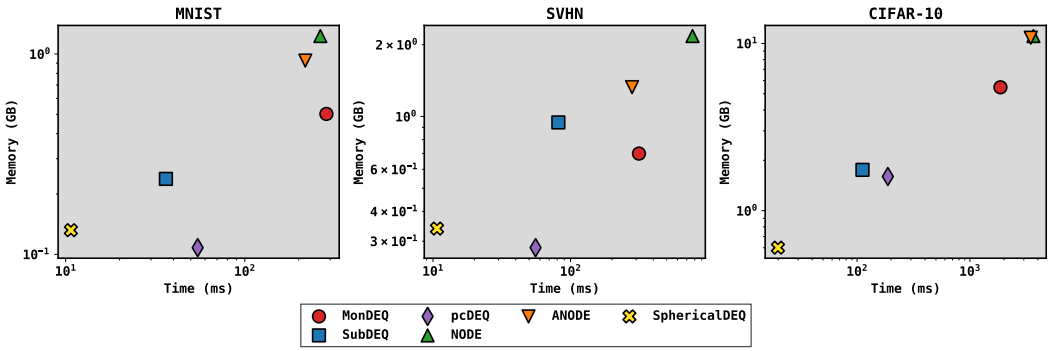


Figure 3: **Training scalability comparison with existing well-posed implicit models.**Comparison with: 4 different DEQ models (MonDEQ (Winston & Kolter, 2020), pcDEQ (Gabor et al., 2024), SubDEQ (Sittoni & Tudisco, 2024), and our model SphericalDEQ) and 2 Neural ODE models (NODE (Chen et al., 2018) and ANODE (Dupont et al., 2019)). We report the training speed on the x-axis and the training memory peak on the y-axis.

Neural PDE solvers. We next compare FNO-SphericalDEQ with FNO-DEQ (Marwah et al., 2023) and its weight-tied explicit counterpart FNO-WT on three datasets from The Well (Ohana et al., 2024). As shown in Figure 4, FNO-SphericalDEQ is both the fastest and the most memory-efficient model on all three datasets. In contrast to image classification, where the memory advantage becomes most pronounced on the largest benchmark, the improvement is substantial across all three physical-simulation tasks.

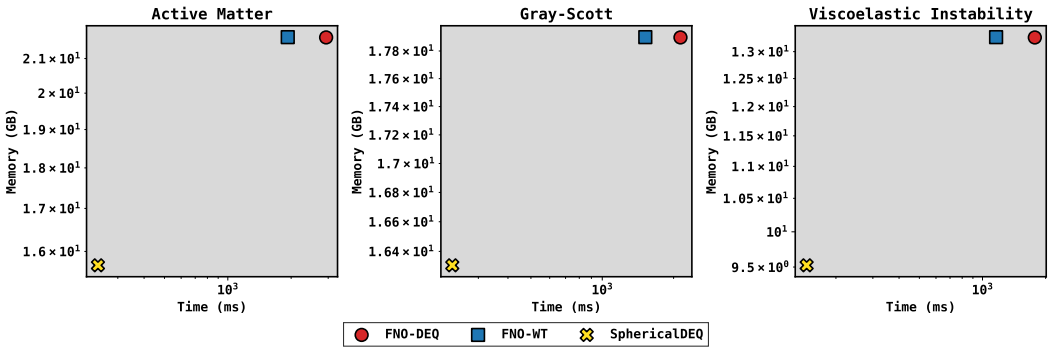


Figure 4: **Comparison of training scalability with other DEQ and weight-tied Neural PDE Solver.** We compare against FNO-DEQ(Marwah et al., 2023) and its weight-tied variant. We report the training speed on the x-axis and the training memory peak on the y-axis.

5.2 Comparison with well-posed implicit architectures

We next compare the predictive performance and inference cost of SphericalDEQ with five well-posed implicit architectures: MonDEQ (Winston & Kolter, 2020), pcDEQ (Gabor et al., 2024), SubDEQ (Sittoni & Tudisco, 2024), NODE (Chen et al., 2018), and ANODE (Dupont et al., 2019). We evaluate all methods on MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011), and CIFAR-10 (Krizhevsky et al., 2009), following a training protocol similar to (Winston & Kolter, 2020). Complete training details are provided in Appendix E. As shown in Section 5.2, SphericalDEQ achieves the highest classification accuracy and the lowest inference time on all three datasets. The difference is particularly pronounced on CIFAR-10, where SphericalDEQ reaches 93.18% accuracy while the strongest competing DEQ baseline reaches 89.2%. At the same time, its per-batch inference time is 3.7 ms, compared with 57.8–1017.3 ms for the competing implicit models.

	MonDEQ			SubDEQ			pcDEQ			NODE			ANODE			SphericalDEQ		
Dataset	*Params*	*Acc*	*Infer*	*Params*	*Acc*	*Infer*	*Params*	*Acc*	*Infer*	*Params*	*Acc*	*Infer*	*Params*	*Acc*	*Infer*	*Params*	*Acc*	*Infer*
MNIST	84K	99.0	131.1	30K	99.3	14.6	40K	99.3	23.1	84K	96.4	70.8	84K	98.2	53.6	83K	**99.61**	**2.3**
SVHN	170K	92.4	263.2	30K	90.64	35.7	131K	93.0	25.3	172K	81.0	128.9	172K	83.5	75.4	83K	**95.21**	**2.5**
CIFAR-10	1M	89.0	887.2	30K	75.59	57.8	661K	89.2	88.2	1M	59.9	1017.3	1M	73.4	912.5	925K	**93.18**	**3.7**

Table 1: **Comparison with well-posed implicit models.** Number of trainable parameters, classification accuracy (%), and per-batch inference time (ms) for SphericalDEQ and five well-posed implicit baselines on MNIST, SVHN, and CIFAR-10. The best result for each dataset and metric is highlighted in **bold**.

5.3 Image Classification on ImageNet

We next test SphericalDEQ on a larger-scale benchmark: ImageNet-1K classification task (Deng et al., 2009). Following the experimental setting of (Lin et al., 2026), we use an Inception-V2 backbone (Ioffe & Szegedy, 2015) and replace its batch-normalization layers with depthwise SphericalDEQ layers. Full architectural, training, data-augmentation details are provided in Appendix E.

Table 2 compares SphericalDEQ with the equilibrium models reported by (Lin et al., 2026). Following their convention, NFE denotes the number of function evaluations required by an equilibrium layer. SphericalDEQ improves over the explicit Inception-V2 baseline while introducing only a modest computational overhead. Compared with the DEQ baselines, it achieves higher predictive performance at comparable or lower computational cost, without requiring iterative equilibrium computation. In particular, C-DEQ (the strongest competing method in accuracy) relies on consistency distillation of the original model, whereas SphericalDEQ is trained directly using the exact equilibrium formulation.

Model	# Params	NFE	Top1 Acc.	Top5 Acc.	Infe. (ms)
SphericalDEQ	13M	2	**76.15**	**93.12**	35.1
DEQ Bai et al. (2019)	18M	1	1.17	4.64	48.3
HyperDEQ Bai et al. (2022b)	18M	1	5.46	18.11	19.6
MDEQ Bai et al. (2020)	18M	1	1.44	4.24	19.7
C-DEQ Lin et al. (2026)	18M	1	47.12	68.71	20.6
DEQ Bai et al. (2019)	18M	2	8.12	19.43	64.4
HyperDEQ Bai et al. (2022b)	18M	2	27.82	41.59	30.8
MDEQ Bai et al. (2020)	18M	2	9.70	23.31	35.8
C-DEQ Lin et al. (2026)	18M	2	58.28	77.10	31.1
DEQ Bai et al. (2019)	18M	8	64.13	82.49	164.8
HyperDEQ Bai et al. (2022b)	18M	8	69.90	84.12	119.4
MDEQ Bai et al. (2020)	18M	8	71.29	89.57	151.2
C-DEQ Lin et al. (2026)	18M	8	74.04	91.45	99.1

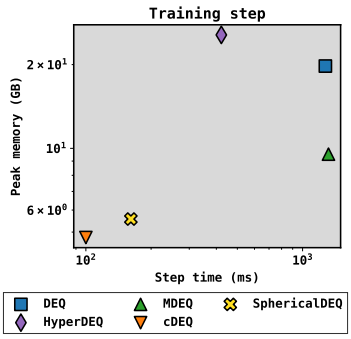


Table 2: **Left:** Results on Inception-V2 full training on ImageNet1K. **Right:** Training step timing compared to peak memory for all models. All timings are done on a batch size of 128.

5.4 Autoregressive Physical Simulations

We now evaluate the performance of the neural PDE models considered in Section 5.1. Each model is trained for next-step prediction and evaluated by autoregressively rolling out the predicted dynamics over the full trajectory. Complete experimental details are provided in Appendix E.2.

As shown in section 5.4, FNO-SphericalDEQ outperforms FNO-DEQ and its weight-tied variant FNO-WT both in rollout accuracy and in inference speed, achieving the lowest relative ℓ_2 rollout error in 11 of the 12 dataset/horizon combinations. On Active Matter and Gray–Scott, the advantage grows with the rollout length: all models perform similarly at short horizons, but FNO-SphericalDEQ accumulates error considerably more slowly. On Gray–Scott in particular, its error remains essentially constant along the rollout (between 0.448 and 0.453), whereas that of the baselines increases by 10–14%. Over the full horizon, FNO-SphericalDEQ reduces the error of the strongest baseline by about 12% on both datasets. On Viscoelastic Instability, by contrast, the improvement is largest at short horizons (22% lower error than FNO-DEQ over the first 15%) and narrows to 4% over the full rollout. FNO-SphericalDEQ is also substantially faster at inference, with a 14–17× speedup over FNO-WT and a 19–25× speedup over FNO-DEQ, depending on the dataset (e.g., 17.2 ms vs. 287.8 ms and 432.2 ms per batch on Viscoelastic Instability).

	FNO-SphericalDEQ					FNO-WT					FNO-DEQ				
Dataset	*15%*	*30%*	*60%*	*100%*	*Infer*	*15%*	*30%*	*60%*	*100%*	*Infer*	*15%*	*30%*	*60%*	*100%*	*Infer*
Viscoelastic In.	**0.4137**	**0.4295**	**0.4665**	**0.5190**	**17.2**	0.5507	0.5536	0.5585	0.5627	287.8	0.5321	0.5353	0.5395	0.5420	432.2
Active Matter	**0.3457**	**0.4153**	**0.6109**	**0.8377**	**17.3**	0.3648	0.4661	0.7250	0.9474	285.5	0.3468	0.4657	0.7304	0.9607	427.4
Gray–Scott	**0.4483**	**0.4515**	**0.4527**	**0.4521**	**80.5**	0.4690	0.4419	0.4767	0.5148	1140.7	0.4709	0.4482	0.4935	0.5376	1517.9

Table 3: **Comparison with other DEQ and weight-tied Neural PDE Solver.** We compare against FNO-DEQ(Marwah et al., 2023) and its weight-tied variant.Rollout relative ℓ_2 error at $\varepsilon = 0$. Rollout relative ℓ_2 error.Test error over the first 15%, 30%, 60%, and 100% of the maximum rollout horizon, together with per-batch inference speed (ms); FNO-SphericalDEQ is evaluated with the exact gradient. The best value in each column group is highlighted in **bold**.

6 Conclusion

We introduced SphericalDEQ, a family of deep equilibrium layers based on homogeneous normalization. The resulting structure yields a well-posed equilibrium for Lebesgue-almost every set of parameters and, crucially, reduces its computation to a linear solve followed by normalization, avoiding iterative nonlinear fixed-point solvers. It also admits closed-form parameter derivatives, enabling exact backpropagation, and can be naturally composed with explicit mixing operators within modern neural architectures. Across image classification and neural PDE simulation, SphericalDEQ substantially reduces the computational and memory overhead of conventional DEQs and other well-posed implicit models while matching or improving their predictive performance, scaling to ImageNet-1K classification and large-scale autoregressive physical simulation.

AI USE STATEMENT

In this work, we used AI tools to polish writing and to improve code efficiency. We confirm that we have reviewed all AI-assisted improvements in the method implementation. We **have not** used generative AI tools for producing theoretical results, and the rest of the required disclosure tasks are not applicable to this work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Donald G Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965.

Uri M Ascher, Steven J Ruuth, and Brian TR Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *Advances in neural information processing systems*, 33:5238–5250, 2020.

Shaojie Bai, Vladlen Koltun, and Zico Kolter. Stabilizing equilibrium models by jacobian regularization. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 554–565. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/bai21b.html.

Shaojie Bai, Zhengyang Geng, Yash Savani, and J Zico Kolter. Deep equilibrium optical flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 620–630, 2022a.

Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Neural deep equilibrium solvers. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=B0oHOwT5ENL.

C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19:577–593, 1965. URL https://api.semanticscholar.org/CorpusID:2802972.

Andreas Burger, Luca Thiede, Alan Aspuru-Guzik, and Nandita Vijaykumar. DEQuify your force field: More efficient simulations using deep equilibrium models. In *AI for Accelerated Materials Design - ICLR 2025*, 2025. URL https://openreview.net/forum?id=XACVRYePQQ.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L.

Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/21be9a4bd4f81549a9d1d241981cec3c-Paper.pdf.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1), January 2022. ISSN 1532-4435.

Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley Osher, and Wotao Yin. Jfb: Jacobian-free backpropagation for implicit networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6648–6656, 2022.

Mateusz Gabor, Tomasz Piotrowski, and Renato LG Cavalcante. Positive concave deep equilibrium models. In *International Conference on Machine Learning*, pp. 14365–14381. PMLR, 2024.

Tianxiang Gao, Hailiang Liu, Jia Liu, Hridesh Rajan, and Hongyang Gao. A global convergence theory for deep relu implicit networks via over-parameterization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=R332S76RjxS.

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=3EwcMzmUbNd.

Zhengyang Geng, Ashwini Pokle, and J Zico Kolter. One-step diffusion distillation via deep equilibrium models. *Advances in Neural Information Processing Systems*, 36:41914–41931, 2023.

Dobrik Georgiev, Joseph J Wilson, Davide Buffelli, and Pietro Liò. Deep equilibrium algorithmic reasoning. *Advances in Neural Information Processing Systems*, 37:33638–33667, 2024.

Davis Gilton, Gregory Ongie, and Rebecca Willett. Deep equilibrium architectures for inverse problems in imaging. *IEEE Transactions on Computational Imaging*, 7:1123–1133, 2021.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models, 2024. URL https://arxiv.org/abs/2402.00838.

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. *Advances in neural information processing systems*, 33:11984–11995, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Junchao Lin, Zenan Ling, Jingwen Xu, and Robert C Qiu. Consistency deep equilibrium models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=uw63l3T9un.

Zenan Ling, Xingyu Xie, Qiuhao Wang, Zongpeng Zhang, and Zhouchen Lin. Global convergence of over-parameterized deep equilibrium models. In *International Conference on Artificial Intelligence and Statistics*, pp. 767–787. PMLR, 2023.

Juncheng Liu, Bryan Hooi, Kenji Kawaguchi, and Xiaokui Xiao. Mgnni: Multiscale graph neural networks with implicit layers. *Advances in Neural Information Processing Systems*, 35:21358–21370, 2022a.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022b.

Tanya Marwah, Ashwini Pokle, J Zico Kolter, Zachary Lipton, Jianfeng Lu, and Andrej Risteski. Deep equilibrium based neural operators for steady-state pdes. *Advances in Neural Information Processing Systems*, 36:15716–15737, 2023.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 4. Granada, 2011.

Ruben Ohana, Michael McCabe, Lucas Thibaut Meyer, Rudy Morel, Fruzsina Julia Agocs, Miguel Beneitez, Marsha Berger, Blakesley Burkhart, Stuart B. Dalziel, Drummond Buschman Fielding, Daniel Fortunato, Jared A. Goldberg, Keiya Hirashima, Yan-Fei Jiang, Rich Kerswell, Suryanarayana Maddu, Jonah M. Miller, Payel Mukhopadhyay, Stefan S. Nixon, Jeff Shen, Romain Watteaux, Bruno Régaldo-Saint Blancard, François Rozet, Liam Holden Parker, Miles Cranmer, and Shirley Ho. The well: a large-scale collection of diverse physics simulations for machine learning. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=00Sx577BT3.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Ashwini Pokle, Zhengyang Geng, and J Zico Kolter. Deep equilibrium approaches to diffusion models. *Advances in Neural Information Processing Systems*, 35:37975–37990, 2022.

Ernest K Ryu and Stephen Boyd. Primer on monotone operator methods. *Appl. comput. math*, 15 (1):3–43, 2016.

Philipp Schleich, Marta Skreta, Lasse Bjørn Kristensen, Rodrigo Vargas-Hernandez, and Alan Aspuru-Guzik. Quantum deep equilibrium models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=CWhwKb0Q4k.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Pietro Sittoni and Francesco Tudisco. Subhomogeneous deep equilibrium models. In *International Conference on Machine Learning*, pp. 45794–45812. PMLR, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Haixiang Sun and Ye Shi. Understanding representation of deep equilibrium models from neural collapse perspective. *Advances in Neural Information Processing Systems*, 37:9634–9667, 2024.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Shuai Wang, Yao Teng, and Limin Wang. Deep equilibrium object detection. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6273–6283. IEEE, 2023.

Ezra Winston and J Zico Kolter. Monotone operator equilibrium networks. *Advances in neural information processing systems*, 33:10718–10728, 2020.

Zhoutong Wu, Yimu Zhang, Cong Fang, and Zhouchen Lin. Separation and bias of deep equilibrium models on expressivity and learning dynamics. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=UJ9k3j93MD.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

A EXPERIMENTAL DETAILS: MOTIVATION SECTION

The equilibrium state has dimension 128, while the hidden layer inside the DEQ block has width 1280. The fixed point is computed using plain fixed-point iteration with a maximum of 30 iterations and a stopping tolerance of 10^{-4}. To compute the gradient of the DEQ block, we use Jacobian-Free Backpropagation (JFB) (Fung et al., 2022). The model is trained for 20 epochs using the AdamW optimizer with learning rate 10^{-3}, weight decay 10^{-5}, a batch size of 256, and a cosine-decay learning-rate schedule. We initialize all linear layers with Xavier uniform initialization, and all biases to zero.

B PROOF OF MAIN THEORETICAL RESULTS

B.1 PROOF OF LEMMA 4.4

Proof. ($\Longrightarrow$) Let z be a fixed point of $\mathcal{F}_\alpha$. Then by definition of $\mathcal{F}_\alpha$ we have

$$Wz + \alpha \odot b = \varphi(Wz + \alpha \odot b) \odot z.$$

By applying φ on both sides of the equation and by using homogeneity, we get

$$\varphi(Wz + \alpha \odot b) = \varphi\left(\varphi(Wz + \alpha \odot b) \odot z\right) = \varphi(z) \odot \varphi(Wz + \alpha \odot b),$$

which in turn implies $\varphi(z) = \mathbf{1}$.

($\Longleftarrow$) Now assume that z solves the constrained linear system

$$Wz + \alpha \odot b = z, \quad \text{subject to} \quad \varphi(z) = \mathbf{1}.$$

Then, we have

$$\mathcal{F}_\alpha(z) = (Wz + \alpha \odot b) \oslash \varphi(Wz + \alpha \odot b) = z \oslash \varphi(z) \underset{\varphi(z)=\mathbf{1}}{=} z$$

□

B.2 PROOF OF THEOREM 4.5

Proof. We aim to show that there exist an entrywise positive $\alpha^* \in \mathbb{R}^n$ and a unique $z^* \in \mathbb{S}_\varphi$ such that

$$\mathcal{F}_{\alpha^*}(z^*) = z^*. \tag{2}$$

To this end, thanks to Lemma 4.4, we can equivalently study the problem

$$Wz^* + \alpha^* \odot b = z^* \qquad \text{subject to} \qquad \varphi(z^*) = \mathbf{1}. \tag{3}$$

We now solve eq. (3). For fixed $\alpha \in \mathbb{R}^n_{>0}$, the equation $Wz + \alpha \odot b = z$ is equivalent to

$$\begin{aligned} Wz + \alpha \odot b &= z \\ (I - W)z &= \alpha \odot b \\ z &= (I - W)^{-1}(\alpha \odot b), \end{aligned}$$

where the last step uses the fact that $(I - W)$ is non-singular, since 1 is not an eigenvalue of W by hypothesis. Setting

$$y := (I - W)^{-1}b,$$

and by using the fact that $W = \sum_{i=1}^n W_i^\top \otimes \Lambda_i$ (Λ_i diagonal), we have the commutation relation

$$W(I \otimes \mathrm{diag}(\alpha)) = \sum_i W_i^\top \otimes \Lambda_i \mathrm{diag}(\alpha) = \sum_i W_i^\top \otimes \mathrm{diag}(\alpha)\Lambda_i = (I \otimes \mathrm{diag}(\alpha))W.$$

Since W commutes with $I \otimes \mathrm{diag}(\alpha)$, also $(I - W)^{-1}$ does because the inverse is a matrix polynomial in $I - W$, and therefore by using the entrywise notation, we have $(I - W)^{-1}(\alpha \odot b) = \alpha \odot \big((I - W)^{-1}b\big)$. We thus obtained a parametric family of candidate solutions to the unconstrained equation in eq. (3), given by

$$z^*(\alpha) := \alpha \odot y, \qquad \alpha \in \mathbb{R}^n_{>0}.$$

It remains to choose α so that the constraint $\varphi(z^*(\alpha)) = \mathbf{1}$ holds. Since $\varphi \in \mathcal{H}(\mathbb{R}^{n\times d}, \mathbb{R}^n)$ and $\alpha \in \mathbb{R}^n_{>0}$ (so that $|\alpha| = \alpha$), we have

$$\varphi(z^*(\alpha)) = \varphi(\alpha \odot y) = \alpha \odot \varphi(y).$$

Setting this equal to $\mathbf{1}$ and solving for α yields

$$\alpha^* := \mathbf{1} \oslash \varphi(y),$$

which is well defined and lies in $\mathbb{R}^n_{>0}$ provided $\varphi(y) > 0$ entrywise. This gives the desired pair (z^*, α^*) solving eq. (2). Uniqueness follows by noticing that for each choice of α, for $I - W$ invertible, there is a unique possible solution to the unconstrained linear system, which concludes the proof. □

B.3 Proof of Theorem 4.6

Proof. The α^* that guarantees the existence of a unique fixed point in Theorem 4.5 is $\alpha^* = \mathbf{1}\oslash\varphi(y)$. We compute now, for $z = y \oslash \varphi(y)$, the fixed point map $\mathcal{F}_{\alpha^*}$, obtaining

$$\mathcal{F}_{\alpha^*}(z) = W\left(y \oslash \varphi(y)\right) + b \oslash \varphi(y) = (Wy + b) \oslash \varphi(y) \underset{(I-W)^{-1}y=b}{=} y \oslash \varphi(y) = z,$$

where in the first equality we used the fact that the structure of W in Theorem 4.5 allows it to commute with the entrywise product, and in the last passage we used the fact that $y = (I-W)b$. □

B.4 Proof of Theorem 4.7

Proof. We can easily calculate

$$\frac{\partial z^*}{\partial y} = \operatorname{diag}\left(1 \oslash \varphi(y)\right) - \operatorname{diag}\left(y \oslash \varphi(y)^2\right) J_\varphi =: K,$$

where J_φ is Jacobian of the vector valued map φ. Using the chain rule, we get

$$\frac{\partial z^*}{\partial \operatorname{vec}(W)} = \frac{\partial z^*}{\partial y}\frac{\partial y^*}{\partial \operatorname{vec}(W)} = K(I-W)^{-1}(y^\top \otimes I), \quad \frac{\partial z^*}{\partial b} = \frac{\partial z^*}{\partial y}\frac{\partial y^*}{\partial b} = K(I-W)^{-1},$$

and therefore

$$\begin{aligned}
&\nabla_y L = g \oslash \varphi(y) - J_\varphi^\top\left(g \odot y \oslash \varphi(y)^2\right), \\
&\nabla_b L(W,b) = (I-W)^{-\top}\nabla_y L \\
&\nabla_W L(W,b) = (I-W)^{-\top}\nabla_y L\, y(W,b)^\top = \nabla_b L(W,b) y(W,b)^\top.
\end{aligned}$$

□

C Ablation: Exact gradient vs Jacobian-free backpropagation

We compare two backpropagation methods: the Jacobian-free approach and the exact method. We do not compare them with higher-order approximations, such as phantom gradients, or with implicit backpropagation. Higher-order phantom-gradient methods require additional computational and memory resources, while implicit backpropagation introduces additional computational overhead. Instead, the Jacobian-free backpropagation needs a similar amount of compute and memory.

Well-posed DEQ task As we can see from table 4, the exact gradient always improves the performance over the Jacobian-free approach.

D Numerical sanity check

We checked numerically in double precision (FP64) the results of theorem 4.5 in the following scenario: $x_{ij} \sim N(0,1), x_{ij}$ i.i.d $x \in \mathbb{R}^{4\times 32}$, the bias initialized to zeros, and the matrix weights

Dataset	*Jacobian-Free Backprop*	*Exact Gradient*
MNIST	99.60	**99.61**
SVHN	94.99	**95.21**
CIFAR-10	89.48	**93.18**

Table 4: **Comparison between Jacobian-free and exact backpropagation.** We report classification accuracy (%) on MNIST, SVHN, and CIFAR-10. The best result for each dataset is highlighted in **bold**.

	Jacobian-Free Backprop					Exact Gradient				
Dataset	*15%*	*30%*	*60%*	*100%*	*Infer*	*15%*	*30%*	*60%*	*100%*	*Infer*
Viscoelastic In.	**0.3999**	**0.4188**	**0.4604**	**0.5167**	17.2	0.4137	0.4295	0.4665	0.5190	17.2
Active Matter	**0.3297**	0.4322	0.6753	0.9148	17.3	0.3457	**0.4153**	**0.6109**	**0.8377**	17.3
Gray–Scott	**0.4475**	**0.4335**	**0.4378**	**0.4341**	80.5	0.4483	0.4515	0.4527	0.4521	80.5

Table 5: **Comparison between Jacobian-free and exact backpropagation.** Rollout relative ℓ_2 error at $\varepsilon = 0$ over the first 15%, 30%, 60%, and 100% of the maximum rollout horizon, together with per-batch inference speed (ms), for FNO-SphericalDEQ trained with Jacobian-free backpropagation (JFB) or with the exact gradient. For each dataset and horizon, the lower error of the two is highlighted in **bold**.

initialized with the Kaiming uniform, we obtain the following results, ensuring the correctness of the theorem 4.5

$$|\mathrm{RMS}(z^*) - 1| = 1.7 \times 10^{-9}$$
$$\|\mathrm{SphericalDEQ}(z^*) - z^*\|_\infty = 8.147 \times 10^{-9}$$

E Experimental Details

All models are implemented in PyTorch (Paszke et al., 2019) and trained on a single NVIDIA GPU.

E.1 Image classification

Datasets. We evaluate on MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011) and CIFAR-10 (Krizhevsky et al., 2009), using the standard train/test splits distributed with `torchvision`. Images are normalized with the per-channel statistics of the training set.

Optimization. The learning rate follows a cosine-decay schedule from the peak value to the final value reported in Table 7. On CIFAR-10 we use SGD with Nesterov momentum, while on SVHN we use AdamW; the complete set of hyperparameters is listed in Table 7.

E.2 Physical simulation

Datasets. We use three two-dimensional datasets from The Well (Ohana et al., 2024): *viscoelastic_instability*, *active_matter* and *gray_scott_reaction_diffusion*. They span very different physical regimes: elasto-inertial turbulence in dilute polymer solutions, a continuum model of rod-like active particles suspended in a Stokes fluid, and a two-species reaction. Diffusion system differ substantially in the number of state variables (8, 11 and 2 fields respectively) and in the number of physical parameters conditioning each trajectory (5, 3 and 2). Trajectories are spatially downsampled to a common 128×128 grid and, for Gray–Scott, temporally subsampled with a stride of 10, so that consecutive frames exhibit a comparable amount of change across datasets. Fields are standardized channel-wise with the training-set mean and standard deviation; for the viscoelastic dataset, whose

Model	# Params	Top1 Acc.	Top5 Acc.
SphericalDEQ (exact gradient)	13M	**76.15**	93.12
SphericalDEQ (phantom gradient)	13M	76.10	**93.17**

Table 6: Results on Inception-V2 full training on ImageNet1K, comparison between exact and phantom gradient implementation of SphericalDEQ.

	MNIST	*SVHN*	*CIFAR-10*
Architecture			
Channels C	32	32	96
Blocks L	8	8	10
Mixing kernel size k	3	3	3
Bias	yes	yes	yes
Dropout	0.1	0.1	0.1
RMS stabilizer ϵ	10^{-8}	10^{-8}	10^{-8}
Trainable parameters	83K	83K	925K
Optimization			
Optimizer	AdamW	AdamW	SGD
Epochs	50	40	300
Batch size	128	128	256
Peak learning rate	10^{-3}	10^{-3}	5×10^{-2}
Final learning rate	10^{-3}	10^{-5}	10^{-4}
Schedule	cosine	cosine	cosine
Weight decay	10^{-6}	10^{-6}	10^{-6}
Momentum	–	–	0.9 (Nesterov)
(β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)	–

Table 7: Well-posed DEQ training hyperparameters.

pressure field spans several orders of magnitude, standardization is preceded by a logarithmic rescaling.

Task. All models are trained as autoregressive next-step operators: given a single input frame together with the physical parameters of the trajectory, the model predicts the following frame. Training minimizes the loss over an autoregressive unroll of K steps, where $K = 3$ for the viscoelastic dataset, $K = 5$ for active matter and $K = 1$ for Gray–Scott. At test time, the models are rolled out autoregressively over the full horizon of the dataset (19, 80 and 50 steps respectively), and we report the relative ℓ_2 error accumulated over the first 15%, 30%, 60% and 100% of that horizon.

Optimization. All models are trained with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.99$) and a cosine-decay learning rate schedule annealed over the full number of optimization steps down to the final learning rate of Table 8. Training uses full `float32` precision, no gradient accumulation, and the seed 42; for the viscoelastic dataset an exponential moving average of the weights with decay 0.998 is used for evaluation. The three models share *identical* data and optimization settings within each dataset, so that the comparison isolates the effect of the pondering stage.

Robustness to input noise. To probe the stability of the rollout, we repeat the evaluation with Gaussian noise of variance ε injected into the input frame, for $\varepsilon \in \{0,\ 10^{-4},\ 10^{-3}\}$. The noise-free results are reported in Table section 5.4 and the perturbed ones in Tables appendix F and appendix F.

	Viscoelastic	*Active Matter*	*Gray–Scott*
Data			
Temporal stride	1	1	10
Input frames	1	1	1
Rollout horizon (steps)	19	21	500
State fields (in = out)	8	11	2
Physical parameters	5	3	2
Normalization	$\log + z$-score	z-score	none
Optimization			
Autoregressive unroll K	3	5	1
Epochs	100	50	3
Batch size	64	64	256
Peak learning rate	10^{-4}	5×10^{-4}	10^{-5}
Final learning rate	10^{-5}	10^{-6}	10^{-6}
Schedule	cosine	cosine	cosine
Weight decay	10^{-5}	10^{-4}	10^{-5}
(β_1, β_2)		$(0.9,\ 0.99)$	

Table 8: **Physical simulation: data and optimization.** Settings shared by FNO-SphericalDEQ, FNO-WT and FNO-DEQ within each dataset, so that the comparison isolates the pondering stage.

	FNO-SphericalDEQ	*FNO-WT*	*FNO-DEQ*
Fourier modes	12×12	12×12	12×12
Width	48	32	32
Depth	2	–	–
Blocks	–	1	1
Domain padding	–	9	9
Forward pass	direct solve	fixed unroll	Anderson, 16 steps
Backward pass	1-step Jacobian-free	truncated BPTT, 3 steps	phantom, $\tau = 0.8$, S=3
Warm-up	–	–	300 steps, 8 iterations
Trainable parameters	2M (matched across models)		

Table 9: **Physical simulation model settings.** Width and depth of FNO-SphericalDEQ are reported as Viscoelastic / Active Matter / Gray–Scott; the baselines use the same configuration on all three datasets. Widths and depths are chosen so that the three models are parameter-matched.

F ROLLOUT WITH NOISE INJECTION

	FNO-SphericalDEQ				FNO-WT				FNO-DEQ			
Dataset	*15%*	*30%*	*60%*	*100%*	*15%*	*30%*	*60%*	*100%*	*15%*	*30%*	*60%*	*100%*
Viscoelastic	**0.4137**	**0.4295**	**0.4665**	**0.5189**	0.5507	0.5536	0.5585	0.5627	0.5322	0.5353	0.5395	0.5419
Active Matter	**0.3456**	**0.4150**	**0.6106**	**0.8387**	0.3648	0.4660	0.7250	0.9476	0.3468	0.4657	0.7304	0.9609
Gray–Scott	**0.4483**	**0.4515**	**0.4527**	**0.4521**	0.4692	0.4422	0.4770	0.5151	0.4712	0.4485	0.4938	0.5380

Table 10: **Rollout relative ℓ_2 error** at $\varepsilon = 10^{-3}$. Test error over the first 15%, 30%, 60%, and 100% of the maximum rollout horizon. The best model at each horizon is highlighted in **bold**.

	FNO-SphericalDEQ				FNO-WT				FNO-DEQ			
Dataset	*15%*	*30%*	*60%*	*100%*	*15%*	*30%*	*60%*	*100%*	*15%*	*30%*	*60%*	*100%*
Viscoelastic	**0.4138**	**0.4296**	**0.4665**	**0.5189**	0.5507	0.5536	0.5584	0.5627	0.5321	0.5353	0.5395	0.5419
Active Matter	**0.3457**	**0.4154**	**0.6121**	**0.8397**	0.3649	0.4660	0.7250	0.9478	0.3468	0.4658	0.7307	0.9613
Gray–Scott	**0.4486**	**0.4518**	**0.4528**	**0.4520**	0.4717	0.4447	0.4795	0.5179	0.4738	0.4512	0.4965	0.5409

Table 11: **Rollout relative ℓ_2 error** at $\varepsilon = 10^{-2}$. Test error over the first 15%, 30%, 60%, and 100% of the maximum rollout horizon. The best model at each horizon is highlighted in **bold**.