NO FREE SOUP FOR PINNS: JUST KEEP THE BEST

Anonymous authors
Paper under double-blind review

ABSTRACT

Averaging the weights of a pool of fine-tuned models into a model soup yields ensemble-level accuracy at the cost of a single inference, a practice well established in vision and language. For Scientific Machine Learning (SciML) and its techniques, such as Physics-Informed Neural Networks (PINNs), this is especially attractive: the result stays a single continuous, differentiable surrogate. We present the first systematic benchmark of weight averaging for PINNs for the forward problem, spanning several PDE families in both soft- and hard-constrained form and a range of widths, in double precision throughout. The odd activations standard in PINNs induce a sign symmetry that permutation matching leaves intact. We therefore enlarge the matching group to $(\mathbb{Z}/2\mathbb{Z})^d \rtimes S_d$, an extension we call Sign Re-Basin. We evaluate it across training regimes, from independently trained ingredients to models that share most of their optimisation trajectory. The output-space ensemble, exact by superposition on a linear PDE, serves as the control that isolates parameter-space from function-space averaging. Using a reference-free selection criterion, the greedy recipe almost never accepts a second ingredient and never a third, and using the reference solution in its acceptance test barely changes this. When a soup does improve on the selected ingredient, the gain is within selection noise, i.e. the gap between the model selected by a reference-free criterion and the best model in the pool. The ensemble stays accurate but loses the main advantage of a PINN over classical mesh-based solvers, a single continuous and differentiable representation, multiplying the query cost by k for a variance reduction that model selection already provides. Given a pool of PINNs, just keep the best model.

1 INTRODUCTION

Physics-Informed Neural Networks (Raissi et al., 2019) solve a PDE by encoding it in the loss function, so that the solution they return is a continuous and differentiable representation of the field, available at any point of the domain rather than at the nodes of a mesh as in classical discretisation methods. Averaging the weights of a pool of fine-tuned models into a *model soup* matches the accuracy of their ensemble at the cost of a single inference (Wortsman et al., 2022), an attractive prospect for a PINN, since the average is again a single network, and therefore again such a representation, whereas an ensemble of k of them is evaluated k times at every query.

Naive weight averaging is only meaningful modulo the symmetries of the parametrisation. The best known is permutation symmetry: relabelling the hidden units of a layer leaves the realised function unchanged, so independently trained networks must be brought into a common labelling before averaging, as Git Re-Basin does in vision and language (Ainsworth et al., 2023). PINNs, however, typically use odd activations such as $\tanh$, which add a sign symmetry. A soup works only if its ingredients lie in a region where the loss stays low along the segment joining them, because they share training or because a symmetry maps one onto the other's basin (Frankle et al., 2020; Entezari et al., 2022). We give PINNs every chance to meet this premise, with regimes from independent to largely shared, five averaging methods and an enlarged symmetry group, and it is almost never met: the loss sees the network through $\boldsymbol{\theta} \mapsto u(\cdot\,; \boldsymbol{\theta})$, which does not commute with averaging.

Our contributions are the following. (i) **A systematic benchmark.** The first, to our knowledge, to study weight averaging for PINNs on the forward problem, 864 networks trained, aligned and averaged in double precision, against the output-space ensemble as a control. (ii) **Sign Re-Basin.** An extension of weight and activation matching that enlarges the group searched to the signed per-

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.

mutations $(\mathbb{Z}/2\mathbb{Z})^d \rtimes S_d$ and recovers the sign component in closed form, at the same per-layer cost as permutation-only matching. (iii) **A new possible PINN pathology.** PINNs fail in structured, well-characterised ways (Krishnapriyan et al., 2021; Rathore et al., 2024), and weight averaging joins this list: the greedy recipe almost never accepts a second ingredient, few soups beat the best single ingredient beyond selection noise, and almost every pair is separated by a loss barrier.

2 METHODOLOGY

2.1 ODD ACTIVATIONS INDUCE A SIGN SYMMETRY

Odd activations introduce an additional symmetry, which is simple to state. Write a single-hidden-layer PINN acting on the input $\boldsymbol{x} = (t, x)$ as $u(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{v}^\top \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, where $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{v})$ are the network parameters and σ is an activation function. If σ is odd and $\boldsymbol{S} = \mathrm{diag}(\epsilon_1, \ldots, \epsilon_d)$ with $\epsilon_i \in \{-1, +1\}$, then $\sigma(\boldsymbol{S}\,\boldsymbol{z}) = \boldsymbol{S}\,\sigma(\boldsymbol{z})$ and $\boldsymbol{S}^2 = \boldsymbol{I}$, so the map

$$(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{v}) \;\longmapsto\; (\boldsymbol{S}\,\boldsymbol{W}, \boldsymbol{S}\,\boldsymbol{b}, \boldsymbol{S}\,\boldsymbol{v})$$

leaves $u(\cdot\,; \boldsymbol{\theta})$ unchanged pointwise: flipping a hidden unit's incoming weights, its bias and its outgoing weight is invisible at the output.

The standard PINN activation, $\tanh$, is odd. Two ingredients can therefore realise the same function while sitting far apart in parameter space solely because of their signs. Take $d = 2$ and $\boldsymbol{S} = -\boldsymbol{I}$:

$$\boldsymbol{\theta} = \left(\begin{pmatrix} 2 & -1 \\ 1 & 3 \end{pmatrix}, \begin{pmatrix} -1 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix} \right), \qquad \boldsymbol{\theta}' = \left(\begin{pmatrix} -2 & 1 \\ -1 & -3 \end{pmatrix}, \begin{pmatrix} 1 \\ -2 \end{pmatrix}, \begin{pmatrix} -3 \\ -1 \end{pmatrix} \right),$$

so that $\boldsymbol{\theta}' = -\boldsymbol{\theta}$ and $u(\cdot\,; \boldsymbol{\theta}') \equiv u(\cdot\,; \boldsymbol{\theta})$. Their uniform soup is $\frac{1}{2}(\boldsymbol{\theta} + \boldsymbol{\theta}') = \boldsymbol{0}$, so two copies of the same solution average to the zero network. This can occur between any two independently initialised networks and permutation matching over S_2 cannot repair this. We therefore enlarge the group searched by both of the matching heuristics proposed by Git Re-Basin to the signed permutations described by $(\mathbb{Z}/2\mathbb{Z})^2 \rtimes S_2$. We prove (Proposition 5) that for a layer of width d, within each layerwise subproblem the sign component admits a closed form, so the search collapses to a single linear assignment on the entrywise absolute value of the same similarity matrix, at the $O(d^3)$ cost of the permutation-only problem. This, along with the matching algorithms, is derived in Appendix C.

2.2 EXPERIMENTAL DESIGN OVERVIEW

The benchmark is a full factorial design over the PDE, its constraint type, the architecture and the regime. A set of six Cauchy problems, each solved in both soft- and hard-constrained form, is crossed with three architectures and three training regimes, giving 108 configurations of $k = 8$ ingredients each, for a total of 864 networks trained in double precision. Appendix B specifies and justifies the protocol (Tables 3–5; PDE suite in Appendix B.1).

Each axis addresses a hypothesis about when averaging should work, argued in full in Appendix B: the distinction between linear and nonlinear PDEs isolates the equation as a cause (see below); width is swept at a fixed depth because the barrier between independently trained solutions is predicted to narrow with it (Entezari et al., 2022; Ainsworth et al., 2023); and the regimes interpolate, at fixed budget, from ingredients with nothing in common (`independent`) to ingredients sharing most of their training (the *branching* regimes `early` and `late`), where alignment should be unnecessary.

A soup $\bar{\boldsymbol{\theta}}$ and an output ensemble $\bar{u}$ average the ingredients with the same weights $\beta_i \geq 0$, summing to one, in parameter and in function space. Therefore, for a linear differential operator $\mathcal{F}$,

$$\mathcal{F}[\bar{u}] = \mathcal{F}\left[\sum_{i=1}^{k} \beta_i\, u(\cdot\,; \boldsymbol{\theta}_i)\right] = \sum_{i=1}^{k} \beta_i\, \mathcal{F}[u(\cdot\,; \boldsymbol{\theta}_i)], \tag{1}$$

so on a linear PDE the ensemble is exact whenever its ingredients are, and a failure there cannot be blamed on the equation. The two averages differ pointwise by the *nonlinear distortion*

$$\Delta_{\mathrm{NL}}(\boldsymbol{x}) = \bar{u}(\boldsymbol{x}) - u(\boldsymbol{x}; \bar{\boldsymbol{\theta}}), \tag{2}$$

which vanishes for an affine parametrisation and therefore isolates the nonlinearity of $\boldsymbol{\theta} \mapsto u(\cdot\,; \boldsymbol{\theta})$.

Against every soup we place two references: the best single ingredient, which is what one would keep in practice, and the output ensemble. Predictors are scored by the relative L^2 error against the reference solution, an oracle unavailable in practice, and by the reference-free *physics score*, the unweighted PINN loss evaluated on held-out points,

$$\mathrm{Score}(\boldsymbol{\theta}) = \mathcal{L}_{\mathrm{PDE}}(\boldsymbol{\theta}) + \mathcal{L}_{\mathrm{BC}}(\boldsymbol{\theta}) + \mathcal{L}_{\mathrm{IC}}(\boldsymbol{\theta}), \tag{3}$$

which reduces to its PDE term under hard constraints. Pairwise, we also measure the loss barrier between ingredients (Section 3.4). The score is defined on parameters, so ensembles can only be ranked by the oracle error, a comparison deliberately generous to it. Soups are ranked under both: the score, a reference-free proxy for the error (Kendall $\tau = 0.54$ pooled within configurations, Appendix D.7), and the oracle error, an upper bound on what any proxy could achieve.

3 Results

The PDE suite, training regimes and averaging methods that define a *configuration* of the benchmark are specified in Appendix B; the full per-configuration results are reported in Appendix D.

3.1 Most Soups Do Not Improve on the Best Single Ingredient

Figure 1 reports, on the left, the best soup each configuration admits over the five averaging methods of the study (see Table 6), considering both recipes (uniform and greedy) of Wortsman et al. (2022) and every pool size $k \leq 8$; on the right, an identical view for the best ensemble, ranked by the oracle error since the physics score is defined on parameters.

The results show *a generalised degradation of the quality of the solution in the soup case, with a moderate improvement in the ensemble one*. In the case of the hard-constrained Helmholtz equation, whose differential operator is linear, the relative L^2 error of the soup is almost five orders of magnitude larger than that of its best ingredient under `independent`, and more than two under `early` and `late`. In this aggregated view, the soup reaches the improvement side of the scale only on soft-constrained Allen–Cahn, an artefact of the fact that its ingredients have all collapsed to $u \equiv 0$. Meanwhile, the best ensemble, selected from the very same pool of ingredients, offers a moderate improvement under `independent`. This advantage shrinks under `early` and vanishes under `late`, possibly as a consequence of a loss of diversity due to their shared training budget.

A similar trend is observed in the rest of the linear equations, where equation 1 holds, as confirmed experimentally. Parameter-space averaging thus performs worst precisely on the PDEs where output-space averaging is provably exact. The gain from the ensemble is more subtle on nonlinear PDEs, where the superposition argument does not hold, while the degradation from the soup is not as pronounced. Lastly, the asymmetry between soups from different constraint formulations is consistent with the blow-up of the boundary residual, a term that hard-constrained networks zero by construction for every $\boldsymbol{\theta}$, averaged ones included. This fact is presented in Appendix D.1, while the individual configurations in which a soup does improve are shown in Appendix F.

3.2 How Many Ingredients Can Be Averaged, and at What Cost

The preceding subsection reports the best soup each configuration admits, irrespective of pool size. Two complementary questions remain: (a) how many ingredients a principled recipe is willing to combine at all, and (b) what the damage looks like when the pool size is imposed rather than chosen. The greedy recipe of Wortsman et al. (2022) answers the first by construction since, given a ranking by quality, it appends an ingredient only when the held-out criterion improves; the uniform recipe answers the second, since it averages whatever pool it is handed.

Figure 2 shows that, in parameter space, *the greedy recipe essentially never gets past the first ingredient* (the best in the ranking): a second one is accepted in a marginal fraction of runs, a third in none, and under `independent` not a single acceptance occurs. This is not caused by the reference-free criterion: replacing it by the oracle relative L^2 error leaves the results qualitatively unchanged. The identical procedure applied to the same ingredients in output space accepts a second member in about half of the runs and occasionally all eight. Consistently, *every best soup of the previous subsection is uniform*, most often over two ingredients: once the greedy soups that accepted nothing are excluded, no greedy soup is ever competitive.

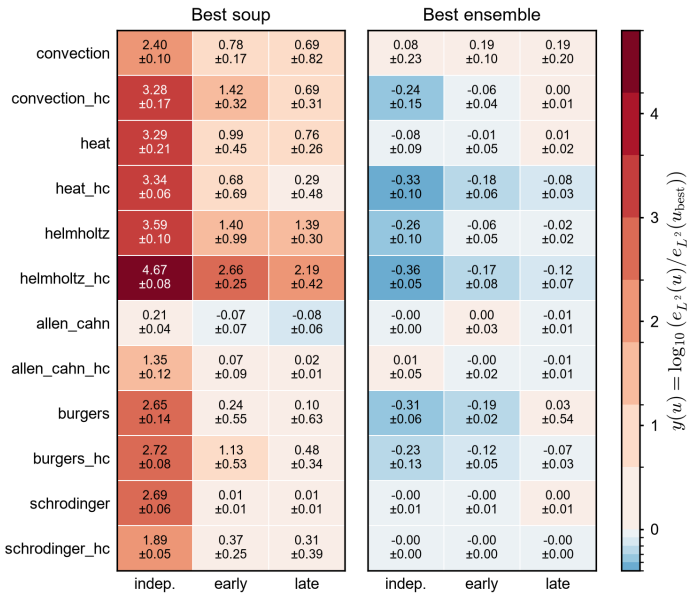


Figure 1: **Most soups do not improve on the best single ingredient.** Comparison of the top-performing soup and ensemble aggregated by architectures and separated by training regime (red tones for degradation and blue for improvement). Every cell represents the best over the five averaging methods and two recipes from the study.

Furthermore, imposing the pool size k separates two failure behaviours. Under `independent` the damage is a discontinuity at the first average and the curve is flat afterwards: any averaging is harmful, and two ingredients already produce the full degradation. Under the branching regimes the degradation accumulates steadily instead, so shared training makes the first average tolerable, but each additional ingredient further increases the error. In contrast, the ensemble saturates after a few members under `independent` and stays at the level of the best ingredient under the branching regimes, a difference visible only on a separate scale. The sweep itself is reported in Appendix D.2.

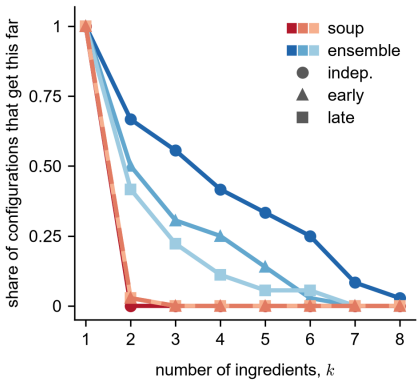


Figure 2: **Greedy recipe survival curves.** Share of greedy runs that retain at least $k \leq 8$ ingredients, by PDE family and training regime. Soup curves (red) pool the five averaging methods, 180 runs per regime; output-space runs (blue) are the 36 ensembles per regime. The soups collapse at $k = 2$ and reach zero at $k = 3$; the ensemble curves decay as more ingredients are added.

3.3 Soups Against Ensembles

Figure 3 reports that under `independent` the nonlinear distortion of equation 2 saturates at the scale of $\|u^\star\|$ (the reference solution) itself from the first average onwards, i.e. the soup is not a degraded approximation but a different function altogether; under the branching regimes it sits two orders of magnitude lower and grows with the pool, mirroring the growth of the error with pool size. Measured as a fraction, it accounts, in the median, for the whole of the soup's error on every linear operator, in every regime and at every pool size, consistent with superposition: when the ensemble is exact, the soup's error is entirely due to the distortion. Only on nonlinear operators with shared training does the fraction fall, and even there it climbs back towards one as the pool grows. Configuration by configuration, the two combinations do not trade off: their performances are positively correlated except under `late`, and the ensemble almost always obtains the larger gain (see Appendix D.3).

3.4 The Geometry Along the Segment

The *loss barrier* (Frankle et al., 2020) measures how far the loss rises along the segment joining two ingredients:

$$B(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \sup_{\alpha \in [0,1]} \mathcal{L}(\alpha \boldsymbol{\theta}_i + (1-\alpha)\boldsymbol{\theta}_j) - \max\left(\mathcal{L}(\boldsymbol{\theta}_i), \mathcal{L}(\boldsymbol{\theta}_j)\right). \tag{4}$$

A barrier $B = 0$ (its minimum) signals *linear mode connectivity* (LMC), i.e. every convex combination of the pair is no worse than its worse endpoint, so averaging is a priori safe. Here, $\mathcal{L}$ is evaluated as the physics score of equation 3. Figure 4 sweeps this barrier along the segment joining each configuration's two top-ranked ingredients, before and after each matching criterion, normalised by the score of the worse endpoint. LMC is almost never observed: the barrier is strictly positive in every sweep but 4 of the 108 pairs, with its maximum at or next to $\alpha = 1/2$ in the median of every regime and method, which corresponds to the uniform two-ingredient soup. The barrier maximum therefore coincides with the two-ingredient soup itself, rather than merely correlating with its failure. Its magnitude tracks shared training, falling by orders of magnitude from `independent` to `late`, yet still sits above the endpoints' own loss scale in the median at the end of that axis: shared training greatly improves connectivity but does not achieve linear mode connectivity.

Alignment reduces the barrier far less than needed, and only when the ingredients share no training history. Activation matching lowers the barrier under `independent` and raises it under the branching regimes. Weight matching barely moves it, and under the branching regimes the permutation it recovers is exactly the identity (see next experiment). Ingredients that share a trajectory already lie close in parameter space, and a symmetry transformation moves one of them away. Even the best-connected pair of the independent regime remains far from connectivity.

Nevertheless, measured in relative L^2 error against the reference, the midpoint is a different function rather than a degraded solution. Under `independent` it reaches the error of $u \equiv 0$: averaging two trained PINNs is no better than returning zero. The branching regimes leave the midpoint near its endpoints, and alignment moves it in the same directions and by the same relative amounts as the barrier, consistent with both quantities measuring the same effect (see Appendix D).

3.5 What the Sign-Aware Alignment Actually Does

The preceding subsection evaluates alignment through its effect on the barrier, which does not reveal whether a non-trivial transformation was applied at all. Since the re-basin action is a reindexing of hidden units followed by a sign flip, we recover the applied transformation directly from the aligned checkpoints. Table 1 reports it beside the barrier-based proxy it replaces.

Under the branching regimes, weight matching returns the identity. It does so on every pair that Figure 4 sweeps, and across the full pool of ingredients on all but three of them, each a collapsed Allen–Cahn model whose near-trivial weights provide no signal for the matching; under `independent` it returns the identity in no case at all. Its curve in Figure 4 therefore coincides with the pre-alignment curve.

Activation matching keeps finding a non-trivial permutation, and the fraction of units it leaves in place says what those permutations are: a handful of transposed units on an otherwise untouched network. These few swaps are enough to raise the barrier that weight matching leaves unchanged. Under `independent` the same statistic falls to the random-permutation baseline, indicating that the matching finds no exploitable structure. Under the branching regimes its identity rate falls monotonically with width, for Sign Re-Basin as for Git Re-Basin, while weight matching shows no width dependence: wider networks are not easier to align, they only give the criterion more units to permute unnecessarily. Nor does the barrier fall with width (Appendix D.6), contrary to the prediction of the linear mode connectivity literature.

Accounting for the sign symmetry of $\tanh$ lowers the barrier by less than an order of magnitude, and only under `independent` (Table 2). Sign Re-Basin leaves the lower barrier in most of the paired comparisons under `independent`, *where it inverts about half of all units*; once the ingredients share a trajectory it inverts almost none, and under weight matching it returns exactly the element Git Re-Basin had already found. The margin is small, but it is measured on the same pair of models aligned by both methods, so it reflects the sign component alone.

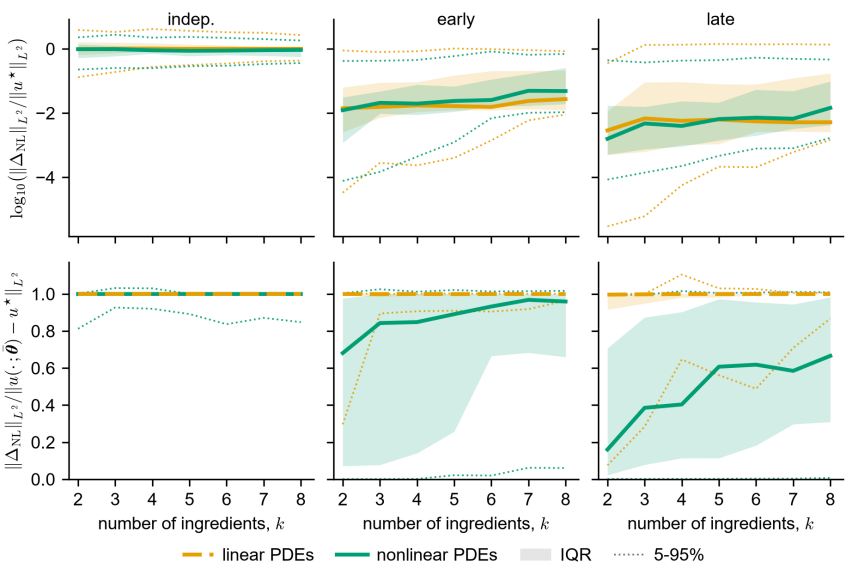


Figure 3: **Magnitude of the nonlinear distortion and its share of the soup error.** Top: relative nonlinear distortion $\|\Delta_{\mathrm{NL}}\|_{L^2}/\|u^\star\|_{L^2}$ against pool size, median with interquartile range over the 36 configurations of each regime and the five averaging methods. Bottom: the same pools measured as a fraction of the soup's own error, $\|\Delta_{\mathrm{NL}}\|_{L^2}/\|u(\cdot\,;\bar{\boldsymbol{\theta}}) - u^\star\|_{L^2}$, on a linear axis; a value of 1 means the distortion accounts for all of it. On linear operators the fraction stays above 0.99 in every regime and at every k, consistent with the superposition argument of equation 1; that series is dashed because it would otherwise be hidden under the nonlinear one. The sweep starts at $k = 2$, where the soup first differs from the ensemble.

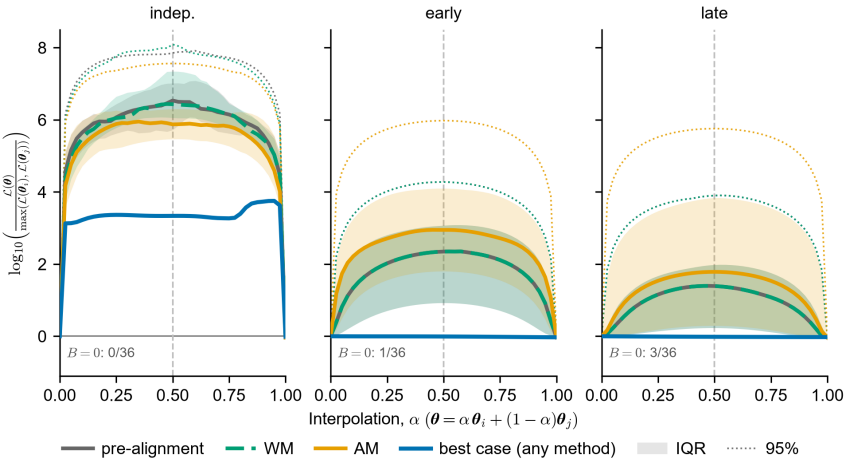


Figure 4: **Almost no pair of ingredients is linearly mode connected.** Score along the segment $[\boldsymbol{\theta}_i, \boldsymbol{\theta}_j]$ for the two top-ranked ingredients of each configuration, normalised by the score of the *worse* endpoint, median with interquartile range and 95% envelope over the 36 configurations of each training regime. The two matching series are Sign Re-Basin, which attains the lower barriers of the two families; Git Re-Basin lies within about 0.3 orders of magnitude of it and is omitted for legibility, but both enter the best case, which is the single most connected pair of each regime, the one whose path peaks lowest over every configuration and method. A value of 0, marked by the grey rule, is the threshold of linear mode connectivity, and the count in each panel gives how many configurations attain it with at least one of the five methods. The dashed vertical marks $\alpha = 1/2$, where the maximum sits, or nearly so, in the median of every regime and every method, which corresponds to the uniform two-ingredient soup. Under `independent`, the best case is a pair of collapsed soft-constrained Allen–Cahn models.

4 Related Work

PINNs and their pathologies. PINNs (Raissi et al., 2019) solve PDEs by encoding the equation in the loss. A recurring theme is that they fail in structured, well-characterised ways: spectral bias makes them learn low frequencies first (Khodakarami et al., 2026), competing loss terms produce stiff, conflicting gradients (Wang et al., 2021a; 2022) and the loss landscape itself obstructs optimisation (Krishnapriyan et al., 2021; Rathore et al., 2024). A recent work reframes these failure modes as provoked by arithmetic precision and places the parameters they stall at in the same basin as the accurate solution (Xu et al., 2025).

Model soups, weight averaging and ensembles. Averaging a pool of fine-tuned models into one, a model soup, matches ensemble accuracy at single-model cost (Wortsman et al., 2022); it is related to but distinct from SWA along a single training trajectory (Izmailov et al., 2018) and from combining independently trained models in output space (Krogh & Vedelsby, 1994; Lakshminarayanan et al., 2017). In PINNs, ensembles have been used to stabilise training, to quantify uncertainty, to assimilate data and to expose solution multiplicity (Haitsiukevich & Ilin, 2023; Davi & Braga-Neto, 2022; Fang et al., 2024; Zou et al., 2025; Pilar et al., 2025; Lu et al., 2025). To our knowledge, weight averaging has only appeared as averaging along a single sequential run (Guan et al., 2025). None performs post-hoc parameter-space averaging of independently trained PINNs.

Table 1: **What the alignment actually does.** The recovered re-basin element against the barrier-based proxy for it. Identity (exact) is the share of elements that fix every unit and flip no sign, over the 504 transformations of each cell (36 configurations $\times$ 2 ranking criteria $\times$ 7 non-anchor seeds); Identity (barrier proxy) is the share of the 36 anchor pairs, one per configuration, whose barrier the alignment left untouched, which is evidence of the identity and not proof of it. The fraction of hidden units the permutation leaves in place (signs ignored) is Mdn. units fixed: 1 is the identity permutation, and $1/\text{width}$ is what a uniformly random permutation fixes.

		Identity (exact)		Identity (barrier proxy)		Mdn. units fixed	
Regime	Matching	Git	Sign	Git	Sign	Git	Sign
`independent`	`weight`	0.0%	0.0%	0.0%	0.0%	0.053	0.031
	`activation`	0.0%	0.0%	0.0%	0.0%	0.007	0.007
`early`	`weight`	99.2%	99.2%	100.0%	100.0%	1.000	1.000
	`activation`	30.8%	22.4%	33.3%	30.6%	0.974	0.941
`late`	`weight`	99.6%	99.6%	100.0%	100.0%	1.000	1.000
	`activation`	43.5%	38.3%	50.0%	50.0%	0.992	0.979

Table 2: **Sign Re-Basin against Git Re-Basin, pair by pair.** Same configuration, same two ingredients and same matching criterion, differing only in whether the odd symmetry of $\tanh$ is quotiented. Mdn. sign flips is the fraction of hidden units Sign Re-Basin inverts, over the recovered transformations of each cell; the remaining columns are per anchor pair. Sign wins counts the pairs with $\log_{10}(B_{\text{git}}/B_{\text{sign}}) > 0$ and Mdn. gain is the median of that quantity, in decades; Ties are the pairs both methods return the same barrier for. Excluded are the pairs already connected under either method ($B = 0$), where the ratio is undefined, out of the 36 of each regime.

		Per transform	Per anchor pair				
Regime	Matching	Mdn. sign flips	Pairs	Sign wins	Mdn. gain ($\log_{10}$)	Ties	Excl.
`indep.`	`weight`	0.496	36	75.0%	0.142	0	0
	`activation`	0.499	36	63.9%	0.058	0	0
`early`	`weight`	0.000	35	0.0%	0.000	35	1
	`activation`	0.026	35	20.0%	0.000	11	1
`late`	`weight`	0.000	33	0.0%	0.000	33	3
	`activation`	0.008	33	12.1%	0.000	15	3

Linear mode connectivity. Soups average models fine-tuned from a common pretrained initialisation (Wortsman et al., 2022), and the usual account of why this works appeals to linear mode connectivity: models that share enough of their training lie in a region where the loss stays low along the segment joining them, so that their average remains a good model (Frankle et al., 2020). Outside that regime the premise is fragile. Independently trained models exhibit barriers that are conjectured to vanish once permutation symmetries are accounted for (Entezari et al., 2022). The evidence supports only the weak form, with permutations found pairwise, and a single alignment shared across a set has so far been shown only between iteratively trained or pruned networks (Sharma et al., 2024). Even a shared initialisation is not sufficient: fine-tuning from a common pretrained checkpoint can still yield models in separate, disconnected basins (Juneja et al., 2023). Our independent regime trains from independent initialisations, deliberately violating the premise, while our branching regimes are intended to restore progressively more of it.

Permutation and sign symmetries. Because permutations of hidden units leave a network's function unchanged, independently trained models can be aligned into a shared basin before averaging: Git Re-Basin (Ainsworth et al., 2023) does so by permutation matching. Permutation is not the only symmetry: most practical activations admit scaling symmetries as well, which for odd activations such as $\tanh$, a standard choice in PINNs, reduce to a sign symmetry (Kalogeropoulos et al., 2024). We extend Git Re-Basin to account for it with Sign Re-Basin, detailed in Appendix C.

DeepXDE and PLeaS Merging. For the benchmark, the selected PDE suite is implemented on top of DeepXDE (Lu et al., 2021b), a well-established library for Physics-Informed Machine Learning. In turn, the *Weight Matching* and *Activation Matching* suite of algorithms originally described in Git Re-Basin (Ainsworth et al., 2023), together with their Sign Re-Basin extension, is implemented on top of the routines provided by PLeaS Merging (Nasery et al., 2024).

FP64 training. Xu et al. (2025) revisit the canonical failure modes of PINNs and argue that they are not the signature of an adversarial loss landscape but of insufficient arithmetic precision: under FP32 the L-BFGS convergence test is satisfied prematurely, freezing the network in the characteristic low-loss, high-error phase that FP64 training simply passes through. Their setting focuses exclusively on pure L-BFGS training, while ours follows an Adam-then-L-BFGS schedule. Their findings are, however, relevant to our methodology: if each ingredient were silently stalled at the precision floor of its own optimiser, the loss barriers and error gaps we report would be measuring round-off rather than the geometry of the parameter space. We therefore train, align, average and evaluate every model in double precision throughout.

5 Discussion and Future Work

Every account of why weight averaging works requires the ingredients to lie in a region where the loss stays low along the segment joining them (Frankle et al., 2020; Wortsman et al., 2022), which is the core premise of the method. Our pools of PINNs almost never satisfy it, and the cause does not appear to be the quality of the individual ingredients. We conjecture that the multi-objective residual loss makes the set of parameters that satisfy the physics narrow and non-convex, so that a soup leaves this set at the first average; Appendix D.1 is consistent with this for soft-constrained problems, where averaging inflates the boundary residual the most. Hard constraints illustrate this, since they remove from the objective precisely the terms averaging is worst for, the boundary and initial residuals, which vanish by construction for every $\boldsymbol{\theta}$ and hence for every soup, and the failure persists: what breaks is the differential residual, through a map $\boldsymbol{\theta} \mapsto u(\cdot\,; \boldsymbol{\theta})$ whose nonlinearity does not commute with the average. The premise cannot simply be imported, either: with no pretrained checkpoint to fine-tune from, the shared training history required by a soup must be created during training rather than inherited, which is what the branching regimes are built to do at a fixed budget, as justified in Appendix B.

Alignment is the standard remedy for a barrier between independently trained networks (Entezari et al., 2022; Ainsworth et al., 2023), and the sign symmetry of Section 2.1 is one that permutation matching leaves untouched: two hidden units encoding the same feature of the solution with opposite sign do not reinforce when averaged but cancel, so the interference is destructive exactly on the units that carry information about $u^\star$, and a criterion searching only S_d cannot notice. The re-

covered transformations of Table 2 confirm that this effect occurs in practice, although correcting it lowers the barrier by less than an order of magnitude. Since permutations and sign flips exhaust the symmetries of an irreducible single-hidden-layer $\tanh$ network (Sussmann, 1992), Sign Re-Basin searches the largest symmetry group available for $\tanh$. This allows us to distinguish between two explanations: the ingredients are misaligned, or they occupy different basins. Our results support the second: after alignment, a barrier remains in all but four of the pairs, which suggests that the obstruction is geometric rather than an artefact of labelling.

Xu et al. (2025) place the canonical failure modes of PINNs on a broad flat plain at the bottom of the same basin as the accurate solution, a claim about the very geometry a soup would need. Our results do not contradict that diagnosis, established along a single trajectory and removed from our own account by working in double precision throughout, but they limit its scope: the midpoints we measure are not low-loss high-error stalls but high in both. If the accurate solutions of a PDE do share a basin, it is one whose low-loss set is not reached along straight lines, so the same-basin hypothesis and the absence of linear mode connectivity are compatible, and it is the latter that governs merging. The ensemble, for its part, behaves as the ambiguity decomposition (Krogh & Vedelsby, 1994) would predict, gaining where the pool is diverse and forfeiting that gain where the ingredients share a trajectory; our controls quantify its cost, paid at every query and in the single continuous differentiable representation that distinguishes a PINN from a mesh-based solver.

Limitations. The sweep covers three widths at a fixed depth, which is not a range from which the asymptotic behaviour of the barrier can be read: the permutation conjecture is stated in the limit (Entezari et al., 2022), and a measurement between widths 64 and 256 cannot speak to it. This limitation is partly computational, at 864 networks trained in double precision, but it is also deliberate, since the question we ask is whether soups are useful in the regime PINNs are actually used in for the forward problem, where their appeal is an accurate, continuous and differentiable surrogate obtained from a network that is not large; a width at which averaging became safe would already lie outside the setting that motivates the technique. For the same reason, the depth of the networks is held fixed at four hidden layers. The loss weights are held at $\lambda_\mu = 1$, which keeps the physics score unbiased but leaves untested whether an adaptively reweighted objective (Wang et al., 2021a; 2022), which modifies the loss landscape, would change the verdict. The result of Sussmann (1992) is proved for one hidden layer and we extrapolate it to our four, so for deeper networks Sign Re-Basin is not guaranteed to exhaust the symmetry group. Likewise, weight and activation matching are heuristics that align layer by layer, so a barrier that survives them bounds from above, but does not determine, the barrier under the best possible alignment. Finally, the barrier is measured only on the top-ranked pair of each configuration, not over all $\binom{8}{2} = 28$ pairs of the pool.

Future work. In the inverse problem, the governing parameters $\boldsymbol{q}$ of the equation are trained jointly with the network, but they enter the loss through the derivatives of $u(\cdot\,;\boldsymbol{\theta})$, so each ingredient's estimate is co-adapted to its own network, and a soup pairs the averaged $\boldsymbol{q}$ with $u(\cdot\,;\bar{\boldsymbol{\theta}})$, a function that, by the distortion measured here, is neither any ingredient nor their average. We therefore expect the soup's residual to inflate there as in the forward problem, and refitting $\boldsymbol{q}$ on the soup network to recover the true parameters less accurately than the best ingredient. This would not hold if the observed data, which anchor $u(\cdot\,;\boldsymbol{\theta})$ inside the domain, confine the ingredients to a region where the loss stays low along the segment joining them.

Beyond PINNs. Sign Re-Basin applies wherever the activation function is odd, a common choice in SciML. For instance, $\tanh$ in PINNs and DeepONets (Lu et al., 2021a) or $\sin$ in SIRENs (Sitzmann et al., 2020). It costs the same $O(d^3)$ per layer as permutation matching and quotients a symmetry that merging pipelines built on weight or activation matching do not see. Two settings would test it where the premise of a soup is more plausible than here: one-shot federated averaging of physics-informed models, which merges networks trained apart by design, and fine-tuned pretrained neural operators, which inherit the shared history a soup requires. Whether the sign component closes there the barrier left by permutation matching is an open question.

AI USE STATEMENT

In this work, we used generative AI tools (Claude, Anthropic) to polish and translate our writing, find related work, refine hypotheses, get feedback on experiments and implement methods. We did not use them to draft sections of the paper, generate datasets, prove mathematical claims or interpret results. The remaining disclosure categories do not apply to this work. We also used them to improve figures, write code and its documentation, and brainstorm titles. All AI-assisted output was reviewed by the authors; in particular, AI-generated code was reviewed with the same rigour we apply to our own code and validated through tests, comparison against known solutions, logging and use of a debugger. The software architecture, including the repository structure, class hierarchy and design decisions, was designed entirely by the authors; AI assistance was limited to the implementation within this human-defined design. We take full responsibility for the final content of this work, including all text, claims, code and figures produced with the aid of generative AI.

ETHICS STATEMENT

This work studies weight averaging for physics-informed neural networks on synthetic PDE benchmarks; it involves no human subjects, personal data or foreseeable harmful applications, and complies with the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, all the source code used for our experiments, along with the complete generated results, logs and evaluation scripts, is provided in the accompanying supplementary material. Detailed instructions on dependencies, environment setup and steps to reproduce the exact figures and tables presented in this paper are documented within the repository. All the models were trained using two NVIDIA RTX 3090 GPUs for approximately 19 days.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=CQsmMYmlP5T.

Anders Björner and Francesco Brenti. *Combinatorics of Coxeter Groups*. Number 231 in Graduate Texts in Mathematics. Springer, New York, 2005. doi: 10.1007/3-540-27596-7.

Caio Davi and Ulisses Braga-Neto. PSO-PINN: Physics-informed neural networks trained with particle swarm optimization. *arXiv preprint arXiv:2202.01943*, 2022.

Suchuan Dong and Naxian Ni. A method for representing periodic functions and enforcing exactly periodic boundary conditions with deep neural networks. *Journal of Computational Physics*, 435: 110242, 2021.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=dNigytemkL.

Zhiwei Fang, Sifan Wang, and Paris Perdikaris. Ensemble learning for physics informed neural networks: a gradient boosting approach, 2024. URL https://arxiv.org/abs/2302.13143.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3259–3269. PMLR, 2020. URL http://proceedings.mlr.press/v119/frankle20a.html.

Haotian Guan, Jinping Dong, and Wei-Ning Lee. A novel training framework for physics-informed neural networks: Towards real-time applications in ultrafast ultrasound blood flow imaging, 2025. URL https://arxiv.org/abs/2309.04755.

Katsiaryna Haitsiukevich and Alexander Ilin. Improved training of physics-informed neural networks with model ensembles. In *2023 International Joint Conference on Neural Networks*, pp. 1–8, 2023.

Pavel Izmailov, Dmitrii Podoprikhin, T. Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence*, 2018.

Ameya D. Jagtap and George Em Karniadakis. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.

Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. Linear connectivity reveals generalization strategies. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=hY6M0JHl3uL.

Ioannis Kalogeropoulos, Giorgos Bouritsas, and Yannis Panagakis. Scale equivariant graph metanetworks. In *Advances in Neural Information Processing Systems*, volume 37, pp. 106800–106840, 2024. URL https://openreview.net/forum?id=8Fxqn1tZM1.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3:422–440, 2021.

Siavash Khodakarami, Vivek Oommen, Nazanin Ahmadi Daryakenari, Maxim Beekenkamp, and George Em Karniadakis. Spectral bias in physics-informed and operator learning: Analysis and mitigation guidelines, 2026. URL https://arxiv.org/abs/2602.19265.

Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL https://proceedings.neurips.cc/paper_files/paper/1994/file/b8c37e33defde51cf91e1e03e51657da-Paper.pdf.

Isaac E. Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

Binghang Lu, Changhong Mou, and Guang Lin. iPINNER: An iterative physics-informed neural network with ensemble kalman filter. *Journal of Computational Physics*, 548:114592, 2025.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021a. doi: 10.1038/s42256-021-00302-5.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021b. doi: 10.1137/19M1274067.

Rambod Mojgani, Maciej Balajewicz, and Pedram Hassanzadeh. Kolmogorov n-width and lagrangian physics-informed neural networks: A causality-conforming manifold for convection-dominated PDEs. *Computer Methods in Applied Mechanics and Engineering*, 404:115810, 2023.

Anshul Nasery, Jonathan Hayase, Pang Wei Koh, and Sewoong Oh. Pleas — merging models with permutations and least squares. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30493–30502, 2024. URL https://api.semanticscholar.org/CorpusID:270878221.

Philipp Pilar, Markus Heinonen, and Niklas Wahlström. Repulsive ensembles for bayesian inference in physics-informed neural networks, 2025. URL https://arxiv.org/abs/2505.17308.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. 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, 2019.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training PINNs: A loss landscape perspective. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Franz M. Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C. Geiger. On the role of fixed points of dynamical systems in training physics-informed neural networks. *Trans. Mach. Learn. Res.*, 2023, 2022. URL https://api.semanticscholar.org/CorpusID:253065215.

Ekansh Sharma, Devin Kwok, Tom Denton, Daniel M. Roy, David Rolnick, and Gintare Karolina Dziugaite. Simultaneous linear connectivity of neural networks modulo permutation. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, volume 14947 of *Lecture Notes in Artificial Intelligence*, pp. 262–279. Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-70368-3_16.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7462–7473, 2020.

N. Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

Héctor J. Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks*, 5(4):589–593, 1992.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a. doi: 10.1137/20M1318043.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021b.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 23965–23998, 2022.

Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.115671. URL https://www.sciencedirect.com/science/article/pii/S0045782522006260.

Chenhui Xu, Dancheng Liu, Amir Nassereldine, and Jinjun Xiong. FP64 is all you need: Rethinking failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2aotKzkOCm.

Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

Zongren Zou, Zhicheng Wang, and George Em Karniadakis. Learning and discovering multiple solutions using physics-informed neural networks with random initialization and deep ensemble. *Proceedings of the Royal Society A*, 481(2325):20250205, 2025. doi: 10.1098/rspa.2025.0205.

A BACKGROUND

A.1 PHYSICS-INFORMED NEURAL NETWORKS

The original idea behind Physics-Informed Neural Networks (PINNs) is to use a neural network (usually, a multilayer perceptron) to solve a Cauchy problem by encoding the partial differential equation (PDE), the boundary conditions and the initial conditions (if any) inside the loss function (Raissi et al., 2019). From now on, we assume an evolutionary differential equation (the stationary case is analogous).

Let $\Omega \subset \mathbb{R}^n$ be a bounded spatial domain and $T > 0$ a final time. We seek $u : [0,T] \times \Omega \to \mathbb{R}$ satisfying:[1]

$$\begin{aligned} \mathcal{F}[u](t,x) &= 0, \quad (t,x) \in (0,T] \times \Omega, \\ \mathcal{B}[u](t,x) &= 0, \quad (t,x) \in [0,T] \times \partial\Omega, \\ u(0,x) &= u_0(x), \quad x \in \Omega. \end{aligned} \tag{5}$$

With this notation, $\mathcal{F}$ represents the differential operator associated with the PDE (the physics), $\mathcal{B}$ the boundary conditions and u_0 the initial condition. We say that u is a solution of the Cauchy problem described by equation 5.

Definition 1 (Raissi et al. 2019)**.** *A* soft-constrained PINN $u(\cdot\,;\boldsymbol{\theta})\colon [0,T] \times \Omega \to \mathbb{R}$, $\boldsymbol{\theta} \in \mathbb{R}^D$, *is a neural network that encodes the structure of equation 5 into the weighted residual loss*

$$\mathcal{L}(\boldsymbol{\theta}) = \lambda_{\mathrm{PDE}}\mathcal{L}_{\mathrm{PDE}}(\boldsymbol{\theta}) + \lambda_{\mathrm{BC}}\mathcal{L}_{\mathrm{BC}}(\boldsymbol{\theta}) + \lambda_{\mathrm{IC}}\mathcal{L}_{\mathrm{IC}}(\boldsymbol{\theta}), \quad \lambda_\mu > 0, \tag{6}$$

whose terms are the mean squared residuals of the PDE, the boundary conditions and the initial condition over the collocation, boundary and initial points $\mathcal{X}_{\mathrm{PDE}}, \mathcal{X}_{\mathrm{BC}}, \mathcal{X}_{\mathrm{IC}}$:

$$\begin{aligned} \mathcal{L}_{\mathrm{PDE}}(\boldsymbol{\theta}) &= \frac{1}{|\mathcal{X}_{\mathrm{PDE}}|} \sum\nolimits_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{PDE}}} \left|\mathcal{F}[u(\cdot\,;\boldsymbol{\theta})](\boldsymbol{x})\right|^2, \\ \mathcal{L}_{\mathrm{BC}}(\boldsymbol{\theta}) &= \frac{1}{|\mathcal{X}_{\mathrm{BC}}|} \sum\nolimits_{\boldsymbol{x} \in \mathcal{X}_{\mathrm{BC}}} \left|\mathcal{B}[u(\cdot\,;\boldsymbol{\theta})](\boldsymbol{x})\right|^2, \\ \mathcal{L}_{\mathrm{IC}}(\boldsymbol{\theta}) &= \frac{1}{|\mathcal{X}_{\mathrm{IC}}|} \sum\nolimits_{x \in \mathcal{X}_{\mathrm{IC}}} \left|u(0,x;\boldsymbol{\theta}) - u_0(x)\right|^2, \end{aligned} \tag{7}$$

where $\boldsymbol{x} = (t,x) \in [0,T] \times \Omega$ denotes the network input.

Unlike classical numerical solvers (finite differences, finite elements or spectral methods), which discretise the domain on a mesh and return the solution only at grid nodes, a PINN yields a continuous, mesh-free and differentiable approximation $u(\cdot\,;\boldsymbol{\theta})$ defined on the whole domain $[0,T] \times \Omega$, which can be queried at any point without interpolation (Karniadakis et al., 2021).

The soft-constrained (SC) network of Definition 1 (often called a *vanilla* PINN) is just the original instance of the framework; numerous architectures have since been built on this idea, from gradient-enhanced and extended PINNs to Lagrangian formulations (Yu et al., 2022; Jagtap & Karniadakis, 2020; Mojgani et al., 2023), among many others. In particular, the *hard-constraint* (HC) approach has gained popularity, enforcing the boundary and initial data architecturally rather than through the loss. This idea can be traced back to Lagaris et al. (1998), well before the term PINN was coined, and has since been extended by more modern constructions such as Sukumar & Srivastava (2022). We adopt the following operative definition.

Definition 2 (Lagaris et al. 1998)**.** *A* hard-constrained PINN $u(\cdot\,;\boldsymbol{\theta})\colon [0,T] \times \Omega \to \mathbb{R}$, $\boldsymbol{\theta} \in \mathbb{R}^D$, *satisfies the boundary and initial conditions of equation 5* by construction*, through the ansatz*

$$u(t,x;\boldsymbol{\theta}) = G(t,x) + A(t,x)N(t,x;\boldsymbol{\theta}),$$

where $N(\cdot\,;\boldsymbol{\theta})$ is the trainable network and G, A are fixed, problem-dependent functions: G matches the boundary and initial data, while A vanishes wherever those data are imposed. Consequently, the boundary and initial conditions of equation 5 are satisfied trivially for every $\boldsymbol{\theta}$, and the composite loss from equation 6 reduces to $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\mathrm{PDE}}(\boldsymbol{\theta})$.

[1] We adopt the scalar-output convention purely for notational simplicity. Every statement in this section extends componentwise to vector-valued solutions $u : [0,T] \times \Omega \to \mathbb{R}^m$ and systems of PDEs.

Hard-constraint enforcement circumvents the well-known gradient-instability pathology of soft-constrained PINNs, where the shared parameters $\boldsymbol{\theta}$ are pulled in conflicting directions by the three competing contributions $\nabla_{\boldsymbol{\theta}}\mathcal{L}_\mu$ (Wang et al., 2021a; 2022), weighted by λ_μ, $\mu = \mathrm{PDE}, \mathrm{BC}, \mathrm{IC}$. Since the loss collapses to $\mathcal{L} \equiv \mathcal{L}_{\mathrm{PDE}}$, no such competition remains.

The same philosophy (enforcing structure by construction rather than through a penalty term) can be applied at the input rather than at the output. Composing the network with a truncated Fourier basis in the periodic coordinate renders $u(\cdot\,;\boldsymbol{\theta})$ exactly periodic in x for every $\boldsymbol{\theta}$, together with all its derivatives (Dong & Ni, 2021). Such embeddings also mitigate the *spectral bias* of neural networks, i.e. their tendency to learn low-frequency components first, which is especially detrimental for PINNs approximating multi-scale or high-frequency solutions (Khodakarami et al., 2026; Wang et al., 2021b).

A.2 Weight Averaging and Output Averaging

Given a pool of trained PINNs, we can collapse it into a single predictor in one of two ways: by averaging their weights or by averaging their outputs. Both are standard techniques from vision and language, and since a PINN is an ordinary feed-forward network trained by gradient descent, they apply without modification, regardless of the loss. We recall them in the notation of Definition 1. Moreover, they can be directly compared against a reference solution (either analytic or computed via numerical methods).

Throughout, a pool consists of $k \geq 2$ PINNs that share a common architecture $u_{(\cdot)}$ and differ only in their trained weights $\{\boldsymbol{\theta}_i\}_{i=1}^k$. We call each such checkpoint an *ingredient*.

Model soups. A *model soup* (Wortsman et al., 2022) averages the ingredients in *parameter* space, producing a single network whose weights are their convex combination,

$$\bar{\boldsymbol{\theta}} = \sum_{i=1}^{k} \beta_i\, \boldsymbol{\theta}_i, \qquad \sum_{i=1}^{k} \beta_i = 1, \quad \beta_i \geq 0, \tag{8}$$

yielding the surrogate $u(\cdot\,;\bar{\boldsymbol{\theta}})$. Following the culinary nomenclature, we distinguish two main methods or *recipes* for building the soup. The *uniform* soup takes $\beta_i = 1/k$ in equation 8. The *greedy* recipe of Wortsman et al. (2022) instead sorts the ingredients by decreasing individual validation performance and appends the next one to the average only if it improves a held-out metric. The appeal for PINNs is that, whatever the recipe, the result is one continuous and differentiable surrogate: it keeps the $O(1)$ query cost and the single deployable model of a PINN, unlike the $O(k)$ cost of evaluating a pool.

Output ensembles. Averaging the outputs of independently trained networks is a long-standing and effective technique, most familiar in classification, where the averaged softmax probabilities of a *deep ensemble* sharpen accuracy and calibration. For regression, the setting of PINNs, output averaging is used chiefly for variance reduction (Krogh & Vedelsby, 1994). An *output ensemble* averages the same ingredients in *function* space,

$$\bar{u}(\boldsymbol{x}) = \sum_{i=1}^{k} \beta_i\, u(\boldsymbol{x};\boldsymbol{\theta}_i), \qquad \sum_{i=1}^{k} \beta_i = 1, \quad \beta_i \geq 0, \tag{9}$$

returning a value rather than a network, at an $O(k)$ query cost. Because differentiation is linear, for a linear PDE, i.e. if its differential operator $\mathcal{F}$ is linear, the ensemble residual is the average of the ingredients' residuals (see equation 1). Thus, if every ingredient solves the PDE, the ensemble solves it too, exactly, by superposition. Throughout, the ensemble therefore serves as the reference against which weight averaging is contrasted: it combines the very same ingredients, but in function rather than parameter space. In order to achieve a fair comparison, we mirror the uniform and greedy recipes for constructing the ensembles. In the next section, we introduce the evaluation metrics we employ for this purpose.

B EXPERIMENTAL DESIGN

Our study is a controlled full factorial design over the PDE, its constraint formulation, the architecture and the training regime. We train both soft- and hard-constrained PINNs to solve six different Cauchy problems. We cross this setting with three architectures and three training regimes, giving 108 configurations, each trained eight times ($k = 8$ ingredients), resulting in 864 networks in total. Afterwards, every configuration is evaluated with the same five parameter-space averaging methods against the same two references: the output ensemble and the best single ingredient.

B.1 PINN SETTING

PDE suite. We use six equations, three with a linear operator and three nonlinear, listed in Table 3. This distinction is essential: for a linear operator the output ensemble is itself an exact solution by superposition (see equation 1), so a soup failure on a linear PDE cannot be blamed on the equation. Each equation is solved with both soft- and hard-constrained PINNs (Definitions 1 and 2). Their explicit forms and the provenance of the reference solutions are given below.

Allen–Cahn equation. This nonlinear reaction-diffusion PDE was originally introduced to model phase separation in multi-component alloys. We solve the following problem:[2]

$$\begin{aligned} u_t - 10^{-4}\, u_{xx} + 5u^3 - 5u = 0, \quad & (t,x) \in (0,1] \times (-1,1), \\ u(t,-1) - u(t,1) = 0, \quad & t \in [0,1], \\ u_x(t,-1) - u_x(t,1) = 0, \quad & t \in [0,1], \\ u(0,x) = x^2 \cos(\pi x), \quad & x \in (-1,1). \end{aligned} \tag{10}$$

Although it is one of the original benchmarks used to introduce PINNs (Raissi et al., 2019), its stiff dynamics make it a well-documented failure mode: soft-constrained PINNs are known to collapse onto the unstable fixed point $u \equiv 0$ instead of the correct phase-separated solution (Rohrhofer et al., 2022), and it is routinely used to stress-test adaptive-sampling and residual-based remedies (Wu et al., 2023). Its hard-constrained ansatz imposes $u(t, \pm 1) = -1$ in place of the periodic conditions.

Burgers' equation. Burgers' equation is a nonlinear PDE combining a convective term ($u\, u_x$) with a diffusive term ($\nu\, u_{xx}$); it is the simplest model capturing the convection–diffusion interplay found in fluid dynamics, often regarded as a one-dimensional analogue of Navier–Stokes.

We study the following problem:[3]

$$\begin{aligned} u_t + u\, u_x - \frac{0.01}{\pi}\, u_{xx} = 0, \quad & (t,x) \in (0, 0.99] \times (-1,1), \\ u(t,-1) = u(t,1) = 0, \quad & t \in [0, 0.99], \\ u(0,x) = -\sin(\pi x), \quad & x \in (-1,1). \end{aligned} \tag{11}$$

With $\nu = 0.01/\pi$ the solution develops a sharp, near-discontinuous shock layer, yet it remains one of the most widely adopted PINN benchmarks since the original formulation (Raissi et al., 2019; Lu et al., 2021b): standard soft-constrained PINNs generally solve it well, although the steep gradient region is a common target for residual-based adaptive sampling schemes (Wu et al., 2023).

Convection equation. The linear convection (advection) equation is the simplest transport PDE, describing the rigid translation of an initial profile at a constant speed c, set here to $c = 1$, with no diffusive term:

$$\begin{aligned} u_t + u_x = 0, \quad & (t,x) \in (0,1] \times (0, 2\pi), \\ u(t,0) - u(t,2\pi) = 0, \quad & t \in [0,1], \\ u_x(t,0) - u_x(t,2\pi) = 0, \quad & t \in [0,1], \\ u(0,x) = \sin(x) + 0.5\sin(4x) + 0.25\sin(8x), \quad & x \in (0, 2\pi). \end{aligned} \tag{12}$$

[2] This problem has no closed-form solution; the reference solution used to evaluate model accuracy was obtained from the official code release accompanying Raissi et al. (2019) (`AC.mat`).

[3] This problem has no closed-form solution; the reference solution used for evaluation is taken from the example dataset distributed with Lu et al. (2021b) (`Burgers.npz`).

This equation admits a closed-form solution, given by the transport of the initial profile, $u(0, x)$, at speed $c = 1$:

$$u(t, x) = \sin(x - t) + 0.5 \sin\big(4(x - t)\big) + 0.25 \sin\big(8(x - t)\big).$$

Despite its linearity, it is one of the canonical PINN failure modes: for sufficiently large convection speeds the soft-constrained loss landscape becomes increasingly non-convex, causing standard PINNs to fail to propagate information from the initial condition and instead collapse to a trivial solution (Krishnapriyan et al., 2021). It has also motivated alternative formulations, such as Lagrangian PINNs (Mojgani et al., 2023). Our $c = 1$ lies below that range.

Heat equation. The heat (diffusion) equation is the archetypal linear parabolic PDE, describing purely diffusive transport processes such as heat conduction along a rod. Its solution is smooth and monotonically decaying in time, with no sharp gradients or multiscale features, which makes it one of the best-behaved cases for PINNs and a standard, largely uncontroversial sanity-check benchmark throughout the PINN literature (Karniadakis et al., 2021). We solve the following initial-boundary value problem on the unit interval:

$$\begin{aligned} u_t - 0.4\, u_{xx} = 0, \quad & (t, x) \in (0, 1] \times (0, 1), \\ u(t, 0) = u(t, 1) = 0, \quad & t \in [0, 1], \\ u(0, x) = \sin(\pi x), \quad & x \in (0, 1). \end{aligned} \tag{13}$$

This equation admits a closed-form solution:

$$u(t, x) = e^{-0.4\pi^2 t} \sin(\pi x).$$

Helmholtz equation. We solve the modified Helmholtz (screened Poisson) equation, a linear, time-independent (elliptic) PDE that models steady diffusion with linear absorption, as in the screened potentials of electrostatics and plasma physics:

$$\begin{aligned} -\Delta u + 16\, u - 29 \sin(2x) \sin(3y) = 0, \quad & (x, y) \in \Omega = (0, 2\pi)^2, \\ u(0, y) - u(2\pi, y) = 0, \quad & y \in [0, 2\pi], \\ u_x(0, y) - u_x(2\pi, y) = 0, \quad & y \in [0, 2\pi], \\ u(x, 0) - u(x, 2\pi) = 0, \quad & x \in [0, 2\pi], \\ u_y(x, 0) - u_y(x, 2\pi) = 0, \quad & x \in [0, 2\pi]. \end{aligned} \tag{14}$$

This equation is constructed as a manufactured solution, so its closed-form solution is known by definition:

$$u(x, y) = \sin(2x) \sin(3y).$$

Because the manufactured solution here oscillates at moderately high spatial frequencies (frequencies 2 and 3 in x and y, respectively), it exercises the spectral bias of coordinate MLPs, whereby PINNs preferentially fit low-frequency components and struggle with higher-frequency ones (Wang et al., 2021b; Khodakarami et al., 2026); the periodic boundary conditions considered here are typically imposed exactly through Fourier feature embeddings rather than as soft loss terms (Dong & Ni, 2021).

Schrödinger equation. The nonlinear Schrödinger equation is a dispersive, cubically nonlinear PDE governing the envelope dynamics of wave packets in nonlinear media, with applications ranging from optical-fibre soliton propagation to Bose–Einstein condensates in quantum mechanics:[4]

$$\begin{aligned} i\, u_t + 0.5\, u_{xx} + |u|^2 u = 0, \quad & (t, x) \in (0, \pi/2] \times (-5, 5), \\ u(t, -5) - u(t, 5) = 0, \quad & t \in [0, \pi/2], \\ u_x(t, -5) - u_x(t, 5) = 0, \quad & t \in [0, \pi/2], \\ u(0, x) = 2\,\mathrm{sech}(x), \quad & x \in (-5, 5). \end{aligned} \tag{15}$$

Its complex-valued solution and periodic boundary conditions make it a standard, moderately challenging PINN benchmark since the original formulation (Raissi et al., 2019): unlike Allen–Cahn or high-speed convection, it is generally solved successfully by vanilla soft-constrained PINNs, provided the network architecture can represent both the real and imaginary output components accurately.

[4] This system has no closed-form solution; the reference solution used for evaluation was obtained from the dataset released with Raissi et al. (2019) (`NLS.mat`).

Architectures. All networks are $\tanh$ MLPs with Glorot-normal initialisation. This axis, listed in Table 4, sweeps architecture width at a fixed depth of four hidden layers: permutation alignment predicts the barrier between independently trained solutions to narrow with width (Entezari et al., 2022; Ainsworth et al., 2023), whereas deepening the network would lengthen the differentiation graph of the residual and thus change the conditioning of the optimisation problem, not only its capacity. Problems that require it use dedicated families: a single-output network for problems like Burgers or heat (family A), a dual-output network for Schrödinger (B), and Fourier-feature input embeddings for the periodic hard-constrained problems, which mitigate spectral bias on their high-frequency modes (C to E). Without these additions, the PINN approximations would be far inferior to those of the rest of the suite.

B.2 Training Setting

Training protocol. Every network is trained with the same fixed budget: $20{,}000$ iterations of Adam, followed by another $15{,}000$ of L-BFGS, choosing the best snapshot of the latter. This two-optimiser schedule can be thought of as an exploratory phase (Adam) followed by a fine-tuning or final convergence phase (L-BFGS), and it is a well-established practice for PINNs (Raissi et al., 2019; Rathore et al., 2024). PDE residual collocation starts from $5{,}000$ Latin-hypercube sampling (LHS) points and is refined by five rounds of greedy residual-based adaptive resampling (RAR-G, Wu et al. 2023), one every $4{,}000$ Adam iterations, each adding the $3{,}000$ highest-residual points out of $9{,}000$ candidates, for a total of $N_{\mathrm{PDE}} = 20{,}000$ interior points. In the case of soft-constrained networks, we add $N_{\mathrm{BC}} = 400$ boundary and $N_{\mathrm{IC}} = 400$ initial points, sampled with LHS. Unlike the PDE collocation set, these points remain fixed during training. We use loss weights $\lambda_\mu = 1$, $\mu = \mathrm{PDE}, \mathrm{BC}, \mathrm{IC}$, for every configuration (the DeepXDE default). The models are trained in 64-bit floating-point precision to ensure that the conclusions on soups are not driven by arithmetic error. The protocol is held identical across all configurations.

Training regimes. The central axis is how much training the k ingredients share, at a fixed budget of iterations and training points (Table 5). This axis is key in judging when weight alignment procedures (Ainsworth et al., 2023) are useful when applied to PINNs. In the *independent* regime the eight ingredients are trained from independent seeds and learning rates, with no training in common (Git Re-Basin framework). We follow a different approach in the two branching regimes: ingredients are children fine-tuned from a common parent checkpoint, which mimics the setting model soups were designed for (Wortsman et al., 2022). The *early* regime forks at iteration $10{,}000$ in the Adam phase, while the *late* regime forks at iteration $15{,}000$. Each child then completes its own Adam and L-BFGS with its own seed and learning rate, inheriting only the RAR points up to its forking instant. This branching procedure was chosen over reinitialising the last hidden layer for further fine-tuning (procedure known as *transfer learning* in disciplines such as computer vision) because it was considered more appropriate for, and transferable to, the training protocol described above.

Ingredient quality. Table 7 presents the quality of the ingredients trained for this research, through the aggregate median of the relative L^2 error across the three training regimes. We should note that soft-constrained Allen–Cahn ingredients reach errors of order 10^{-1} because they collapse to the trivial solution $u \equiv 0$, a documented failure mode of vanilla PINNs (Rohrhofer et al., 2022); Schrödinger reaches that median order only through its spread across hyperparameters. Despite the double precision, our Adam-then-L-BFGS optimisation strategy did not reproduce the results of Xu et al. (2025) for Allen–Cahn in particular, where a full L-BFGS approach is taken instead.

B.3 Soup Setting

Sources of ingredient diversity. In model soups, the ingredients are obtained through a hyperparameter sweep over the optimiser, data augmentation and learning rate (Wortsman et al., 2022). Due to the differences between their original setting and ours, we adapt these three axes as follows:

(i) *Optimiser:* not varied. Adam-then-L-BFGS is the standard PINN optimiser, and changing it would confound optimisation quality with the merging question. This reasoning also partially supports our decision to keep $\lambda_\mu = 1$ fixed, made primarily so that the score of equation 3 is an unbiased metric (see *Evaluation metrics* below).

(ii) *Data augmentation:* its mesh-free analogue is the collocation sampling itself. Each child inherits the collocation points of the parent up to its branch iteration and runs its own remaining RAR-G rounds under its seed, until completing the budget, so ingredients see different point clouds. The seed associated with each ingredient dictates the global RNG of the entire pipeline of its training (e.g. weight initialisation or LHS sampling).

(iii) *Learning rate:* we sweep the learning rate from a base 10^{-3} to a floor 10^{-4}. The upper end is a conventional Adam setting for PINNs (Wang et al., 2021a; 2022), and the range is deliberately narrow: the ill-conditioning of the PINN loss landscape penalises larger steps (Rathore et al., 2024), while an overly aggressive spread would risk driving the children out of the linearly connected region they may share with the parent (Frankle et al., 2020), on which soups rely. In the case of the branching training regimes, the parent is always trained with a base learning rate of 10^{-3}. The children then resume their training with their own step size, with the sweep described above.

Therefore, in each training regime, ingredient diversity emerges from the seed (i.e. the training points available) and learning rate (how much the PINN is allowed to explore the loss landscape each step of the training loop) the model is trained with. Note that, in the *independent* regime, the seed also affects the initialisation of the network.

Averaging methods and controls. We evaluate five parameter-space methods: the naive average and four symmetry-aligned variants, as listed in Table 6. In each configuration, we align against the best ingredient, allowing for a sweep in the number of soup members (ceiling of $k = 8$) and a study of whether the soup degrades with it. Alignment replaces the shared history that model soups obtain by fine-tuning from a pretrained checkpoint (Frankle et al., 2020; Wortsman et al., 2022). In our setting, the branching regimes provide that history, and alignment starts instead from ingredients trained apart. It moves each one towards the reference checkpoint by a function-preserving symmetry (Entezari et al., 2022; Ainsworth et al., 2023). We apply it in every regime, including those where a shared history should make it unnecessary.

Against every soup we place two references: the output ensemble of equation 9, exact on linear PDEs, and the best single ingredient, which is what one would keep in practice. In order to separate parameter-space from function-space averaging we report the *nonlinear distortion* Δ_{NL} of equation 2, both absolutely and relative to the L^2-norm of the reference $u^\star$. It vanishes for an affine parametrisation, so it measures the nonlinearity of $\boldsymbol{\theta} \mapsto u(\cdot\,; \boldsymbol{\theta})$.

Evaluation metrics. We score the quality of a predictor u (an ingredient, a soup or an ensemble) through the relative L^2 error against a reference solution $u^\star$,

$$e_{L^2}(u) = \frac{\|u - u^\star\|_{L^2}}{\|u^\star\|_{L^2}},$$

written $e_{L^2}(\boldsymbol{\theta})$ for a network with parameters $\boldsymbol{\theta}$. In practice, it is computed as its discrete analogue over the N_{eval} points $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N_{\mathrm{eval}}}$ of the grid on which the reference solution of each equation is given,

$$e_{L^2}(u) \approx \frac{\left(\sum_{i=1}^{N_{\mathrm{eval}}} |u(\boldsymbol{x}^{(i)}) - u^\star(\boldsymbol{x}^{(i)})|^2\right)^{1/2}}{\left(\sum_{i=1}^{N_{\mathrm{eval}}} |u^\star(\boldsymbol{x}^{(i)})|^2\right)^{1/2}}.$$

This requires the reference and is therefore the *oracle* for which the physics score of equation 3 is the reference-free proxy, defined for every predictor with a parameter vector.

Reporting scale. The metrics above play the role of the validation accuracy in the original model soups recipe (Wortsman et al., 2022). Unlike an accuracy, however, they vary by orders of magnitude from one problem to another. We therefore report every predictor relative to u_{best}, the ingredient of its configuration ranked first by the physics score, as the L^2 error requires the reference. The reported quantity is

$$y(u) = \log_{10}\left(\frac{f(u)}{f(u_{\mathrm{best}})}\right),$$

where f is either of the two metrics above, as a function of the predictor (equivalently, dependent on the parameters $\boldsymbol{\theta} \in \mathbb{R}^D$). Therefore, $y = 0$ ties the best ingredient, $y < 0$ improves it and $y > 0$ worsens it, in both cases by $|y|$ decades.

For barriers, the path along the segment is measured against the worse endpoint of the pair instead. The figures show either the whole path on this scale or only its peak, which becomes

$$\log_{10}\left(1 + \frac{B(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)}{\max\left(\mathcal{L}(\boldsymbol{\theta}_i), \mathcal{L}(\boldsymbol{\theta}_j)\right)}\right).$$

This quantity is positive exactly when the barrier B of equation 4 is, so linear mode connectivity still corresponds to zero. The same applies to B' of equation 25, with $e_{L^2}(\cdot)$ in place of the loss.

Table 3: **Equation suite.** An axis of six PDEs covering a range of physical behaviours and difficulty levels.

Cauchy problem	Equation ref.	Linear	Evolutionary	Note
Allen–Cahn	10	no	yes	stiff, cubic
Burgers	11	no	yes	sharp front
Convection	12	yes	yes	advection
Heat	13	yes	yes	diffusion
Helmholtz	14	yes	no	2D, oscillatory
Schrödinger	15	no	yes	complex, periodic

Table 4: **Network architectures.** Layer-width axis for the hidden layers of the MLPs. Each label combines a family letter with the width index. Family A is a single-output network (e.g. Burgers, heat), family B a dual-output network (Schrödinger), and families C to E add Fourier-feature input embeddings for the periodic hard-constrained problems (e.g. convection, Helmholtz).

Architecture	Depth	Width	Label
`narrow`	4	64	`A1–E1`
`standard`	4	128	`A2–E2`
`wide`	4	256	`A3–E3`

Table 5: **Training regimes.** An axis of shared training trajectories, at fixed total budget.

Training regime	Adam shared (%)	Fork (iteration)	N_{PDE} shared (%)
`independent`	0	–	0
`early`	50	10,000	55
`late`	75	15,000	70

Table 6: **Averaging methods.** The five parameter-space averaging methods and the output-space control, with the group searched during alignment and the criterion used to rank candidates. Both the uniform and the greedy recipe are built from every row.

Averaging method	Matching	Group	Ranking	Note
`naive`	–	–	score	no alignment
`git-rebasin`	activation, weight	S_d	score	Ainsworth et al. (2023)
`sign-rebasin`	activation, weight	$(\mathbb{Z}/2\mathbb{Z})^d \rtimes S_d$	score	this work
`ensemble`	–	–	L^2 rel. error	function-space control

Table 7: **Training results.** Ingredient quality as relative L^2 error per problem and architecture (median and interquartile range). †Soft-constrained Allen–Cahn collapses to the trivial solution $u \equiv 0$, a PINN failure mode our Adam-then-L-BFGS scheme does not avoid, even in FP64. ‡Soft-constrained Schrödinger satisfies the physics, but its error varies widely across hyperparameters.

Problem	Architecture	Label	Median e_{L^2} [IQR]
burgers	narrow	A1	9.90×10^{-3} [5.87×10^{-4}, 1.25×10^{-2}]
	standard	A2	6.79×10^{-3} [6.63×10^{-4}, 1.12×10^{-2}]
	wide	A3	1.77×10^{-3} [7.15×10^{-4}, 1.30×10^{-2}]
burgers_hc	narrow	A1	3.54×10^{-4} [3.01×10^{-4}, 4.26×10^{-4}]
	standard	A2	3.68×10^{-4} [3.30×10^{-4}, 4.68×10^{-4}]
	wide	A3	3.93×10^{-4} [3.46×10^{-4}, 4.29×10^{-4}]
heat	narrow	A1	7.45×10^{-4} [5.67×10^{-4}, 8.79×10^{-4}]
	standard	A2	8.85×10^{-4} [6.97×10^{-4}, 1.55×10^{-3}]
	wide	A3	9.27×10^{-4} [7.37×10^{-4}, 1.44×10^{-3}]
heat_hc	narrow	A1	6.20×10^{-5} [5.78×10^{-5}, 7.35×10^{-5}]
	standard	A2	3.67×10^{-5} [2.67×10^{-5}, 5.12×10^{-5}]
	wide	A3	3.28×10^{-5} [2.39×10^{-5}, 4.12×10^{-5}]
convection	narrow	A1	3.68×10^{-2} [3.07×10^{-2}, 4.39×10^{-2}]
	standard	A2	3.80×10^{-2} [2.21×10^{-2}, 5.20×10^{-2}]
	wide	A3	4.19×10^{-2} [1.80×10^{-2}, 6.21×10^{-2}]
convection_hc	narrow	D1	2.15×10^{-4} [1.97×10^{-4}, 2.66×10^{-4}]
	standard	D2	2.02×10^{-4} [1.92×10^{-4}, 2.38×10^{-4}]
	wide	D3	2.54×10^{-4} [2.39×10^{-4}, 3.09×10^{-4}]
allen_cahn†	narrow	A1	7.92×10^{-1} [5.97×10^{-1}, 9.95×10^{-1}]
	standard	A2	9.65×10^{-1} [6.54×10^{-1}, 9.97×10^{-1}]
	wide	A3	8.25×10^{-1} [5.52×10^{-1}, 9.95×10^{-1}]
allen_cahn_hc	narrow	A1	1.51×10^{-2} [1.44×10^{-2}, 1.61×10^{-2}]
	standard	A2	1.25×10^{-2} [1.12×10^{-2}, 1.38×10^{-2}]
	wide	A3	1.47×10^{-2} [1.10×10^{-2}, 1.63×10^{-2}]
helmholtz	narrow	A1	3.06×10^{-2} [3.40×10^{-4}, 3.35×10^{-2}]
	standard	A2	2.25×10^{-4} [2.10×10^{-4}, 3.02×10^{-4}]
	wide	A3	2.93×10^{-4} [2.26×10^{-4}, 3.43×10^{-4}]
helmholtz_hc	narrow	E1	1.59×10^{-5} [1.52×10^{-5}, 1.97×10^{-5}]
	standard	E2	1.58×10^{-5} [1.47×10^{-5}, 1.85×10^{-5}]
	wide	E3	1.83×10^{-5} [1.63×10^{-5}, 1.99×10^{-5}]
schrodinger‡	narrow	B1	2.97×10^{-1} [2.71×10^{-3}, 3.11×10^{-1}]
	standard	B2	1.83×10^{-1} [2.25×10^{-3}, 2.00×10^{-1}]
	wide	B3	3.11×10^{-1} [2.04×10^{-3}, 3.70×10^{-1}]
schrodinger_hc	narrow	C1	7.46×10^{-3} [7.38×10^{-3}, 7.51×10^{-3}]
	standard	C2	7.35×10^{-3} [7.31×10^{-3}, 7.39×10^{-3}]
	wide	C3	7.35×10^{-3} [7.31×10^{-3}, 7.42×10^{-3}]

C SIGN RE-BASIN

Git Re-Basin was designed to merge models modulo permutations and does not account for the sign symmetries induced by the choice of an odd activation function such as $\tanh$. This is the case for most PINN architectures, including the ones used in this article. To address this gap, we extend Ainsworth et al. (2023) below, an extension we call *Sign Re-Basin*.

C.1 PERMUTATION AND SIGN SYMMETRIES

Permutation symmetry. Let us introduce two fundamental concepts from Ainsworth et al. (2023) in the form of the following definition and proposition, which we build upon:

Definition 3 (Permutation action on parameters)**.** *Let $u(\cdot\,;\boldsymbol{\theta})$ be a feed-forward network with L hidden layers of widths $d_1,\dots,d_L$. The parameters are*[5] *$\boldsymbol{\theta}=(\boldsymbol{W}_1,\boldsymbol{b}_1,\dots,\boldsymbol{W}_L,\boldsymbol{b}_L,\boldsymbol{W}_{L+1})$, where $\boldsymbol{W}_\ell\in\mathbb{R}^{d_\ell\times d_{\ell-1}}$ is the weight matrix of layer ℓ, $\boldsymbol{b}_\ell\in\mathbb{R}^{d_\ell}$ its bias vector, and $\boldsymbol{W}_{L+1}\in\mathbb{R}^{1\times d_L}$ the output layer (we write the scalar output as $u(\boldsymbol{x};\boldsymbol{\theta})=\boldsymbol{W}_{L+1}\,\boldsymbol{h}_L(\boldsymbol{x})$, with $\boldsymbol{h}_L$ the last hidden activation).*

For any tuple of permutation matrices $\pi=(\boldsymbol{P}_1,\dots,\boldsymbol{P}_L)$ with $\boldsymbol{P}_\ell\in S_{d_\ell}$ (i.e. the group of $d_\ell\times d_\ell$ permutation matrices), define the permuted parameters $\pi(\boldsymbol{\theta})$ *by*

$$\widetilde{\boldsymbol{W}}_\ell=\boldsymbol{P}_\ell\,\boldsymbol{W}_\ell\,\boldsymbol{P}_{\ell-1}^\top,\qquad \widetilde{\boldsymbol{b}}_\ell=\boldsymbol{P}_\ell\,\boldsymbol{b}_\ell,\qquad \ell=1,\dots,L, \tag{16}$$

with $\boldsymbol{P}_0=\boldsymbol{I}$ (the input layer is not permuted) and $\widetilde{\boldsymbol{W}}_{L+1}=\boldsymbol{W}_{L+1}\,\boldsymbol{P}_L^\top$.

Proposition 1 (Permutation invariance)**.** *For any tuple of permutations π, $u(\cdot\,;\pi(\boldsymbol{\theta}))=u(\cdot\,;\boldsymbol{\theta})$.*

Proof. For every input $\boldsymbol{x}$, we show by induction on ℓ that the hidden activations of the permuted network satisfy $\widetilde{\boldsymbol{h}}_\ell(\boldsymbol{x})=\boldsymbol{P}_\ell\,\boldsymbol{h}_\ell(\boldsymbol{x})$, where $\boldsymbol{h}_\ell(\boldsymbol{x})=\sigma(\boldsymbol{W}_\ell\,\boldsymbol{h}_{\ell-1}(\boldsymbol{x})+\boldsymbol{b}_\ell)$ and $\boldsymbol{h}_0(\boldsymbol{x})=\boldsymbol{x}$. Since σ acts elementwise and $\boldsymbol{P}_\ell$ is a permutation matrix, $\sigma(\boldsymbol{P}_\ell\,\boldsymbol{z})=\boldsymbol{P}_\ell\,\sigma(\boldsymbol{z})$ for every $\boldsymbol{z}$; together with $\boldsymbol{P}_{\ell-1}^\top\boldsymbol{P}_{\ell-1}=\boldsymbol{I}$ this gives

$$\begin{aligned}\widetilde{\boldsymbol{h}}_\ell(\boldsymbol{x})&=\sigma\big(\widetilde{\boldsymbol{W}}_\ell\,\widetilde{\boldsymbol{h}}_{\ell-1}(\boldsymbol{x})+\widetilde{\boldsymbol{b}}_\ell\big)=\sigma\big(\boldsymbol{P}_\ell\,\boldsymbol{W}_\ell\,\boldsymbol{P}_{\ell-1}^\top\,\boldsymbol{P}_{\ell-1}\,\boldsymbol{h}_{\ell-1}(\boldsymbol{x})+\boldsymbol{P}_\ell\,\boldsymbol{b}_\ell\big)\\&=\boldsymbol{P}_\ell\,\sigma\big(\boldsymbol{W}_\ell\,\boldsymbol{h}_{\ell-1}(\boldsymbol{x})+\boldsymbol{b}_\ell\big)=\boldsymbol{P}_\ell\,\boldsymbol{h}_\ell(\boldsymbol{x}),\end{aligned}$$

the base case $\ell=1$ being the same computation with $\boldsymbol{P}_0=\boldsymbol{I}$. At the output layer,

$$u(\boldsymbol{x};\pi(\boldsymbol{\theta}))=\widetilde{\boldsymbol{W}}_{L+1}\,\widetilde{\boldsymbol{h}}_L(\boldsymbol{x})=\boldsymbol{W}_{L+1}\,\boldsymbol{P}_L^\top\,\boldsymbol{P}_L\,\boldsymbol{h}_L(\boldsymbol{x})=u(\boldsymbol{x};\boldsymbol{\theta}).\qquad\square$$

The equivalence class $[\boldsymbol{\theta}]=\{\pi(\boldsymbol{\theta}):\pi\in\prod_\ell S_{d_\ell}\}$ has at most $\prod_{\ell=1}^{L}d_\ell!$ elements. Two checkpoints $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$ representing similar functions can therefore sit at very large Euclidean distance $\|\boldsymbol{\theta}_a-\boldsymbol{\theta}_b\|$ simply because their neurons are ordered differently.

Sign symmetry of odd activations. All of the above concerns a symmetry that any feed-forward network possesses, whatever its activation. When σ is odd, a second and independent family of symmetries appears, which acts on the *orientation* of each neuron rather than on its position within its layer.

Definition 4 (Sign-flip action on parameters)**.** *Let $u(\cdot\,;\boldsymbol{\theta})$ be the network of Definition 3, with L hidden layers of widths $d_1,\dots,d_L$ and parameters $\boldsymbol{\theta}=(\boldsymbol{W}_1,\boldsymbol{b}_1,\dots,\boldsymbol{W}_L,\boldsymbol{b}_L,\boldsymbol{W}_{L+1})$, and let σ be an* odd *activation function, i.e. $\sigma(-z)=-\sigma(z)$ for all $z\in\mathbb{R}$.*

For any tuple $\xi=(\xi_1,\dots,\xi_L)$ with $\xi_\ell=(\epsilon_1^{(\ell)},\dots,\epsilon_{d_\ell}^{(\ell)})\in\{-1,+1\}^{d_\ell}$ a sign vector, denote by $\boldsymbol{S}_\ell=\operatorname{diag}(\epsilon_1^{(\ell)},\dots,\epsilon_{d_\ell}^{(\ell)})$ its matrix representation and define the sign-flipped parameters $\xi(\boldsymbol{\theta})$ *by*

$$\widetilde{\boldsymbol{W}}_\ell=\boldsymbol{S}_\ell\,\boldsymbol{W}_\ell\,\boldsymbol{S}_{\ell-1}^\top,\qquad \widetilde{\boldsymbol{b}}_\ell=\boldsymbol{S}_\ell\,\boldsymbol{b}_\ell,\qquad \ell=1,\dots,L, \tag{17}$$

with $\boldsymbol{S}_0=\boldsymbol{I}$ (the input layer is not flipped) and $\widetilde{\boldsymbol{W}}_{L+1}=\boldsymbol{W}_{L+1}\,\boldsymbol{S}_L^\top$.

[5] Throughout, $\boldsymbol{\theta}$ is also viewed as a single vector in $\mathbb{R}^D$ by concatenating the vectorisations of all weight matrices and bias vectors, where $D=\sum_{\ell=1}^{L}(d_\ell d_{\ell-1}+d_\ell)+d_L+1$ (output bias included).

The action in equation 17 is written so as to mirror equation 16 term by term; since each $\boldsymbol{S}_\ell$ is diagonal, $\boldsymbol{S}_{\ell-1}^\top = \boldsymbol{S}_{\ell-1}$, and since its entries represent signs, $\boldsymbol{S}_\ell^2 = \boldsymbol{I}$, so each $\boldsymbol{S}_\ell$ is both symmetric and orthogonal.

Proposition 2 (Sign-flip invariance)**.** *Let σ be odd. For any tuple of sign vectors ξ, $u(\cdot\,; \xi(\boldsymbol{\theta})) = u(\cdot\,; \boldsymbol{\theta})$.*

Proof. For every input $\boldsymbol{x}$, we show by induction on ℓ that the hidden activations of the sign-flipped network satisfy $\widetilde{\boldsymbol{h}}_\ell(\boldsymbol{x}) = \boldsymbol{S}_\ell\, \boldsymbol{h}_\ell(\boldsymbol{x})$. Since σ acts elementwise and is odd, and since the entries of $\boldsymbol{S}_\ell$ are signs, $\big(\sigma(\boldsymbol{S}_\ell\, \boldsymbol{z})\big)_i = \sigma(\epsilon_i^{(\ell)} z_i) = \epsilon_i^{(\ell)}\, \sigma(z_i)$, that is $\sigma(\boldsymbol{S}_\ell\, \boldsymbol{z}) = \boldsymbol{S}_\ell\, \sigma(\boldsymbol{z})$ for every $\boldsymbol{z}$; together with $\boldsymbol{S}_{\ell-1}^\top \boldsymbol{S}_{\ell-1} = \boldsymbol{I}$ this gives

$$\begin{aligned}\widetilde{\boldsymbol{h}}_\ell(\boldsymbol{x}) &= \sigma\big(\widetilde{\boldsymbol{W}}_\ell\, \widetilde{\boldsymbol{h}}_{\ell-1}(\boldsymbol{x}) + \widetilde{\boldsymbol{b}}_\ell\big) = \sigma\big(\boldsymbol{S}_\ell\, \boldsymbol{W}_\ell\, \boldsymbol{S}_{\ell-1}^\top\, \boldsymbol{S}_{\ell-1}\, \boldsymbol{h}_{\ell-1}(\boldsymbol{x}) + \boldsymbol{S}_\ell\, \boldsymbol{b}_\ell\big) \\ &= \boldsymbol{S}_\ell\, \sigma\big(\boldsymbol{W}_\ell\, \boldsymbol{h}_{\ell-1}(\boldsymbol{x}) + \boldsymbol{b}_\ell\big) = \boldsymbol{S}_\ell\, \boldsymbol{h}_\ell(\boldsymbol{x}),\end{aligned}$$

the base case $\ell = 1$ being the same computation with $\boldsymbol{S}_0 = \boldsymbol{I}$. At the output layer,

$$u(\boldsymbol{x}; \xi(\boldsymbol{\theta})) = \widetilde{\boldsymbol{W}}_{L+1}\, \widetilde{\boldsymbol{h}}_L(\boldsymbol{x}) = \boldsymbol{W}_{L+1}\, \boldsymbol{S}_L^\top\, \boldsymbol{S}_L\, \boldsymbol{h}_L(\boldsymbol{x}) = u(\boldsymbol{x}; \boldsymbol{\theta}). \qquad \square$$

Note that the two proofs differ in exactly one step. Permutation matrices commute with σ because σ is applied elementwise, whatever σ is; sign matrices commute with it only because σ is odd. Everything else (the telescoping $\boldsymbol{S}_{\ell-1}^\top \boldsymbol{S}_{\ell-1} = \boldsymbol{I}$ and the cancellation at the output layer) is the orthogonality of the acting matrices, which both families share. This is why both symmetries can be combined below.

The sign-flip group $\prod_{\ell=1}^{L} \{-1, +1\}^{d_\ell}$ has $2^{d_1 + \cdots + d_L}$ elements, and its orbit through a given $\boldsymbol{\theta}$ has as many distinct points whenever no bias entry vanishes, since $\widetilde{\boldsymbol{b}}_\ell = \boldsymbol{S}_\ell \boldsymbol{b}_\ell$ then determines $\boldsymbol{S}_\ell$. Its effect on parameter distance is, however, markedly different from that of a permutation. Flipping layer ℓ and layer $\ell - 1$ together leaves $\boldsymbol{W}_\ell$ untouched, because the two sign matrices cancel on either side of it in equation 17; a sign tuple can therefore move a checkpoint a long way in parameter space while barely touching most of its weight matrices, or move it hardly at all while flipping every neuron. We return to this in Section C.2.3, where it motivates the multistart initialisation of our weight-matching algorithm.

The signed permutation group. In the extreme case of a network with a single hidden layer, taking $\xi_1 = (-1, \ldots, -1)$ gives $\xi(\boldsymbol{\theta}) = -\boldsymbol{\theta}$, so two ingredients may be antipodal in parameter space and identical as functions, yet average to the zero function. Therefore, not accounting for possible sign symmetries can act as a destructive process when averaging in weight space, an effect that merging modulo permutation alone does not mitigate.

Because both actions commute with σ, act by orthogonal matrices and act on the same index set, they may be applied one after the other. Together, the permutation action of Definition 3 and the sign-flip action of Definition 4 generate the hyperoctahedral group $(\mathbb{Z}/2\mathbb{Z})^d \rtimes S_d$, of order $2^d \cdot d!$ per hidden layer, which we now introduce formally (Björner & Brenti, 2005):

Definition 5 (Signed permutation group)**.** *The* signed permutation group *of degree d is the set*

$$G_d \;:=\; (\mathbb{Z}/2\mathbb{Z})^d \rtimes S_d \;=\; \big\{(\xi,\, \pi) \;\big|\; \xi = (\epsilon_1, \ldots, \epsilon_d) \in \{-1, +1\}^d,\; \pi \in S_d\big\},$$

with $|G_d| = 2^d \cdot d!$, equipped with the following structure.

(i) ***Action of S_d on $(\mathbb{Z}/2\mathbb{Z})^d$.*** *Permutations act on sign vectors by permuting coordinates:*

$$\pi \cdot \xi \;:=\; (\epsilon_{\pi^{-1}(1)},\; \ldots,\; \epsilon_{\pi^{-1}(d)}).$$

Permuting neurons and then flipping neuron j is the same as flipping the neuron that π sends to position j.

(ii) ***Composition law.***

$$(\xi,\, \pi) \cdot (\zeta,\, \rho) \;:=\; \big(\xi \odot (\pi \cdot \zeta),\;\; \pi \circ \rho\big), \qquad \big(\xi \odot (\pi \cdot \zeta)\big)_j = \epsilon_j \cdot \zeta_{\pi^{-1}(j)},$$

with $\odot$ the componentwise product in $\{-1, +1\}^d$. One verifies directly that this product is associative.

(iii) ***Identity and inverses.*** *The identity is* $e = (\xi^+, \mathrm{id})$ *with* $\xi^+ = (+1, \dots, +1)$*, and*

$$(\xi,\, \pi)^{-1} \;=\; (\pi^{-1} \cdot \xi,\;\; \pi^{-1}),$$

since every sign vector is its own inverse under $\odot$*.*

(iv) ***Action on network parameters.*** *An element* $(\xi,\, \pi) \in G_d$ *acts on* $\boldsymbol{\theta} = (\boldsymbol{W},\, \boldsymbol{b},\, \boldsymbol{v})$ *by*

$$(\xi,\, \pi)(\boldsymbol{\theta}) \;:=\; \big(\boldsymbol{S}\,\boldsymbol{P}\,\boldsymbol{W},\;\; \boldsymbol{S}\,\boldsymbol{P}\,\boldsymbol{b},\;\; \boldsymbol{S}\,\boldsymbol{P}\,\boldsymbol{v}\big), \tag{18}$$

where $\boldsymbol{P}$ *is the permutation matrix of* π *and* $\boldsymbol{S}$ *the matrix representation of* ξ*.*

Accounting for these symmetries when cooking PINN soups. Practical PINNs are not single-hidden-layer networks; indeed, common architectures use $L \geq 3$ hidden layers of widths $d_1, \dots, d_L$, so the full symmetry group is the product

$$\Gamma \;=\; \Gamma_{d_1} \times \cdots \times \Gamma_{d_L} = \prod_{\ell=1}^{L} \Gamma_{d_\ell}, \tag{19}$$

with $\Gamma_{d_\ell} = S_{d_\ell}$ or G_{d_ℓ} for general and odd activations respectively. At uniform width d this group has $(\,2^d \cdot d!\,)^L$ elements in the worst-case scenario. The per-layer problems do not decouple: the (signed) permutation chosen for layer ℓ constrains the admissible choices for layers $\ell - 1$ and $\ell + 1$, so alignment is a coupled combinatorial optimisation across all L layers at once.

Definition 6 (Signed permutation action on parameters)**.** *Let* $u(\cdot\,; \boldsymbol{\theta})$ *be as in Definition 3, with parameters* $\boldsymbol{\theta} = (\boldsymbol{W}_1, \boldsymbol{b}_1, \dots, \boldsymbol{W}_L, \boldsymbol{b}_L, \boldsymbol{W}_{L+1})$*, and let* σ *be an odd activation function.*

For any tuple $g = (g^{(1)}, \dots, g^{(L)})$ *with* $g^{(\ell)} = (\xi_\ell, \pi_\ell) \in G_{d_\ell}$*, denote by*

$$\boldsymbol{Q}_\ell = \boldsymbol{S}_\ell\, \boldsymbol{P}_\ell, \qquad \boldsymbol{S}_\ell = \mathrm{diag}\,(\epsilon_1^{(\ell)}, \dots, \epsilon_{d_\ell}^{(\ell)}),$$

the associated signed permutation matrix*, with* $\boldsymbol{S}_\ell$ *and* $\boldsymbol{P}_\ell$ *the respective representation matrices of* ξ_ℓ *and* π_ℓ*, and define the* signed permuted parameters $g(\boldsymbol{\theta})$ *by*

$$\widetilde{\boldsymbol{W}}_\ell = \boldsymbol{Q}_\ell\, \boldsymbol{W}_\ell\, \boldsymbol{Q}_{\ell-1}^\top, \qquad \widetilde{\boldsymbol{b}}_\ell = \boldsymbol{Q}_\ell\, \boldsymbol{b}_\ell, \qquad \ell = 1, \dots, L, \tag{20}$$

with $\boldsymbol{Q}_0 = \boldsymbol{I}$ *(the input layer is neither permuted nor flipped) and* $\widetilde{\boldsymbol{W}}_{L+1} = \boldsymbol{W}_{L+1}\, \boldsymbol{Q}_L^\top$*.*

Signed permutation matrices are orthogonal, $\boldsymbol{Q}^\top \boldsymbol{Q} \;=\; \boldsymbol{P}^\top \boldsymbol{S}^2\, \boldsymbol{P} \;=\; \boldsymbol{I}$, and their entries are $Q_{ij} \;=\; \epsilon_i\, \delta_{i,\pi(j)}$ with $\delta_{p,q}$ the Kronecker delta. Setting $L \;=\; 1$ recovers the single-hidden-layer action of equation 18.

Definition 7 (Symmetry-aligned soup)**.** *Let* Γ *be the symmetry group of equation 19, with elements* $g = (g^{(1)}, \dots, g^{(L)})$*,* $g^{(\ell)} \in \Gamma_{d_\ell}$*. Given ingredients* $\{\boldsymbol{\theta}_i\}_{i=1}^k$ *and a reference* $\boldsymbol{\theta}^*$ *(in our experiments, the best ingredient), a* symmetry-aligned soup *is*

$$\bar{\boldsymbol{\theta}} = \sum_{i=1}^{k} \beta_i\, g_i^*(\boldsymbol{\theta}_i), \qquad \sum_{i=1}^{k} \beta_i = 1, \quad \beta_i \geq 0,$$

where

$$g_i^* = \arg\min_{g \,\in\, \Gamma} \|g(\boldsymbol{\theta}_i) - \boldsymbol{\theta}^*\|^2 \tag{21}$$

is the group element that best aligns ingredient i *to the reference.*

Proposition 3 (Perfect alignment eliminates any interference)**.** *Suppose the ingredients are related by exact symmetry-group transformations of a common parameter vector:* $\boldsymbol{\theta}_i = g_i(\boldsymbol{\theta}^*)$ *for some* $\boldsymbol{\theta}^*$ *and* $g_i \in \Gamma$*,* $i = 1, \dots, k$*. Then the optimal alignment elements satisfy* $g_i^*(\boldsymbol{\theta}_i) = \boldsymbol{\theta}^*$ *for all* i*, and the symmetry-aligned soup reduces to*

$$\bar{\boldsymbol{\theta}} = \sum_{i=1}^{k} \beta_i\, \boldsymbol{\theta}^* = \boldsymbol{\theta}^*.$$

In particular the interference vanishes: $u(\cdot\,; \bar{\boldsymbol{\theta}}) = u(\cdot\,; \boldsymbol{\theta}^*)$*.*

Proof. Since $\boldsymbol{\theta}_i = g_i(\boldsymbol{\theta}^*)$, then $g_i^* = \arg\min_{g \in \Gamma} \|g(g_i(\boldsymbol{\theta}^*)) - \boldsymbol{\theta}^*\|^2$. Taking $g = g_i^{-1} \in \Gamma$ gives $\|g_i^{-1}(\boldsymbol{\theta}_i) - \boldsymbol{\theta}^*\| = 0$, the global minimum. Therefore $g_i^*(\boldsymbol{\theta}_i) = \boldsymbol{\theta}^*$ for all i, and $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}^*$. □

This result treats the idealised case in which the ingredients are exact symmetry images of a common parameter vector $\boldsymbol{\theta}^*$; trained models differ by more than a group transformation, so in practice alignment reduces the interference without eliminating it. More importantly, the result depends on the group being searched, not on the quality of the search. An alignment method restricted to $\prod_\ell S_{d_\ell} \subset \prod_\ell G_{d_\ell}$ may return a permutation tuple whose Euclidean distance is substantially larger than the true optimum found over Γ. Standard Git Re-Basin implementations do not account for sign flips and therefore require the extension we develop next.

C.2 Merging Models Modulo Signed Permutation

We extend both matching heuristics of Git Re-Basin (Ainsworth et al., 2023) from the permutation group to the full multi-layer hyperoctahedral group of equation 19. Both rest on the same exact layerwise reduction, which we establish first after recording the properties of the action that make the alignment problem tractable at all.

C.2.1 Exact Layerwise Elimination of the Sign Component

Proposition 4 (Signed permutation properties)**.** *Let σ be odd. Then:*

(i) ***Equivariance.*** *For every $g \in G_d$ and every $\boldsymbol{z} \in \mathbb{R}^d$, $\sigma(\boldsymbol{Q}\,\boldsymbol{z}) = \boldsymbol{Q}\,\sigma(\boldsymbol{z})$, where $\boldsymbol{Q}$ is the matrix representation of g.*

(ii) ***Invariance.*** *For every $g \in \Gamma$, $u(\cdot\,; g(\boldsymbol{\theta})) = u(\cdot\,; \boldsymbol{\theta})$.*

(iii) ***Isometry.*** *The action is a parameter-space isometry: $\|g(\boldsymbol{\theta})\| = \|\boldsymbol{\theta}\|$ for all $\boldsymbol{\theta}$ and $g \in \Gamma$. Consequently the alignment problem from equation 21 over Γ is equivalent to inner-product maximisation:*

$$\begin{aligned}\arg\min_{g \in \Gamma} \|g(\boldsymbol{\theta}') - \boldsymbol{\theta}\|^2 = \arg\max_{g \in \Gamma} \sum_{\ell=1}^{L} \Big[\big\langle \boldsymbol{W}_\ell,\, \boldsymbol{Q}_\ell \boldsymbol{W}'_\ell \boldsymbol{Q}_{\ell-1}^\top \big\rangle_F + \big\langle \boldsymbol{b}_\ell,\, \boldsymbol{Q}_\ell \boldsymbol{b}'_\ell \big\rangle_F\Big] \\ + \big\langle \boldsymbol{W}_{L+1},\, \boldsymbol{W}'_{L+1} \boldsymbol{Q}_L^\top \big\rangle_F,\end{aligned} \tag{22}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product.

Proof. (i) Componentwise, $(\boldsymbol{Q}\,\boldsymbol{z})_i = \epsilon_i\, z_{\pi^{-1}(i)}$, so $\sigma\big((\boldsymbol{Q}\,\boldsymbol{z})_i\big) = \epsilon_i\, \sigma(z_{\pi^{-1}(i)}) = (\boldsymbol{Q}\,\sigma(\boldsymbol{z}))_i$, using oddness of σ and $\epsilon_i \in \{-1, +1\}$.

(ii) By induction, exactly as in the proof of Proposition 1, replacing each $\boldsymbol{P}_\ell$ by $\boldsymbol{Q}_\ell$: the elementwise-commutation step is replaced by part (i), and orthogonality $\boldsymbol{Q}_{\ell-1}^\top \boldsymbol{Q}_{\ell-1} = \boldsymbol{I}$ plays the role of $\boldsymbol{P}_{\ell-1}^\top \boldsymbol{P}_{\ell-1} = \boldsymbol{I}$. Moreover, the hidden activations satisfy $\tilde{\boldsymbol{h}}_\ell(\boldsymbol{x}) = \boldsymbol{Q}_\ell\, \boldsymbol{h}_\ell(\boldsymbol{x})$ for all ℓ.

(iii) Each block of $g(\boldsymbol{\theta}')$ is obtained from the corresponding block of $\boldsymbol{\theta}'$ by multiplication with orthogonal matrices, which preserves Frobenius norms. Therefore, expanding

$$\|g(\boldsymbol{\theta}') - \boldsymbol{\theta}\|^2 = \|\boldsymbol{\theta}\|^2 + \|\boldsymbol{\theta}'\|^2 - 2\langle \boldsymbol{\theta},\, g(\boldsymbol{\theta}') \rangle$$

and dropping the g-independent terms gives equation 22. □

The workhorse behind the proposed algorithms is the following exact reduction: inside each layer-wise matching subproblem the sign component can be *eliminated exactly and in closed form*, the maximisation over the $2^d\, d!$-element group G_d collapsing to a single linear assignment problem (LAP) on the entrywise absolute value of the similarity matrix. For a matrix $\boldsymbol{A}$, denote by $|\boldsymbol{A}|$ its entrywise absolute value, and let

$$\mathrm{LAP}(\boldsymbol{A}) = \arg\max_{\boldsymbol{P} \in S_d} \langle \boldsymbol{P},\, \boldsymbol{A} \rangle_F$$

denote the linear assignment problem with score matrix $\boldsymbol{A}$, where each permutation is identified with its permutation matrix.

Proposition 5 (Exact layerwise elimination of the sign component)**.** *For any* $\boldsymbol{A} \in \mathbb{R}^{d\times d}$*,*

$$\max_{\boldsymbol{Q}\in G_d} \langle \boldsymbol{Q},\, \boldsymbol{A}\rangle_F \;=\; \max_{\boldsymbol{P}\in S_d} \langle \boldsymbol{P},\, |\boldsymbol{A}|\rangle_F, \tag{23}$$

where G_d *(resp.* S_d*) is identified with the set of* $(d \times d)$ *signed permutation (resp. permutation) matrices. A maximiser is* $\boldsymbol{Q}^* = \boldsymbol{S}^*\,\boldsymbol{P}^*$ *with* $\boldsymbol{P}^* = \mathrm{LAP}(|\boldsymbol{A}|)$ *and* $\epsilon_i^* = \mathrm{sign}\big(A_{i,\,(\pi^*)^{-1}(i)}\big)$ *(taking* $\epsilon_i^* = +1$ *when the matched entry vanishes). Consequently the layerwise subproblem over* G_d *is solved* exactly *in* $O(d^3)$ *time (Hungarian or Jonker–Volgenant assignment on* $|\boldsymbol{A}|$*) and* $O(d^2)$ *memory: the same asymptotic cost as the permutation-only subproblem.*

Proof. Writing $\boldsymbol{Q} = \boldsymbol{S}\,\boldsymbol{P}$ with $Q_{ij} = \epsilon_i\,\delta_{i,\pi(j)}$, only the entries selected by $\boldsymbol{P}$ contribute, each modulated by the corresponding sign:

$$\langle \boldsymbol{Q},\, \boldsymbol{A}\rangle_F = \sum_{i,j} Q_{ij}\,A_{ij} = \sum_{j=1}^{d} \epsilon_{\pi(j)}\,A_{\pi(j),\,j}.$$

For fixed $\boldsymbol{P}$ the objective is separable in the signs: each term satisfies $\epsilon_{\pi(j)}\,A_{\pi(j),j} \le |A_{\pi(j),j}|$, with equality at $\epsilon_{\pi(j)} = \mathrm{sign}(A_{\pi(j),j})$ (any sign if the entry is zero). Since π is a bijection, every sign coordinate is optimised independently, so $\max_{\boldsymbol{S}} \langle \boldsymbol{S}\,\boldsymbol{P},\, \boldsymbol{A}\rangle_F = \langle \boldsymbol{P},\, |\boldsymbol{A}|\rangle_F$. Taking the outer maximum over $\boldsymbol{P}$ gives equation 23. □

Enlarging the search group from S_d to G_d is thus exactly equivalent to replacing the score matrix $\boldsymbol{A}$ by $|\boldsymbol{A}|$: the exponential factor 2^d incurs *no* additional asymptotic cost relative to permutation-only matching. As in the unsigned case, the only difficult part is the coupling of permutations across layers. With this foundation, we now present our extension of Ainsworth et al. (2023) to the sign-symmetry scenario.

C.2.2 Matching Activations

Fix a finite matching set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and let $\boldsymbol{H}_\ell = \big(\boldsymbol{h}_\ell(\boldsymbol{x}_1)\ \cdots\ \boldsymbol{h}_\ell(\boldsymbol{x}_N)\big) \in \mathbb{R}^{d_\ell\times N}$ collect the layer-ℓ hidden activations of $u(\cdot\,;\boldsymbol{\theta})$ (resp. $\boldsymbol{H}'_\ell$ and $u(\cdot\,;\boldsymbol{\theta}')$) on $\mathcal{X}$. The aim is for the aligned activations to be as close as possible to those of the reference ingredient. Since

$$\|\boldsymbol{H}_\ell - \boldsymbol{Q}\,\boldsymbol{H}'_\ell\|_F^2 \;=\; \|\boldsymbol{H}_\ell\|_F^2 \;+\; \|\boldsymbol{H}'_\ell\|_F^2 \;-\; 2\,\langle \boldsymbol{Q},\, \boldsymbol{C}_\ell\rangle_F$$

with cross-correlation matrix $\boldsymbol{C}_\ell := \boldsymbol{H}_\ell\,\boldsymbol{H}_\ell'^{\top}$, the per-layer functional matching problem

$$\boldsymbol{Q}_\ell^* = \arg\min_{\boldsymbol{Q}\in G_{d_\ell}} \|\boldsymbol{H}_\ell - \boldsymbol{Q}\,\boldsymbol{H}'_\ell\|_F^2$$

is solved *globally* by Proposition 5 applied to $\boldsymbol{C}_\ell$. As in unsigned activation matching, the activations $\boldsymbol{H}'_\ell$ are computed from the *untransformed* $\boldsymbol{\theta}'$, so the L subproblems decouple and no coordinate descent is needed. Algorithm 1 summarises the procedure, which we call Sign-AM.

Algorithm 1: Sign-ActivationMatching

Require: reference $\boldsymbol{\theta}$, candidate $\boldsymbol{\theta}'$, matching set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$

for $\ell = 1, \ldots, L$ **do**

- $\boldsymbol{H}_\ell \leftarrow \big(\boldsymbol{h}_\ell(\boldsymbol{x}_1), \ldots, \boldsymbol{h}_\ell(\boldsymbol{x}_N)\big) \in \mathbb{R}^{d_\ell\times N}$ — *// activations of* $\boldsymbol{\theta}$
- $\boldsymbol{H}'_\ell \leftarrow$ same for $\boldsymbol{\theta}'$
- $\boldsymbol{C}_\ell \leftarrow \boldsymbol{H}_\ell\,\boldsymbol{H}_\ell'^{\top}$
- $\boldsymbol{P}_\ell \leftarrow \mathrm{LAP}(|\boldsymbol{C}_\ell|)$ — *// $O(d_\ell^3)$*
- $\epsilon_i^{(\ell)} \leftarrow \mathrm{sign}\big(\boldsymbol{C}_\ell[i, (\pi^{(\ell)})^{-1}(i)]\big)$ for $i = 1, \ldots, d_\ell$ — *// $+1$ on zero entries*
- $\boldsymbol{Q}_\ell \leftarrow \boldsymbol{S}_\ell\,\boldsymbol{P}_\ell$

return $g^* \equiv (\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_L)$ *and the aligned parameters* $g^*(\boldsymbol{\theta}')$ *via equation 20*

C.2.3 MATCHING WEIGHTS

Alternatively, one may attack the parameter-space objective directly. Fixing all $\boldsymbol{Q}_m$, $m \neq \ell$, and collecting the terms of equation 22 involving $\boldsymbol{Q}_\ell$, the Frobenius identities

$$\langle \boldsymbol{W}_\ell,\, \boldsymbol{Q}_\ell \boldsymbol{W}'_\ell \boldsymbol{Q}_{\ell-1}^\top \rangle_F = \langle \boldsymbol{Q}_\ell,\, \boldsymbol{W}_\ell \boldsymbol{Q}_{\ell-1} \boldsymbol{W}_\ell'^\top \rangle_F,$$
$$\langle \boldsymbol{b}_\ell,\, \boldsymbol{Q}_\ell \boldsymbol{b}'_\ell \rangle_F = \langle \boldsymbol{Q}_\ell,\, \boldsymbol{b}_\ell \boldsymbol{b}_\ell'^\top \rangle_F,$$
$$\langle \boldsymbol{W}_{\ell+1},\, \boldsymbol{Q}_{\ell+1} \boldsymbol{W}'_{\ell+1} \boldsymbol{Q}_\ell^\top \rangle_F = \langle \boldsymbol{Q}_\ell,\, \boldsymbol{W}_{\ell+1}^\top \boldsymbol{Q}_{\ell+1} \boldsymbol{W}'_{\ell+1} \rangle_F$$

show that the coordinate update for layer ℓ maximises $\langle \boldsymbol{Q}_\ell,\, \boldsymbol{M}_\ell \rangle_F$ with

$$\boldsymbol{M}_\ell \;=\; \boldsymbol{W}_\ell\, \boldsymbol{Q}_{\ell-1}\, \boldsymbol{W}_\ell'^\top \;+\; \boldsymbol{b}_\ell\, \boldsymbol{b}_\ell'^\top \;+\; \boldsymbol{W}_{\ell+1}^\top\, \boldsymbol{Q}_{\ell+1}\, \boldsymbol{W}'_{\ell+1}, \tag{24}$$

under the conventions $\boldsymbol{Q}_0 = \boldsymbol{I}_{d_0}$ and $\boldsymbol{Q}_{L+1} = \boldsymbol{I}_{d_{L+1}}$, [6] and the subproblem is again solved exactly over G_{d_ℓ} by Proposition 5 applied to $\boldsymbol{M}_\ell$. Iterating over layers yields Algorithm 2 (SIGN-WM), which recovers verbatim the permutation coordinate descent of Git Re-Basin weight matching when all signs are constrained to $+1$. Each update replaces $\boldsymbol{Q}_\ell$ by an exact maximiser of its layer subproblem, so the objective of equation 22 never decreases; since Γ is finite and the sweeps stop as soon as one of them brings no strict increase, the algorithm terminates at a partial optimum, optimal with respect to every single-layer update but not necessarily over Γ.

Algorithm 2: SIGN-WEIGHTMATCHING

Require: reference $\boldsymbol{\theta}$, candidate $\boldsymbol{\theta}'$

Initialise: $\boldsymbol{Q}_\ell \leftarrow \boldsymbol{I}_{d_\ell}$, $\ell = 1, \ldots, L$; $\boldsymbol{Q}_0 \leftarrow \boldsymbol{I}_{d_0}$; $\boldsymbol{Q}_{L+1} \leftarrow \boldsymbol{I}_{d_{L+1}}$

repeat

  for $\ell \in$ RANDOMPERMUTATION$(1, \ldots, L)$ **do**

    $\boldsymbol{M}_\ell \leftarrow \boldsymbol{W}_\ell\, \boldsymbol{Q}_{\ell-1} \boldsymbol{W}_\ell'^\top + \boldsymbol{b}_\ell \boldsymbol{b}_\ell'^\top + \boldsymbol{W}_{\ell+1}^\top \boldsymbol{Q}_{\ell+1} \boldsymbol{W}'_{\ell+1}$ *// equation 24*

    $\boldsymbol{P} \leftarrow \mathrm{LAP}(|\boldsymbol{M}_\ell|)$; $\epsilon_i \leftarrow \mathrm{sign}\big(\boldsymbol{M}_\ell[i, \pi^{-1}(i)]\big)$; $\boldsymbol{Q}_\ell \leftarrow \boldsymbol{S}\,\boldsymbol{P}$

until *a full sweep yields no strict increase of equation 22*

return $g^* \equiv (\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_L)$ *and* $g^*(\boldsymbol{\theta}')$ *via equation 20*

Multistart initialisation. Algorithm 2 is particularly prone to one such partial optimum. As noted in Section C.1, flipping two consecutive layers together leaves the weight matrix between them untouched, so a uniform flip of consecutive layers is invisible to the weights inside and can only be undone by flipping several layers at once, a move no sweep of single-layer updates can make. Algorithm 3 therefore restarts the descent from each of the 2^L layerwise-uniform patterns, or from a random subsample of them when 2^L exceeds a budget R, crossed with the terminal iterate of the *unsigned* descent so as to inherit its progress, and keeps the best run. This costs a constant factor in sweeps and can never return a worse alignment, the identity being itself among the starting points.

Algorithm 3: MULTISTART SIGN-WM

Require: reference $\boldsymbol{\theta}$, candidate $\boldsymbol{\theta}'$, pattern budget R

$\pi^* \leftarrow$ terminal iterate of unsigned weight matching (Git Re-Basin PCD) aligning $\boldsymbol{\theta}'$ to $\boldsymbol{\theta}$

$H \leftarrow \{-1, +1\}^L$ if $2^L \leq R$; otherwise $\{(+1, \ldots, +1),\, (-1, \ldots, -1)\}$ together with $R - 2$ patterns drawn uniformly from $\{-1, +1\}^L$

for $\boldsymbol{P}^{(0)} \in \{\mathrm{id},\, \pi^*\}$ **and** $\eta \in H$ **do**

  run Algorithm 2 initialised at $\boldsymbol{Q}_\ell^{(0)} \leftarrow \eta_\ell\, \boldsymbol{P}_\ell^{(0)}$, $\ell = 1, \ldots, L$

  record the terminal iterate g and its objective $\langle \boldsymbol{\theta},\, g(\boldsymbol{\theta}') \rangle$

return $g^* \leftarrow$ *the recorded* g *with the largest objective*

[6] The output layer is never permuted (its coordinates are functionally distinguishable) so $\boldsymbol{Q}_{L+1}$ unifies the update for $\ell = L$, where the last term of equation 22 contributes $\boldsymbol{W}_{L+1}^\top \boldsymbol{W}'_{L+1}$ to $\boldsymbol{M}_L$.

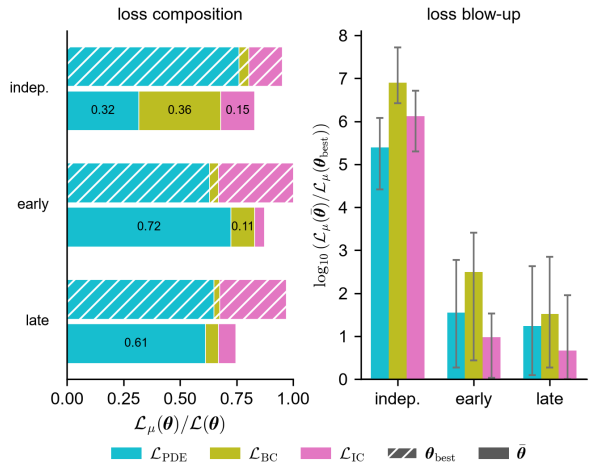


Figure 5: **Loss redistribution.** Left: median share of the physics score carried by each residual term, for the best ingredient (hatched) and the uniform two-ingredient naive soup (solid), soft-constrained problems only. Right: median blow-up of each term in the soup relative to the best ingredient of the same pool, with interquartile range. The boundary residual is the smallest term in the ingredients and the one that inflates most in the soup.

D Extended Results

D.1 Weight Averaging Redistributes the Loss

The physics score quantifies the soup's degradation but not its source. On soft-constrained problems it decomposes into the three residuals of equation 7, and the decomposition identifies it: the term best satisfied by the ingredients is the most affected by averaging (Figure 5). Under `independent` the boundary residual carries a few percent of an ingredient's score against some three quarters for the PDE term, and in the uniform two-ingredient soup it becomes the largest single contribution of the three. It is also the term that inflates most, by a median of almost seven orders of magnitude there, and it leads in every regime and in more than half of the individual configurations. Only under `independent`, though, is the ranking by blow-up the exact reverse of the ranking by size: under the branching regimes the boundary residual still leads, but the term left behind is the initial condition rather than the PDE.

We attribute this to how far each term is from its optimum. The PDE residual is sampled on the full-dimensional interior and is still far from zero when training stops, so it has little room to grow; the boundary and initial residuals are pinned on thin subsets of the domain and reach their small values through a cancellation particular to each ingredient, which a linear interpolation of the parameters has no reason to preserve. This is also consistent with the asymmetry between constraint types reported above: a hard-constrained network satisfies both terms by construction for every $\boldsymbol{\theta}$, averaged ones included, so the most sensitive term is removed from the loss: under the branching regimes, their two-ingredient soups stay at about one percent error in the median, against several percent for soft-constrained soups on the same equations, although under `independent` both reach the error of the trivial predictor. Hard constraints mitigate the degradation but do not prevent it: the PDE residual still inflates, and the soup typically falls short of its best ingredient.

D.2 Full Result Grids

Figure 6 pools the headline quantity of Section 3.1 along the axes the suite is built from. The main text quotes it resolved by problem; here it is averaged instead by constraint enforcement, by the linearity of the differential operator and by width. Figure 7 then gives the pool-size sweep discussed there, with the ensemble track repeated on an axis of its own, since its dynamic range is two orders of magnitude smaller than the soup's and it is otherwise illegible.

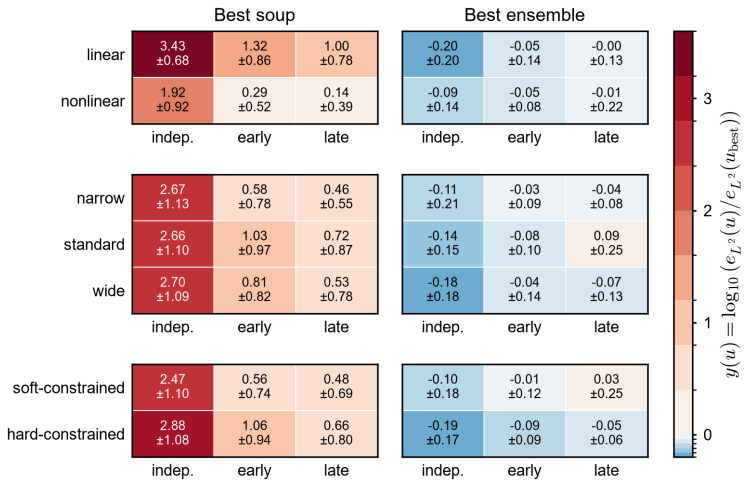


Figure 6: **The comparison pooled along the axes of the suite.** The headline quantity of Figure 1, there resolved by problem, averaged instead over the groups named beside each panel: by the linearity of the differential operator, by width and by constraint enforcement. The soup panel stays on the degradation side in every group. Degradation and ensemble gain both peak on the linear group and both recede on the nonlinear one, and the ensemble advantage is largest under `independent`, consistent with the role of ingredient diversity.

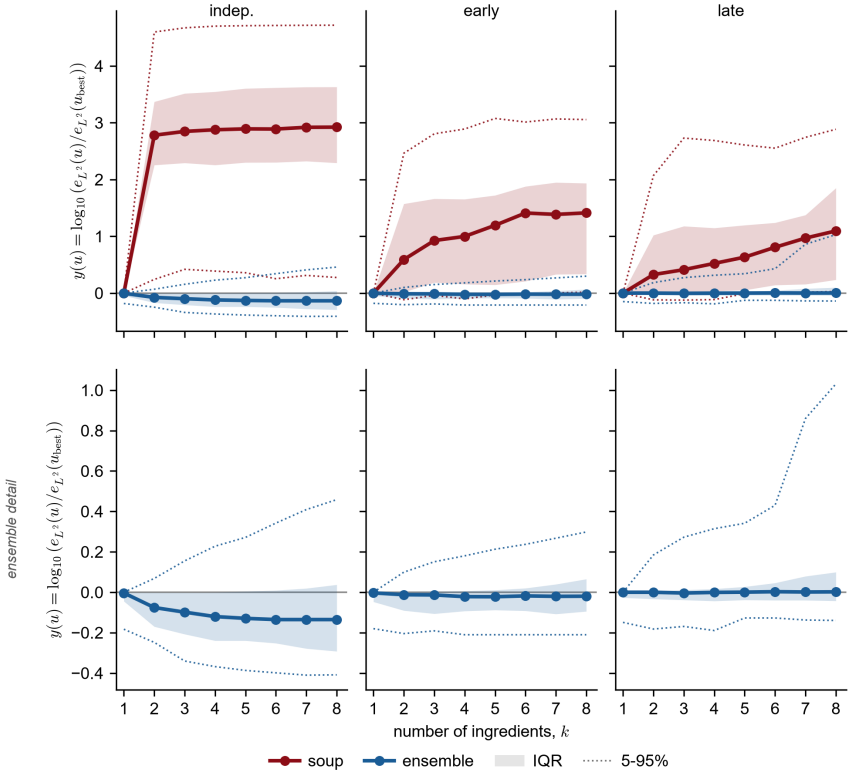


Figure 7: **Forcing the pool size.** Top: best soup and best ensemble at each pool size $k \leq 8$, median over the 36 configurations of each training regime, with interquartile range and 5–95% envelope. At $k = 1$ each track is its own top-ranked ingredient (score for soups, oracle for ensembles). Under `independent` the soup reaches its full degradation at $k = 2$ and remains there; under `early` and `late` it degrades progressively. Bottom: the ensemble track alone, on its own vertical scale. The ensemble benefits mostly from the diversity of the independent training trajectories.

D.3 Soups Against Ensembles, Configuration by Configuration

Figure 8 places the two combinations against each other configuration by configuration, both measured against the same best single ingredient, and is the evidence behind the claim that they do not trade off. The vertical band it carries is the mean selection gap quantified in Appendix D.7, i.e. differences that fall inside it are not distinguishable from having kept a different ingredient in the first place.

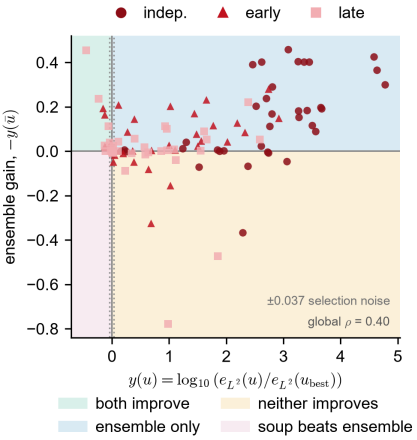


Figure 8: **Soups and ensembles do not trade off.** Soup degradation against ensemble gain, one marker per configuration, both measured against the same best single ingredient. The vertical dotted band is the mean selection gap of Appendix D.7, ± 0.037: differences inside it are not distinguishable from having kept a different ingredient. The quadrant where both improve is reached by 14 configurations, in 9 of which the ensemble's gain is at least as large as the soup's.

D.4 The Barrier and the Quality of the Soup

Figure 9 regresses, per configuration, the degradation of the best soup against the lowest pairwise barrier attained by any of the five parameter-space methods. The two are strongly correlated within every training regime, quantifying the claim of Section 3.4: the barrier along the segment and the eventual quality of the average reflect the same phenomenon, and the failure of weight averaging on these landscapes is geometric rather than a matter of tuning.

This correlation must be interpreted with care. Sweeping the segment means evaluating $\boldsymbol{\theta}(\alpha) = \alpha\boldsymbol{\theta}_i + (1-\alpha)\boldsymbol{\theta}_j$ at every α, and at $\alpha = 1/2$ that model is exactly the uniform two-ingredient soup, so the sweep itself performs the averaging. The correlation is moreover partly implied by the pool-size sweep of Appendix D.2, where under `independent` the damage is already complete at $k = 2$ and flat thereafter, so that a two-ingredient average and the best soup of the whole pool are close to the same measurement. Taking the lowest barrier over the five methods, finally, presupposes that all five alignments have been computed, which is the expensive part of the procedure. What the figure supports is the identification of the two quantities, not a screening test that could be run before committing to a soup.

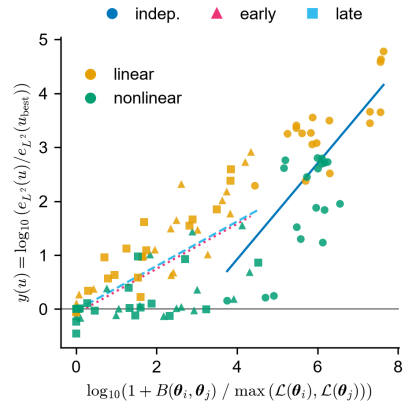


Figure 9: **Barrier and soup degradation are one phenomenon.** Best soup degradation against the lowest pairwise barrier attained by any of the five parameter-space methods, one marker per configuration; colour and shape give the training regime, fill gives the linearity of the operator, and the line is the least-squares fit within each regime. Slopes are: 0.89 `independent`, 0.42 `early` and 0.41 `late`.

D.5 The Barrier in the Error Metric

Section 3.4 scores the segment on $\mathrm{Score}(\cdot)$, the quantity the ingredients were trained to minimise. The same segment can be scored on the relative L^2 error, the quantity a user of the surrogate cares about, and the two need not agree. We therefore write B' for the barrier of equation 4 with the metric $e_{L^2}(\cdot)$ in place of $\mathcal{L}$,

$$B'(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \sup_{\alpha \in [0,1]} e_{L^2}(\boldsymbol{\theta}(\alpha)) - \max\left(e_{L^2}(\boldsymbol{\theta}_i), e_{L^2}(\boldsymbol{\theta}_j)\right). \tag{25}$$

The two share their reference, the worse of the pair's own two endpoints, so $B' = 0$ says in the error metric exactly what $B = 0$ says in the loss: the interpolation path never leaves the level the pair already held. Neither condition implies the other, and the error metric is the more forgiving of the two here, though not by much: $B' = 0$ in 12 of the 108 pairs against 4 for B, and in neither metric under `independent`.

Figure 10 sweeps B' as Figure 4 sweeps B, on the same pairs and with the same series. The conclusions match those obtained with the loss, now in directly interpretable units. Under `independent` the midpoint sits 3.4 decades above the pair's own error before alignment, against the 3.2 decades at which the trivial predictor $u \equiv 0$ sits: in the median, averaging two trained PINNs is worse than returning zero. Alignment moves the midpoint down by about half a decade, to 2.8 under activation matching, and nowhere near the threshold. Under the branching regimes the midpoint stays within 0.4 (`early`) and 0.1 (`late`) decades of the endpoints, and alignment moves it in the same directions and by the same relative amounts as it moves the barrier: weight matching leaves the curves unchanged, since it returns the identity, and activation matching lifts them. Table 8 shows the same on the uniform two-ingredient soups of each method.

D.6 Width Does Not Flatten the Barrier

Width is the axis along which the linear mode connectivity literature states its prediction: the barrier between independently trained solutions is conjectured to vanish as the network widens (Entezari et al., 2022). Figure 11 resolves both barriers along it, in the same median, interquartile and 5–95% summary the rest of the suite is drawn with.

The barrier does not fall with width, in either metric. Under `independent`, the regime the conjecture speaks about, the three medians sit within a quarter of a decade of one another, less than the spread among the twelve problems at any one width: which equation is being solved moves the barrier further than widening the network does. Under the branching regimes it does not fall either, and in the physics score it is higher at the widest width than at the narrowest; under `late` the interquartile range widens with every doubling, so extra width increases variability rather than connectivity. In the physics score nothing comes near the threshold, with medians some six decades above it under `independent` and one to three under the branching regimes.

Our widths are modest, and the conjecture is asymptotic, as noted among the limitations in Section 5. These results do not refute it. They only show that the widths used in practice do not yield connectivity on their own. Width does affect the alignment, though: the wider the network, the less often activation matching returns the identity (Section 3.5).

D.7 Selection Criterion and Selection Noise

Every comparison in the main text is anchored on the ingredient the physics score of equation 3 selects, the criterion available without a reference solution. We define the *selection gap* as the distance in relative L^2 error, in $\log_{10}$, between that ingredient and the best one the pool contains. The gap is exactly zero in half of the 108 configurations and averages 0.037. Under the score ranking, ten configurations admit a soup that improves on the oracle ingredient, five of them the degenerate soft-constrained Allen–Cahn. The margins are small: the largest is 0.127, below the 90th percentile of the gap (0.130), and the largest outside Allen–Cahn is 0.053, for Burgers.

Table 9 reports the criterion as a ranking rather than as a selector, through the Kendall correlation between the score ordering and the oracle ordering of each pool's eight ingredients. Agreement is clear when pooled but noisy per configuration, since eight ingredients provide little statistical power: about half of them reach significance on their own, and a few rank their pool backwards. It is weakest where the ingredients are least alike, under `independent`, and on hard constraints, where the score sees one residual instead of three. The physics score is a usable but noisy proxy: reliable enough for model selection, but not a substitute for the oracle. Table 10 repeats the main-text verdicts with the oracle in place of the score. Fewer soups improve on their reference, and the greedy recipe keeps a few more ingredients, never more than four and none under `independent`. The conclusions do not change.

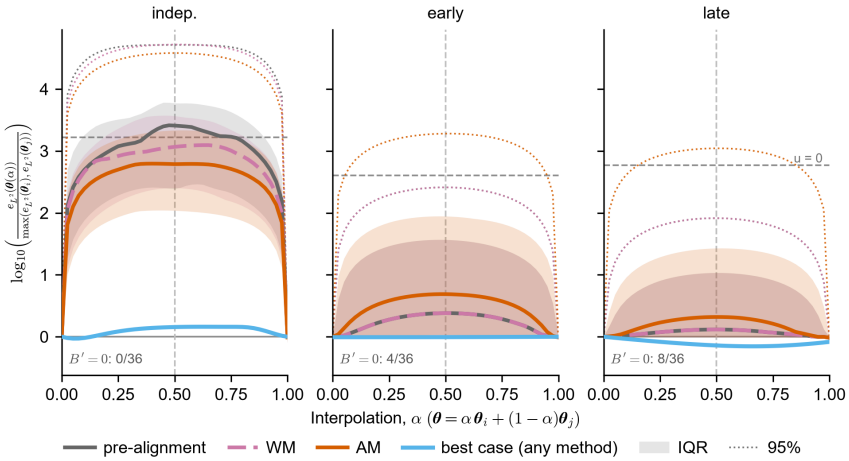


Figure 10: **The same segment, measured against the reference solution.** Relative L^2 error along the segment $[\boldsymbol{\theta}_i, \boldsymbol{\theta}_j]$ for the two top-ranked ingredients of each configuration, normalised by the error of the *worse* endpoint, median with interquartile range and 95% envelope over the 36 configurations of each training regime. Series, reference and best case are those of Figure 4, with $e_{L^2}(\cdot)$ in place of $\mathrm{Score}(\cdot)$. A value of 0, marked by the grey rule, is the threshold $B' = 0$ of equation 25, and the count in each panel gives how many configurations attain it with at least one of the five methods. The dashed horizontal rule is where the trivial predictor $u \equiv 0$ falls, labelled on the rightmost panel; the dashed vertical marks $\alpha = 1/2$, which corresponds to the uniform two-ingredient soup.

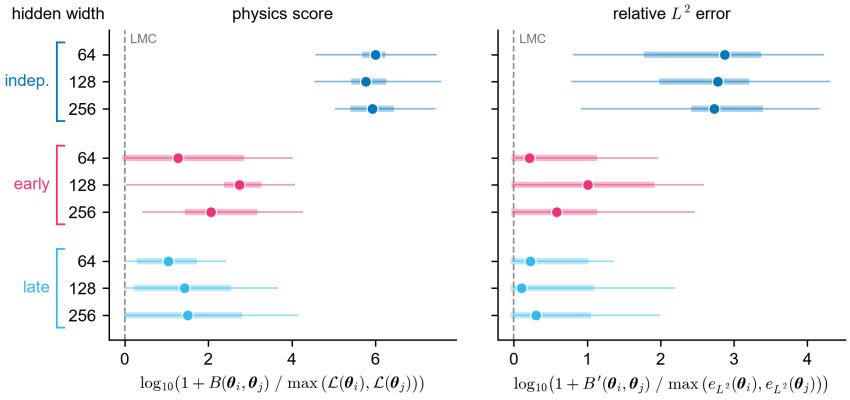


Figure 11: **Width does not flatten the barrier.** Lowest pairwise barrier attained by any of the five parameter-space methods, by training regime and hidden width: median (dot), interquartile range (thick bar) and 5–95% envelope (thin bar) over the twelve problems trained at each width, with B of equation 4 on the left and B' of equation 25 on the right. Zero, marked by the dashed rule, is the threshold of linear mode connectivity, as in Figures 4 and 10; the conjecture predicts the dots move towards it as the network widens, and within each regime they do not approach it. The widths are three settings rather than samples of a continuum, so they are drawn as rows, and are not joined by lines, since three points do not support a trend.

Table 8: **Soups by averaging method.** Uniform two-ingredient soup of the two top-ranked ingredients, on the reporting scale y of Appendix B: median with interquartile range over the 36 configurations of each training regime, and the number n of configurations in which it improves on u_{best} ($y < 0$). The ensemble pools the two ingredients ranked first by the relative L^2 error, since the score is defined on parameters, and is measured against the same u_{best}. Under `early` and `late`, weight matching returns the naive soup, as its recovered transformation is the identity (Table 1).

		`independent`		`early`		`late`	
Method	Matching	Mdn. [IQR]	n	Mdn. [IQR]	n	Mdn. [IQR]	n
`naive`	–	3.42 [2.55, 3.88]	0	0.59 [0.04, 1.57]	4	0.33 [0.01, 1.10]	5
`git-rebasin`	activation	2.83 [2.40, 3.56]	0	0.90 [0.07, 1.84]	4	0.37 [0.00, 1.72]	8
	weight	3.28 [2.64, 3.82]	0	0.59 [0.04, 1.57]	4	0.33 [0.01, 1.10]	5
`sign-rebasin`	activation	2.81 [2.26, 3.44]	0	0.92 [0.06, 1.95]	3	0.37 [0.00, 1.77]	7
	weight	3.10 [2.54, 3.72]	0	0.59 [0.04, 1.57]	4	0.33 [0.01, 1.10]	5
`ensemble`	–	−0.08 [−0.17, 0.00]	29	−0.01 [−0.09, 0.00]	27	0.00 [−0.03, 0.01]	18

Table 9: **The physics score is a usable but noisy proxy.** Agreement between the physics score and the relative L^2 error, as Kendall's τ. Every pair compared lies inside one configuration, the pool one chooses from in practice, and a subset's τ pools its configurations' concordant and discordant pair counts; the CI is a percentile bootstrap over configurations and p a stratified test of no agreement within any of them. The last two columns instead summarise the configurations' own τ.

Subset	Cfg.	Ingr.	τ	95% CI	p	Median τ [IQR]	$p < 0.05$
All	108	864	0.540	[0.476, 0.598]	$< 10^{-10}$	0.571 [0.357, 0.786]	53/108
By tr. regime							
`indep.`	36	288	0.472	[0.353, 0.581]	$< 10^{-10}$	0.571 [0.214, 0.714]	13/36
`early`	36	288	0.536	[0.435, 0.631]	$< 10^{-10}$	0.643 [0.339, 0.732]	19/36
`late`	36	288	0.611	[0.508, 0.706]	$< 10^{-10}$	0.679 [0.482, 0.804]	21/36
By constraint							
`soft`	54	432	0.607	[0.517, 0.689]	$< 10^{-10}$	0.643 [0.500, 0.857]	31/54
`hard`	54	432	0.472	[0.390, 0.549]	$< 10^{-10}$	0.571 [0.232, 0.714]	22/54

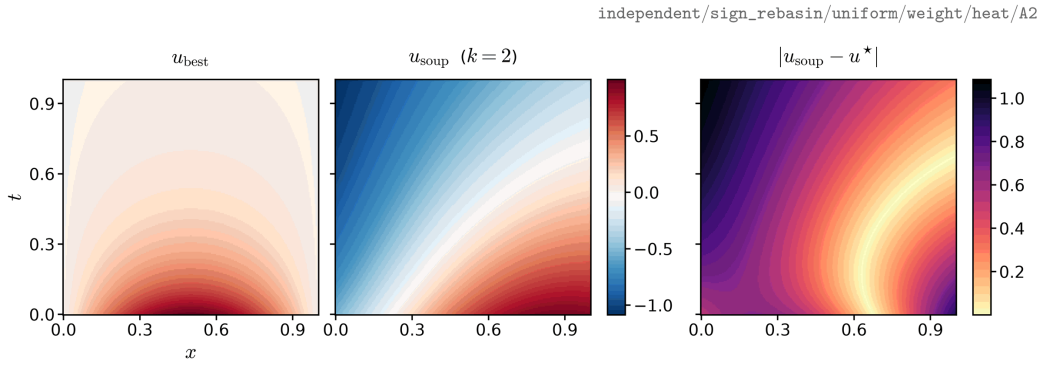
Table 10: **The verdicts under the oracle ranking.** Main-text verdicts with each pool ranked, and greedy acceptance decided, by the relative L^2 error instead of the physics score. Here, the best soup is taken over the five averaging methods, both recipes and $k \leq 8$, and measured against the reference ingredient: the top-ranked one under the score, the best in the pool under the oracle. The best-soup columns count the configurations where it improves on that ingredient ($y < 0$) and where it is a greedy soup, and give its mean y; the median is zero in every row. The greedy columns count the parameter-space runs that keep at least two and three ingredients, and give the most any run keeps.

		Best soup			Greedy runs		
Subset	Ranking	$y < 0$	Mean y	Greedy	$k \geq 2$	$k \geq 3$	Max. k
All	score	15/108	−0.015	0/108	10/540	0/540	2
	oracle	11/108	−0.006	7/108	41/540	8/540	4
By tr. regime							
`independent`	score	0/36	0.000	0/36	0/180	0/180	1
	oracle	0/36	0.000	0/36	0/180	0/180	1
`early`	score	6/36	−0.015	0/36	5/180	0/180	2
	oracle	4/36	−0.007	2/36	12/180	5/180	4
`late`	score	9/36	−0.029	0/36	5/180	0/180	2
	oracle	7/36	−0.011	5/36	29/180	3/180	3

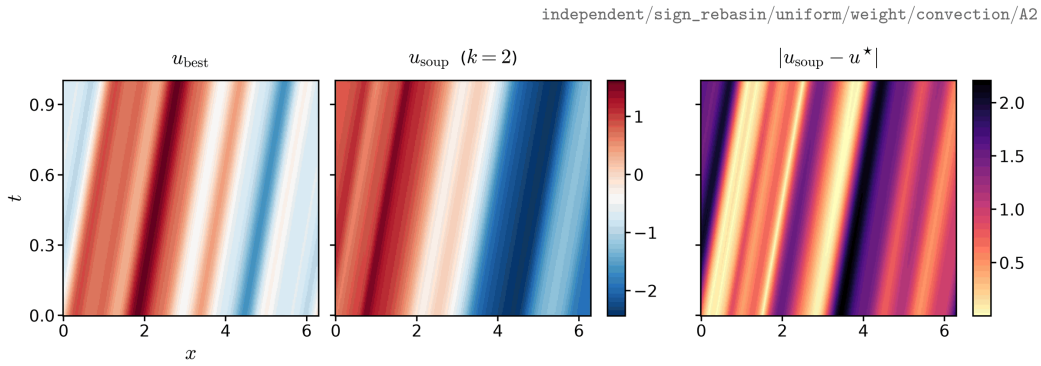
E SOUP DEGRADATION EXAMPLES

The figures below show examples of the soups analysed in the previous sections. Each row is one equation at the reference width, under `independent`, for the uniform two-ingredient soup of the two top-ranked ingredients aligned by Sign Re-Basin with weight matching, which is the pair whose interpolation path Figure 4 sweeps. Panels give the best ingredient solution according to $\mathrm{Score}(\cdot)$, the soup and the pointwise error against the reference. For a fair comparison, every example has been chosen from the sweep axis fixed at the `standard` architecture (width 128).

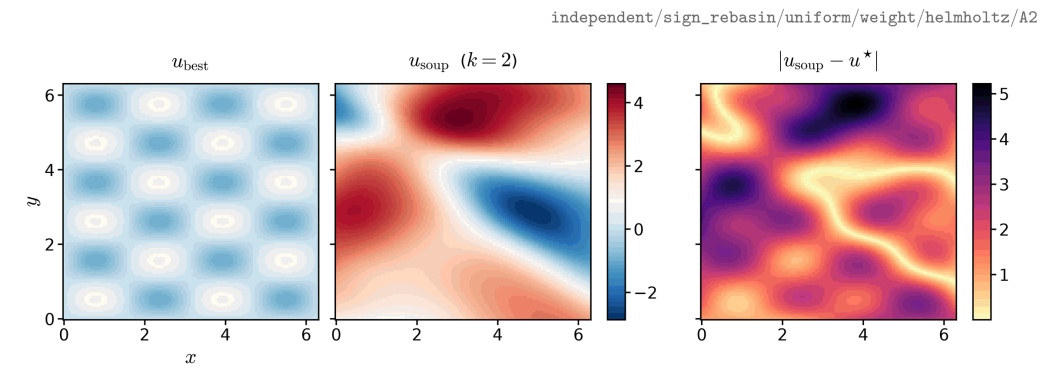
E.1 SOFT-CONSTRAINED PROBLEMS



(a) Heat

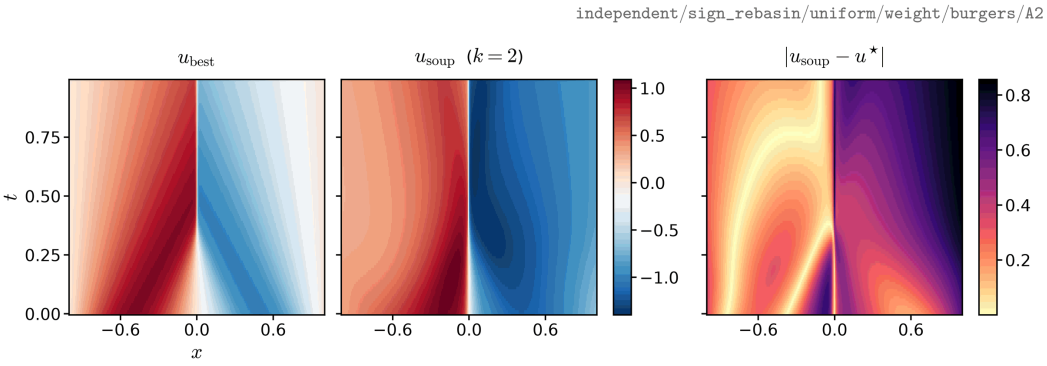


(b) Convection

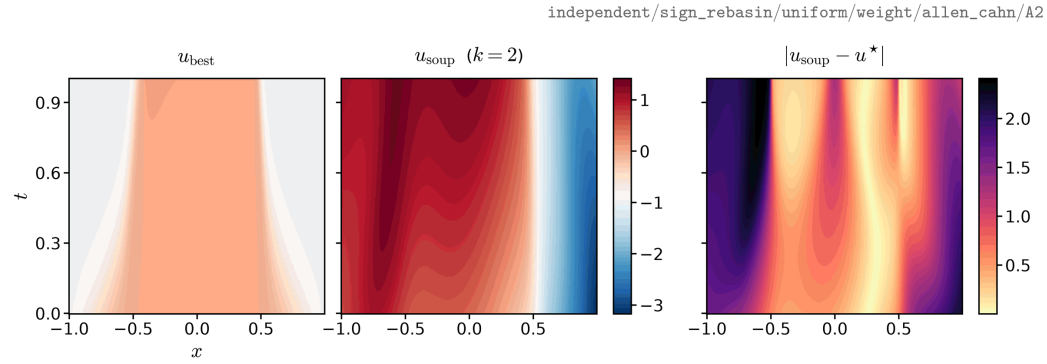


(c) Helmholtz

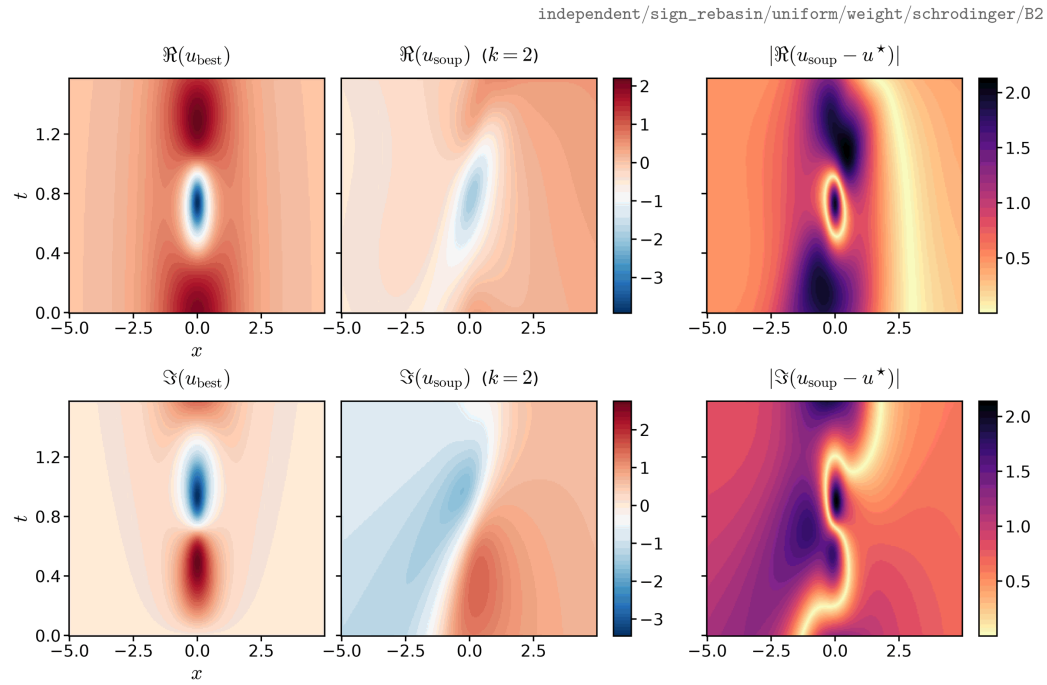
Figure 12: **Two-ingredient soups, soft-constrained linear problems.**



(a) Burgers



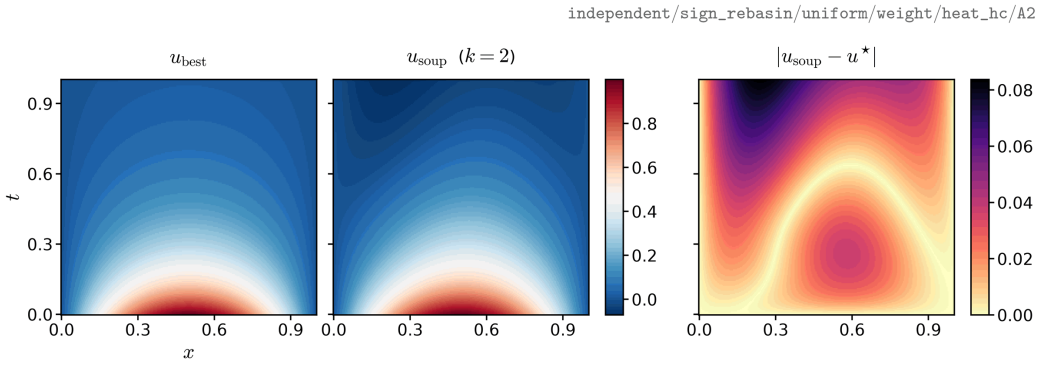
(b) Allen–Cahn



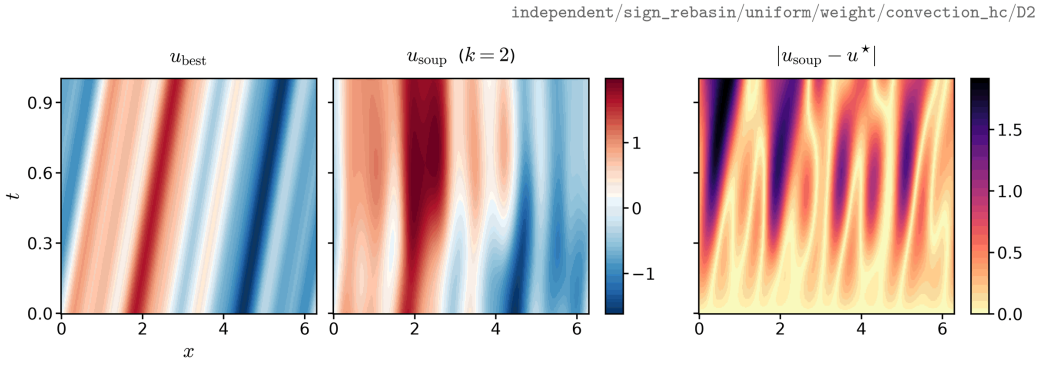
(c) Schrödinger, two components

Figure 13: **Two-ingredient soups, soft-constrained nonlinear problems.**

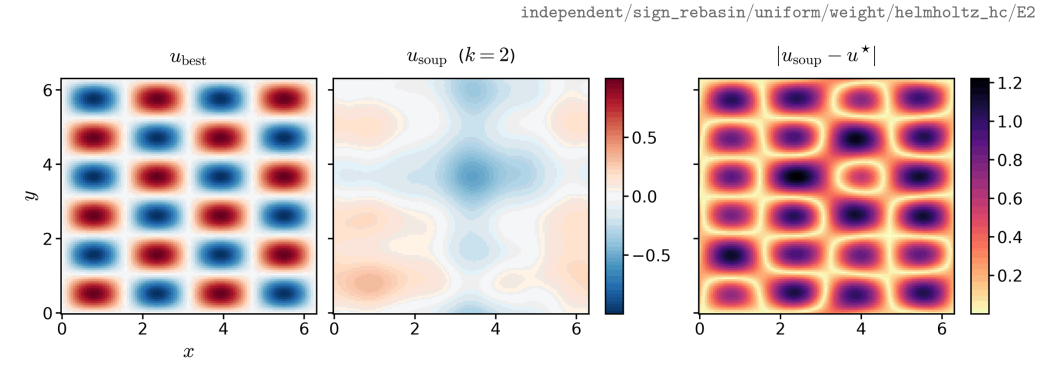
E.2 HARD-CONSTRAINED PROBLEMS



(a) Heat



(b) Convection

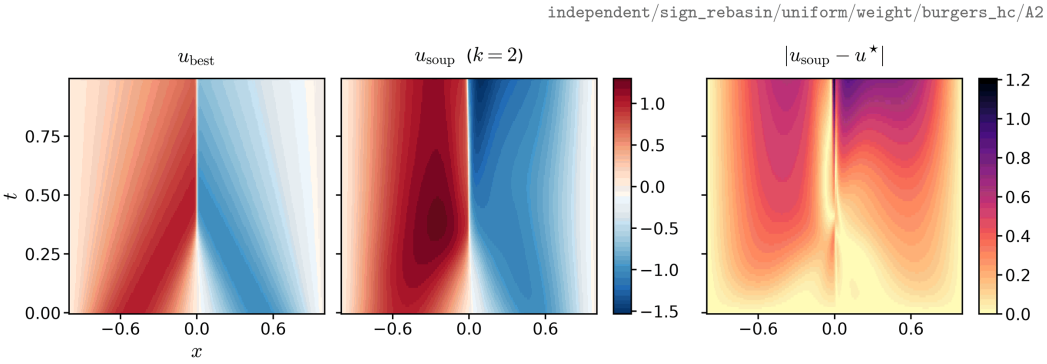


(c) Helmholtz

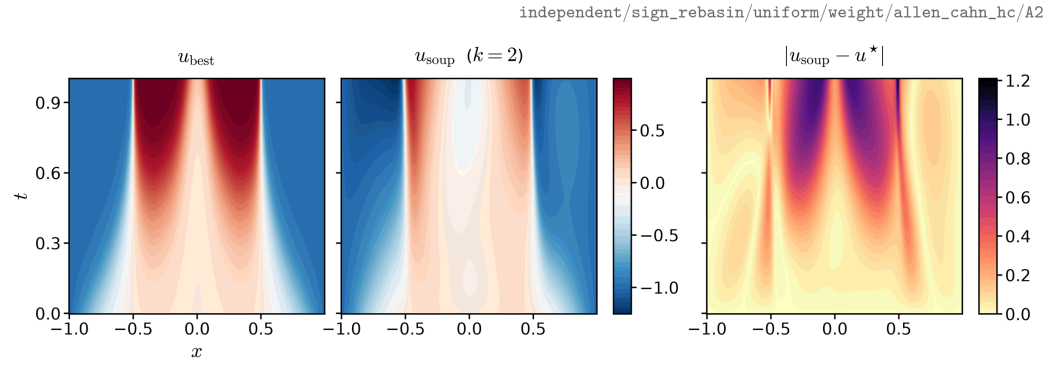
Figure 14: **Two-ingredient soups, hard-constrained linear problems.** The boundary and initial conditions hold by construction, and the soup still misses the solution.

F SOUP IMPROVEMENTS

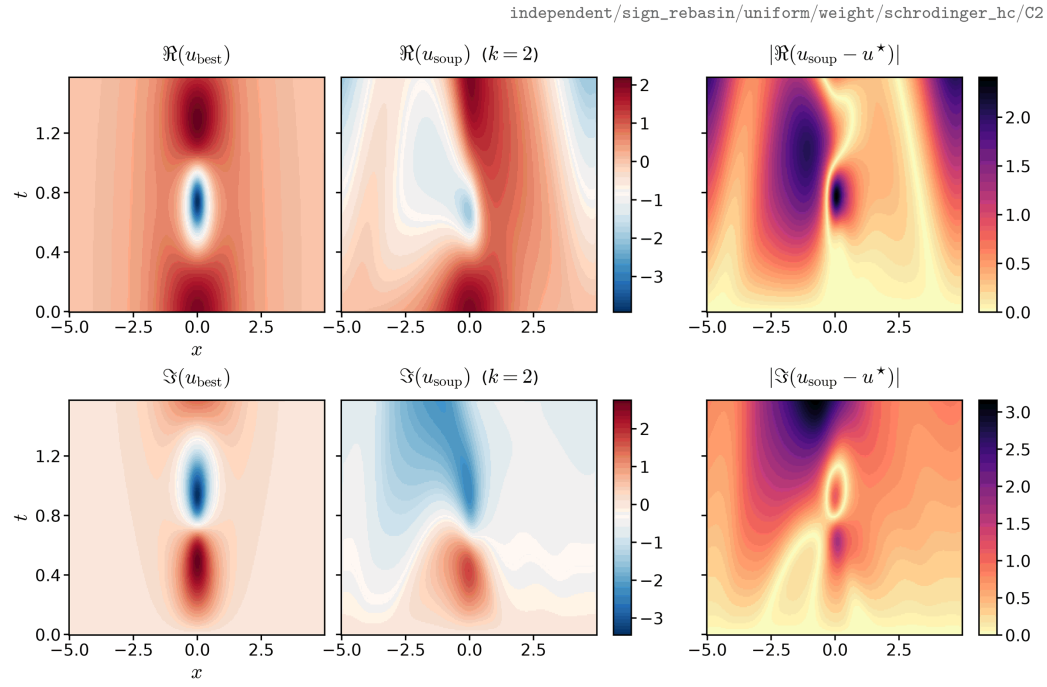
Fifteen of the 108 configurations admit a soup with a lower relative L^2 error than the top-ranked ingredient, i.e. the one selected by the physics score, which is the practical setting, since the relative L^2 error requires a reference solution. This is 13.9% of all configurations, none of which comes from `independent`; within the branching regimes, it is 20.8% of the configurations. Against the oracle-best ingredient the count drops to ten (Appendix D.7). This section presents the figures showing the gain of the soup relative to the top-ranked ingredient for the fifteen configurations. The first two panels show the same quantities as in the previous section; the third shows the gain.



(a) Burgers

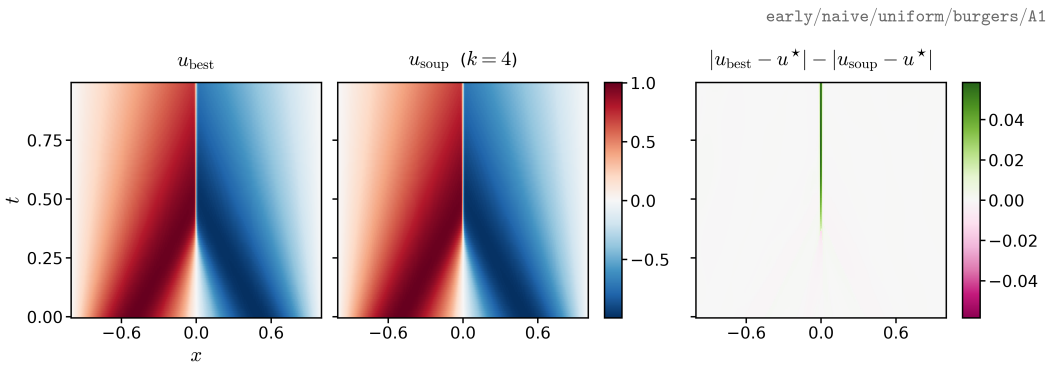


(b) Allen–Cahn

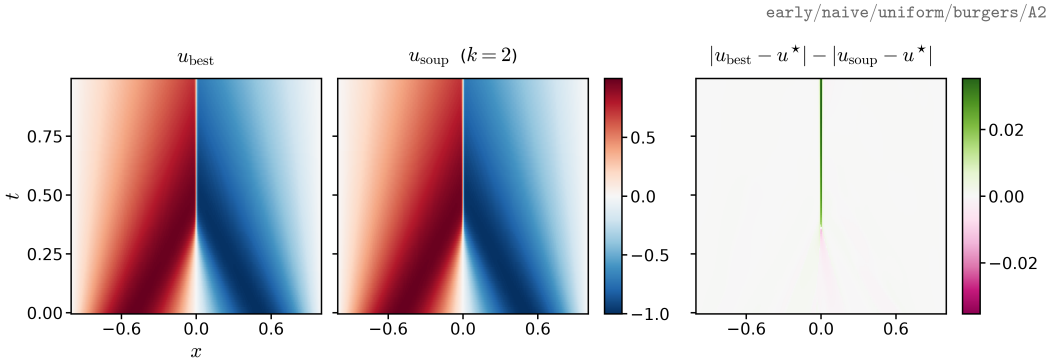


(c) Schrödinger, two components

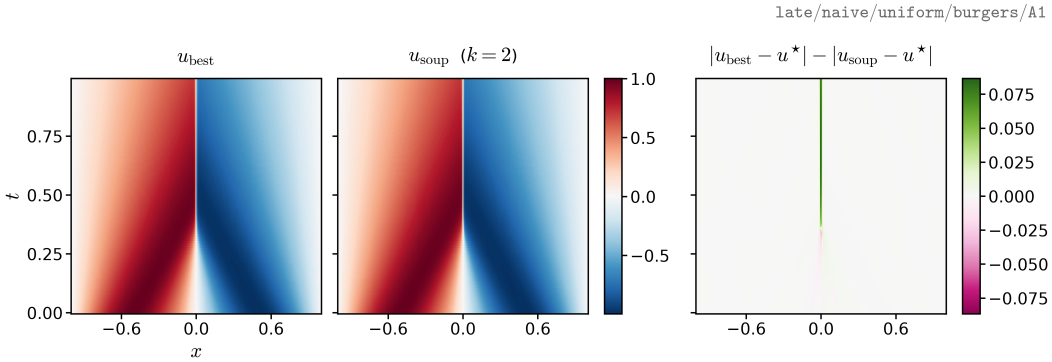
Figure 15: **Two-ingredient soups, hard-constrained nonlinear problems.** The boundary and initial conditions hold by construction, and the soup still misses the solution.



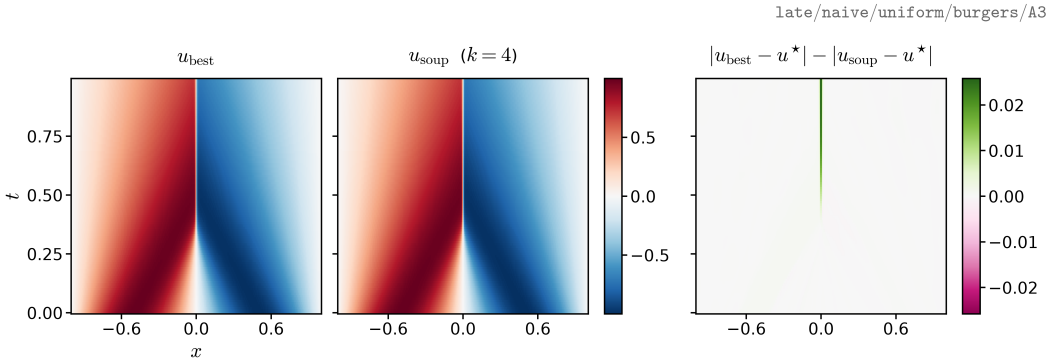
(a) Burgers, width 64, `early`, no alignment, $k = 4$



(b) Burgers, width 128, `early`, no alignment, $k = 2$



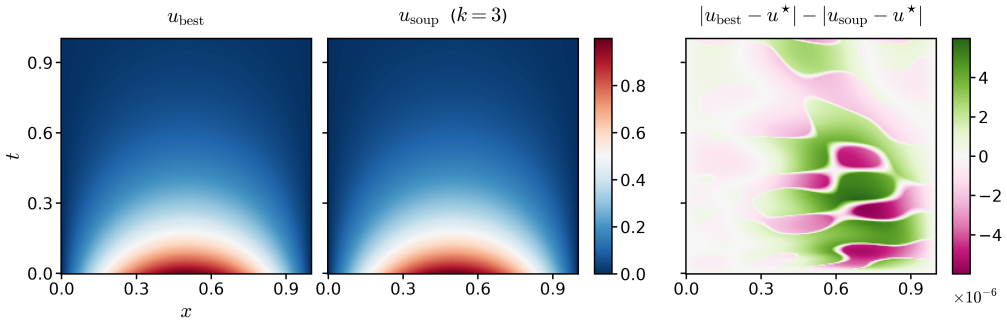
(c) Burgers, width 64, `late`, no alignment, $k = 2$



(d) Burgers, width 256, `late`, no alignment, $k = 4$

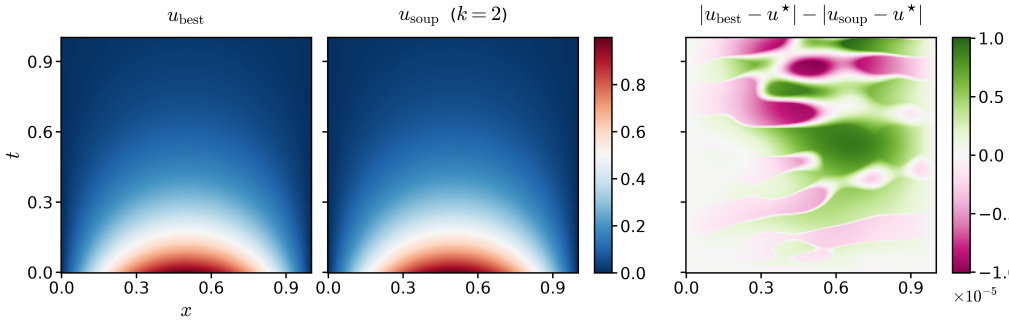
Figure 16: **Soups that improve on the top-ranked ingredient.** Part 1 of 5.

late/naive/uniform/heat_hc/A2



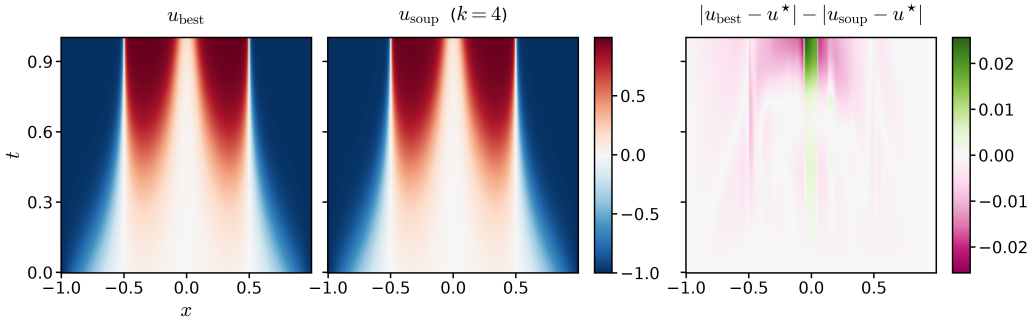
(a) Heat, hard-constrained, width 128, `late`, no alignment, $k = 3$

late/naive/uniform/heat_hc/A3



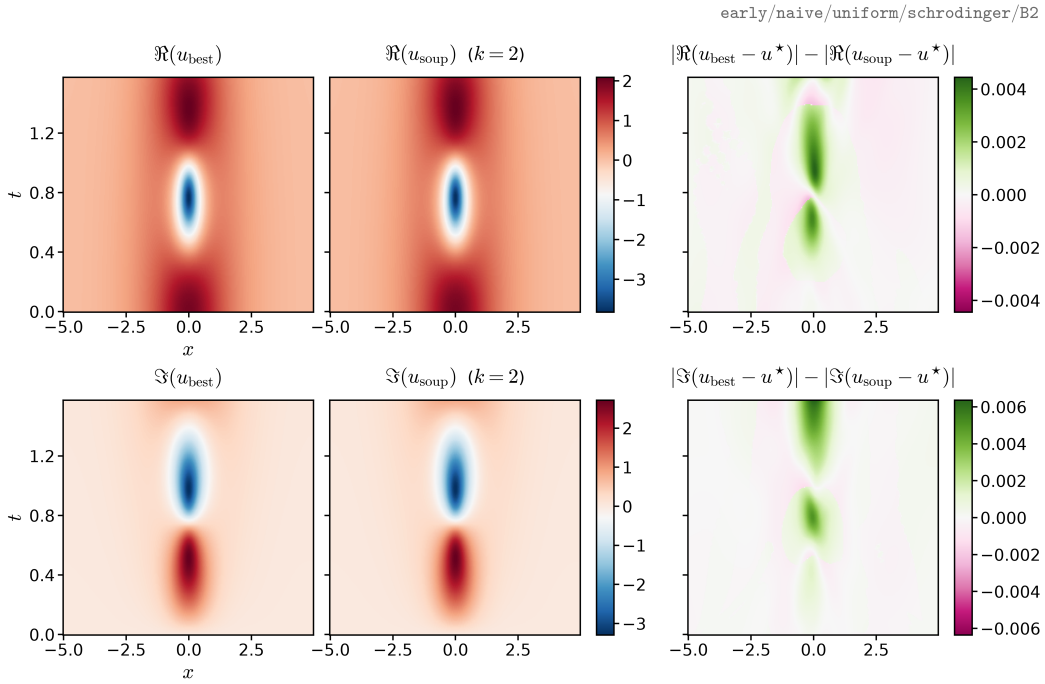
(b) Heat, hard-constrained, width 256, `late`, no alignment, $k = 2$

early/git_rebasin/uniform/activation/allen_cahn_hc/A3

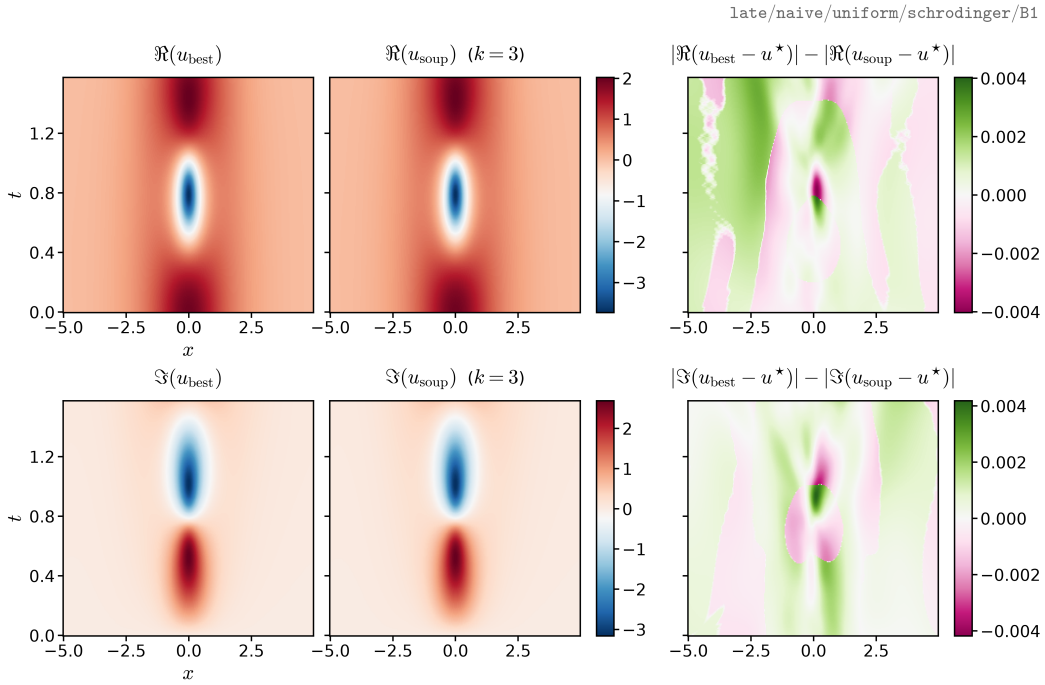


(c) Allen–Cahn, hard-constrained, width 256, `early`, Git Re-Basin with activation matching, $k = 4$

Figure 17: **Soups that improve on the top-ranked ingredient.** Part 2 of 5.

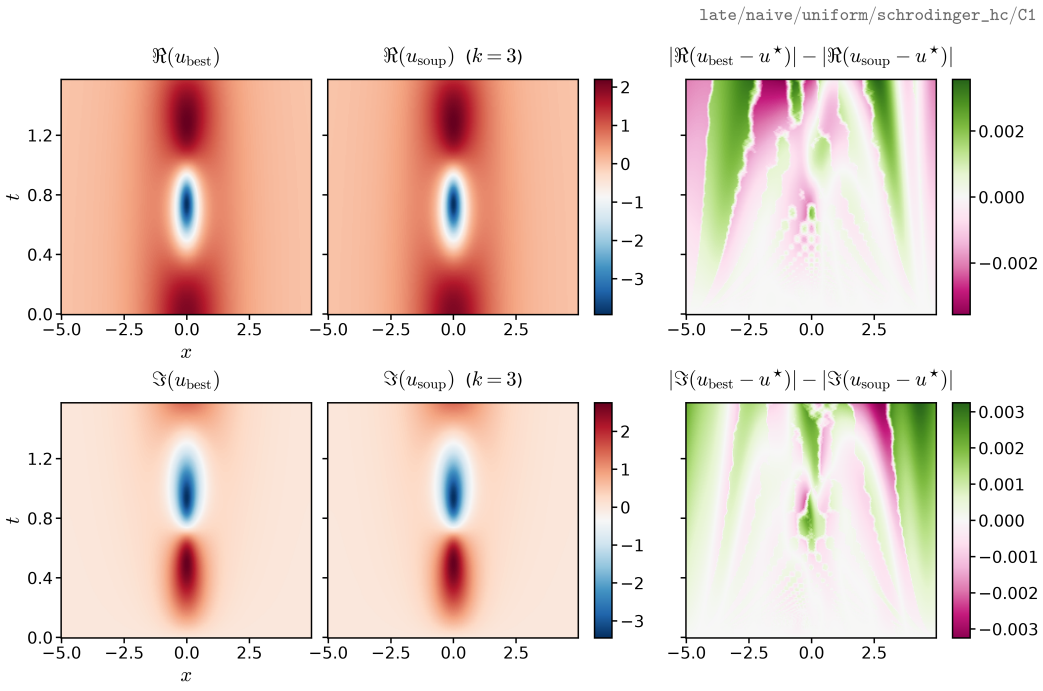


(a) Schrödinger, width 128, `early`, no alignment, $k = 2$

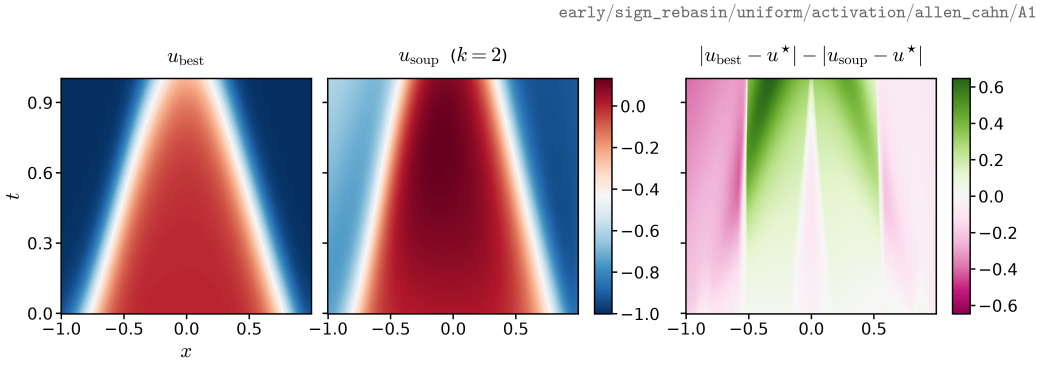


(b) Schrödinger, width 64, `late`, no alignment, $k = 3$

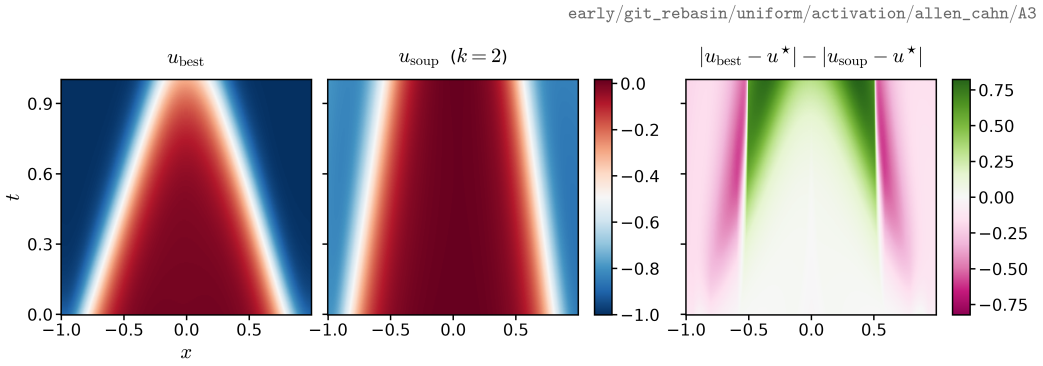
Figure 18: **Soups that improve on the top-ranked ingredient.** Part 3 of 5.



(a) Schrödinger, hard-constrained, width 64, `late`, no alignment, $k = 3$



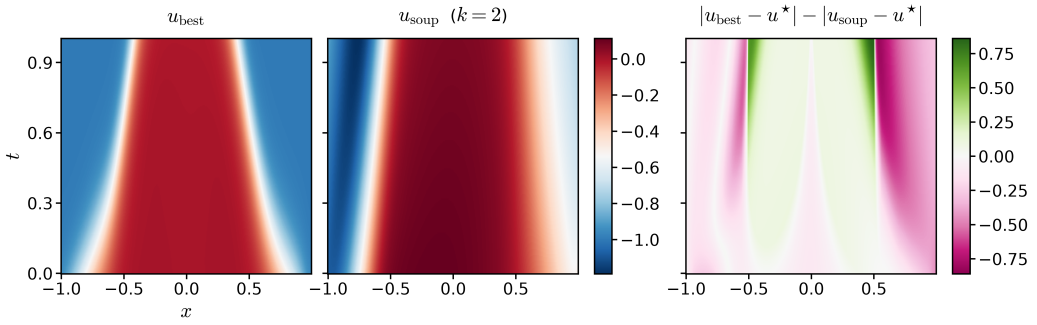
(b) Allen–Cahn, width 64, `early`, Sign Re-Basin with activation matching, $k = 2$



(c) Allen–Cahn, width 256, `early`, Git Re-Basin with activation matching, $k = 2$

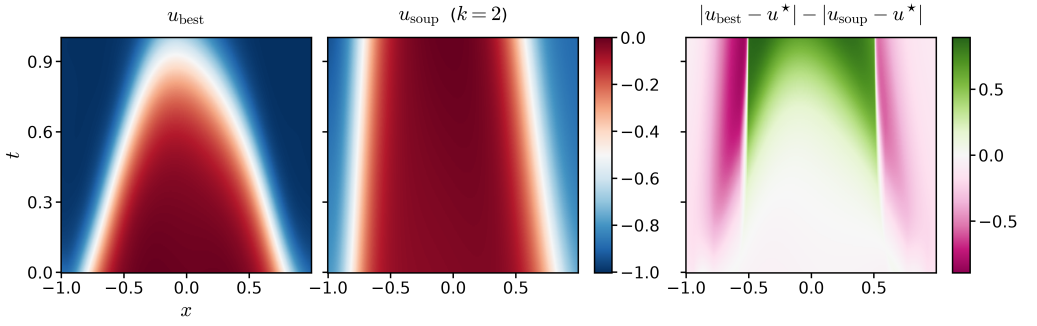
Figure 19: **Soups that improve on the top-ranked ingredient.** Part 4 of 5.

late/git_rebasin/uniform/activation/allen_cahn/A1



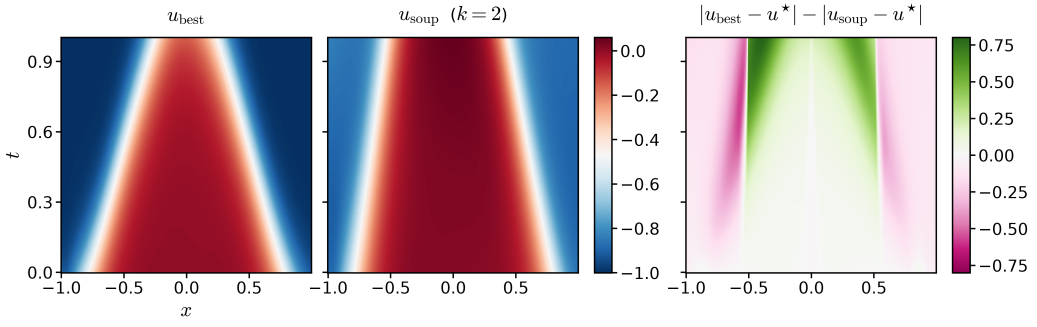
(a) Allen–Cahn, width 64, `late`, Git Re-Basin with activation matching, $k = 2$

late/sign_rebasin/uniform/activation/allen_cahn/A2



(b) Allen–Cahn, width 128, `late`, Sign Re-Basin with activation matching, $k = 2$

late/git_rebasin/uniform/activation/allen_cahn/A3



(c) Allen–Cahn, width 256, `late`, Git Re-Basin with activation matching, $k = 2$

Figure 20: **Soups that improve on the top-ranked ingredient.** Part 5 of 5.