Anchoring on an axis the optimizer does not favor

> Adam scales each coordinate of the residual stream separately, so the coordinate axes are a privileged basis: they are directions the optimizer makes cheap to hold, and e_1, the direction the anchor targets, is one of them. We trained the recipe of record and its control again with Adam's scaling applied in a random rotation of the coordinates, so that e_1 is an ordinary direction. The anchor held on every seed, and the edit removed the anchored op (`difference`) and spilled as little as before. What changed is the control. On some seeds e_1 now overlaps the direction where the un-anchored model puts its syntax tokens, so the edit costs the control something too, and the removal net of the control is low there.

On stored checkpoints, we found that every run parks its separators (`?`, `,` and ⏎) one-hot on a coordinate of their own and `=` on a second, and that a coordinate axis is cheaper to project out than a random direction. In short training runs, Adam built this basis, and Adam in a rotated basis built the same thing on the rotated axes.

The anchoring design had taken e_1 for an arbitrary direction. If it is not, the results that rest on e_1 might lean on it being a cheap axis: the anchor itself, or the comparisons around it, or both. This study trains sixteen new runs, with the basis of the optimizer rotated and nothing else changed: `falloff` (the recipe of record of split-anti-seeds) and the un-anchored control, at the eight seeds where both are stored.

Observations

- Does the anchor hold? (E1): yes, on every seed. The alignment° at the example answers rises with depth as in the stored runs, and the profiles of the new and stored runs lie on top of each other.
- Removal, spill and landing (E2)°: on every seed the edit removes nearly all of the op and spills within the criterion, as before. The removal net of the control falls short on 2 seeds, and all of the shortfall comes from the control: where the rotation puts e_1 near the syntax direction of the control, the edit removes part of the op from the control as well.
- Where the privileged basis went (E3): to the rotated axes. The energy spectrum is flat along the coordinate axes and spiked along one rotated axis, which holds the syntax tokens. In a control, e_1 costs what an ordinary direction costs, and more where it overlaps that spike.

Alignment. How far a state points along e_1…

Spill. The side effect of an edit on the other ops…

Scope and method

This is an exploratory study, with no preregistration and no gate. It covers `falloff` (the recipe of record) and the control at model seeds 704 to 711, trained for 400 epochs with Adam's per-coordinate scaling applied in a random orthogonal rotation of the residual coordinates (one rotation per seed, drawn from the model seed).

Each new run pairs with a stored twin at the same seed (the `falloff` run from split-anti-seeds or the 400-epoch control from ex-2.2.23). The pair shares its initialization, batches and label draws, and differs only in the basis of the optimizer. The model, schedules, anchoring code, measurements and dose scoring are those of split-anti-seeds, unchanged.

A difference between a new run and its twin is compared with the seed range of the stored arm. Split-anti-seeds found `falloff` within 0.002 of `split-anti` on removal, with every seed within the spill° criterion.

Spill. The side effect of an edit on the other ops…

The loss-rate screen put the loss rate near one in forty, so one loss in eight here would be unremarkable. Its upper bound is about one in nine, which leaves two losses in eight inside chance as well. So a change in the loss rate would show only at three or more losses; a smaller effect would show in α° on the pulled slices, or in removal on the seeds that kept the anchor.

Alignment. How far a state points along e_1…

The measurements

- Whether the anchor held: α° at the example answers at the end of training, per slice, measured as in split-anti-seeds. A run that lost the anchor sits near zero on the pulled slices. It counts as lost when the full edit removes less than half of `difference`, net of the control (the rule of split-anti-seeds and the loss-rate screen).
- Removal, spill$^{\circ}$, and landing: the dose scoring of ex-2.2.22 (the projection at every slice and position, at four doses), net of the control at the same seed. Removal is the net drop on `difference` over its clean-to-null span, spill the largest net drop on another op, and landing the fraction of the distance to the target null$^{\circ}$ that the full edit covers. E2 also shows removal without subtracting the control, since the control turns out to matter here.
- The task scores of every op before any edit, compared with those of the control.
- The basis measurement, new here: the share of the energy of the stream[1] that each direction holds, along the coordinate axes and along the rotated basis, and the mean kurtosis[2] along each. It also records the cost of projecting out one direction at full dose, for e_1, each coordinate axis, each rotated axis, and sixteen random directions. The stored runs give the reference, from our check on their checkpoints: one coordinate holds about half of the energy, a typical axis costs 0.03, and a random direction 0.08 to 0.13.

Alignment. How far a state points along e_1…

Spill. The side effect of an edit on the other ops…

Target null. Where a removal should land…

1 Energy…

2 Kurtosis…

Does the anchor hold? (E1)

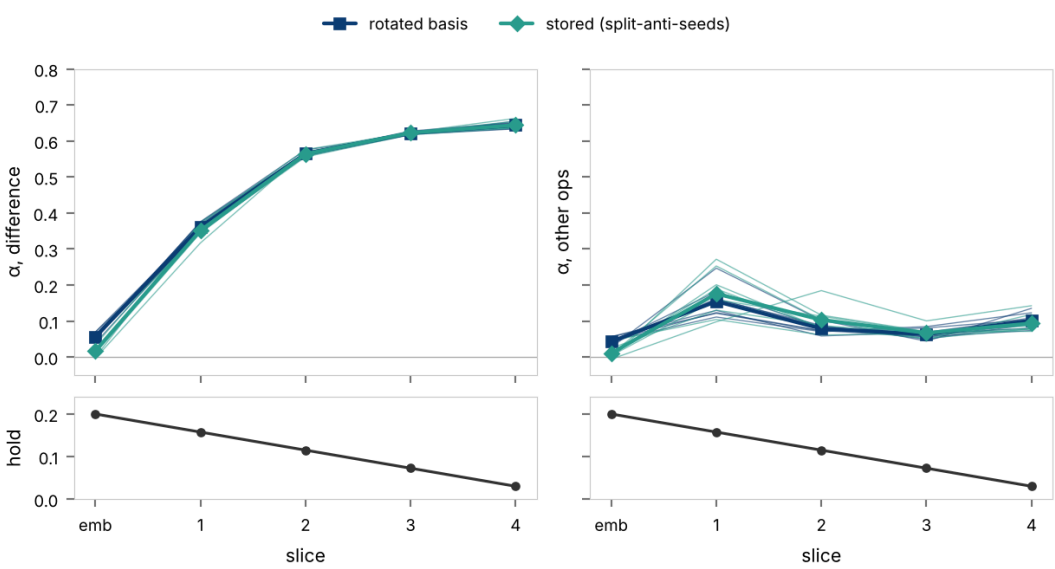
With the basis rotated, is e_1 still held through training at every seed, and as firmly? If the anchor leans on e_1 being a privileged axis, some seeds should lose it, or α° at the example answers should sit lower on the pulled slices than in the stored `falloff` runs. If not, the per-slice α profiles should lie within the stored seed range.

Alignment. How far a state points along e_1…

A second explanation needs keeping apart. In the stored runs the syntax spike sits on a coordinate axis, orthogonal to e_1 by construction; under the rotation it sits on a rotated axis whose cosine with e_1 is a few tenths at most, and known per seed.

So an anchor that is harder to hold could mean that the anchor needs a privileged axis, or that the anti term now has to push the syntax spike off e_1. E3 reports that cosine per seed, to show how much of any seed-to-seed difference it accounts for.

What we saw. The anchor held on every seed. The alignment profile of each new run lies on that of its stored twin at every slice after the embedding (the largest gap on any seed and slice is 0.04), and the eight new profiles span the same range the eight stored ones do.



***Alignment at the example answers by slice**, at the end of training, on the probe contexts of `difference` (left) and of the other ops (right): the new runs and their stored twins. One hairline per run and the seed mean bold. Underneath is the hold of the anti weight at each slice, the same in both.*

The one visible difference is at the embedding, where the new runs sit at about 0.06 against 0.02 in the stored runs. The anti weight is highest there, and under the rotation the syntax tokens no longer have a coordinate of their own, orthogonal to e_1 for free. So the few hundredths may be what it costs the anti term to keep them off e_1.

But the same uplift shows in the contexts of the other ops, which fits another explanation at least as well: under the rotation e_1 is a generic direction at the embedding, so every embedding has a small share of it. Past the embedding, the alignment in those contexts is unchanged.

So the anchor does not need e_1 to be a privileged axis: the pull finds and holds an ordinary direction as well as it held an axis.

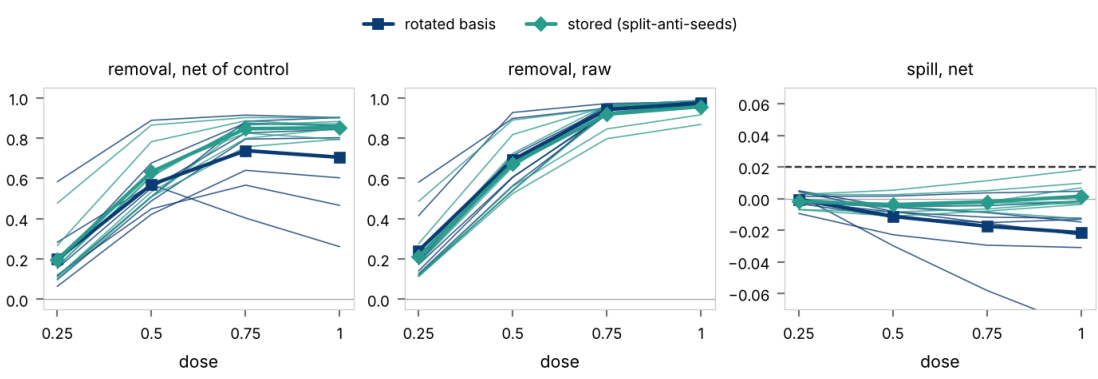
Removal, spill and landing (E2)

On the seeds that kept the anchor, does the edit remove `difference` as it did, and does it spill° as little? The stored `falloff` runs removed four fifths to nine tenths of the op at full dose with every seed within the spill criterion of 0.02. The comparison is paired by seed, against the new control at the same seed.

Spill. The side effect of an edit on the other ops...

What we saw. Two things happened at once, and the table separates them. First, the anchored runs themselves remove the op as they did: before subtracting the control, the drop on `difference` under the full edit is 0.98 (0.97 to 0.99) of the clean-to-null span on the new runs and 0.96 (0.87 to 0.99) on the stored ones.

Every new run spills within the criterion (the largest net spill over the doses is 0.000 (−0.009 to 0.005), and the raw spill at the full dose, 0.017 (−0.002 to 0.038), matches what the stored runs show). The landing is 0.50 (0.42 to 0.54) against 0.52 (0.49 to 0.56) stored, and the task change is within the seed range (−0.001 (−0.013 to 0.006)).



***Removal and spill by dose**, the edit at every position and slice: removal net of the control (left), removal without subtracting the control (middle), and spill (right). One hairline per run and the seed mean bold; the stored `falloff` runs beside the new ones. The dashed rule is the selectivity criterion.*

Second, the removal net of the control (the measurement the earlier reports quote) falls short on 2 seeds by the kept/lost rule, and is lower on a third. The shortfall sits in the control. In the stored runs the control loses about a tenth of the op when e_1 is projected out, because e_1 there is an ordinary coordinate axis that holds little.

Under the rotation e_1 is an ordinary direction instead, which is not orthogonal to the syntax spike of the model. On seeds 706 and 710 the cosine between e_1 and the spike direction of the control is about a quarter, and the control loses half or more of the op to the edit on its own. The shortfall in net removal tracks that cosine across the eight seeds (Pearson r = 0.94).

seed	raw removal, rotated ↑	raw removal, stored ↑	control drop, rotated	control drop, stored	control, other ops, rotated	control, other ops, stored	net removal, rotated ↑	net removal, stored ↑	spill, rotated ↓	raw spill, rotated ↓	raw spill, stored ↓	landing, rotated ↑	task change, rotated ↑	cos(e_1, control spike)
704	0.97	0.97	0.07	0.09	0.08	0.03	0.90	0.88	−0.002	0.025	0.024	0.42	−0.002	0.05
705	0.97	0.99	0.12	0.19	0.06	0.10	0.84	0.79	0.005	0.038	0.038	0.47	−0.013	0.03
706	0.99	0.92	0.52	0.07	0.18	0.03	0.47	0.85	0.002	0.022	0.025	0.54	+0.006	0.26
707	0.97	0.98	0.36	0.11	0.18	0.03	0.60	0.87	0.004	0.011	0.020	0.52	+0.001	0.11
708	0.97	0.87	0.17	0.07	0.09	0.07	0.80	0.79	−0.003	0.007	0.011	0.54	+0.001	0.02
709	0.97	0.99	0.11	0.12	0.06	0.05	0.86	0.86	−0.009	−0.002	0.006	0.53	−0.002	0.06
710	0.98	0.97	0.72	0.07	0.44	0.02	0.26	0.90	0.005	0.025	0.010	0.47	+0.005	0.25
711	0.98	0.97	0.08	0.11	0.05	0.04	0.90	0.86	−0.002	0.008	0.003	0.53	−0.004	0.04

***Removal, spill, landing and task by seed**, at the full dose. The raw removal is the anchored run's own drop on `difference` as a share of its clean-to-null span; the control drop is the same for the un-anchored control at that seed, under the same edit, and "control, other ops" is the largest drop the same edit causes the control on any other op (the spill of the control, before any netting); the net removal is their difference, the usual measurement. "Stored" is the twin from split-anti-seeds and ex-2.2.23. Spill is the largest net drop on another op over the four doses, the raw spill the largest drop on another op at the full dose without subtracting the control, and the task change is the mean score over ops less that of the control. The last column is the cosine between e_1 and the rotated basis vector that holds the syntax tokens of the control (E3). Arrows mark which direction is better.*

So the kept/lost rule, applied as written, would count 2 of 8 new runs as having lost the anchor. That is inside chance by the reckoning in Scope, and the raw removal and the alignment° both say that no run lost the anchor (0 are below half by the raw drop).

Alignment. How far a state points along e_1...

The drops on the other ops show the same thing from the other side. On seeds 706 and 710 the control loses on every op, up to 0.31 (0.18 to 0.44) in score on an op other than `difference`, while the anchored run at the same seeds loses 0.024 (0.022 to 0.025) at most on any other op. The stored controls lose at most 0.10 on any other op.

So the edit on the rotated control takes syntax, spread over the whole task, while the edit on the anchored run takes the anchored concept and little else. Netting against the control assumes the edit on the control touches nothing much, which held when e_1 was a quiet axis and does not hold for a direction that overlaps the spike of the control.

At those seeds the anchored run holds its own syntax spike on a rotated axis away from e_1 (E3), which fits the anti term steering it there. So e_1 is clean in the anchored run where it is not in the control.

Where the privileged basis went (E3)

Did the rotation do what it was meant to? If so, the energy spectrum should be flat along the coordinate axes and spiked along the rotated axes, with the syntax tokens on a rotated axis. And e_1 should cost what an ordinary direction costs in the stored runs (0.08 to 0.13 in a control, where it touches the spikes), not what an ordinary axis costs (0.03).

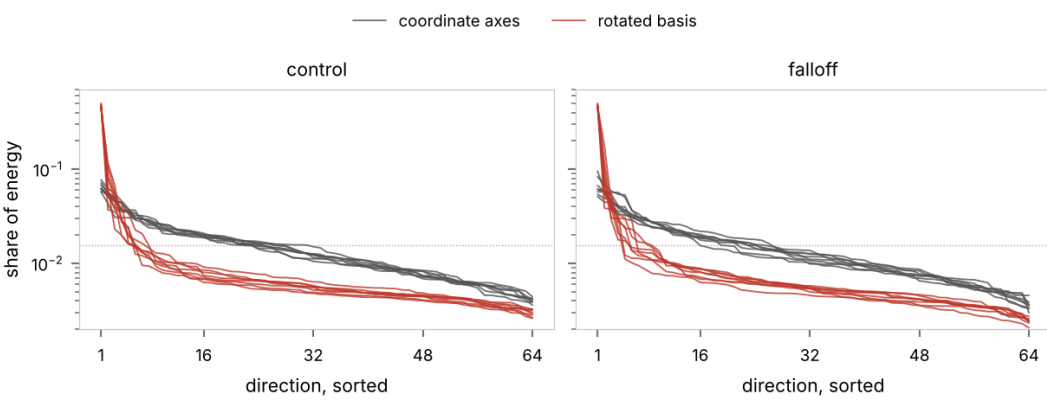
In the anchored runs e_1 holds the anchored op, so its cost there is the removal plus the spill°, and it is compared with e_1 in the stored `falloff` runs. The cleaner comparison is within a run (e_1 against the random directions of that run and its rotated axes off the spikes), since the stored reference numbers are means over draws that touch the spikes by varying amounts.

Spill. The side effect of an edit on the other ops…

The basis measurement also records where each token embedding sits (its largest cosine with a coordinate axis and with a rotated axis), so the published data show whether the syntax tokens moved to a rotated axis.

What we saw. The privileged basis moved with the optimizer. Along the rotated basis one direction holds 0.48 (0.46 to 0.50) of the energy of the stream at the last slice, as the one coordinate did in the stored runs. Along the coordinate axes the largest share is 0.07 (0.05 to 0.10), and the spectrum is much flatter, though it still spans about an order of magnitude from the largest direction to the smallest.

Kurtosis runs the reverse of the stored runs: near the Gaussian value of 3 along the axes (3.2 (2.9 to 3.4) in the controls), and above it along the rotated basis (4.2 (4.0 to 4.4)).



***The energy spectrum of the stream at the last slice**, along the coordinate axes (grey) and along the rotated basis (red), each sorted from the largest, one hairline per seed, on a log scale. A share of a direction is the mean square, over positions, of the component of the state along it; the 64 shares along a basis sum to one, and the dotted rule is one 64th.*

The syntax tokens sit on that rotated spike: the `?`, `,`, ⏎ and `=` embeddings have a cosine of 0.88 (0.74 to 0.99) with a rotated basis vector, and on most seeds all four share the vector that holds the spike (the table). Against the coordinate axes their largest cosine is about a third, what a random unit vector in 64 dimensions gives.

On 3 of 8 seeds (704, 706, 710) the spike of the anchored run sits on a different rotated axis from that of the control, and on the others on the same one. In the stored runs we found the spike on the same coordinate in the control and the anchored run at every seed we checked. So a move here fits the anti term steering the choice, but the two runs also differ in their loss terms, and could settle differently for that reason alone.

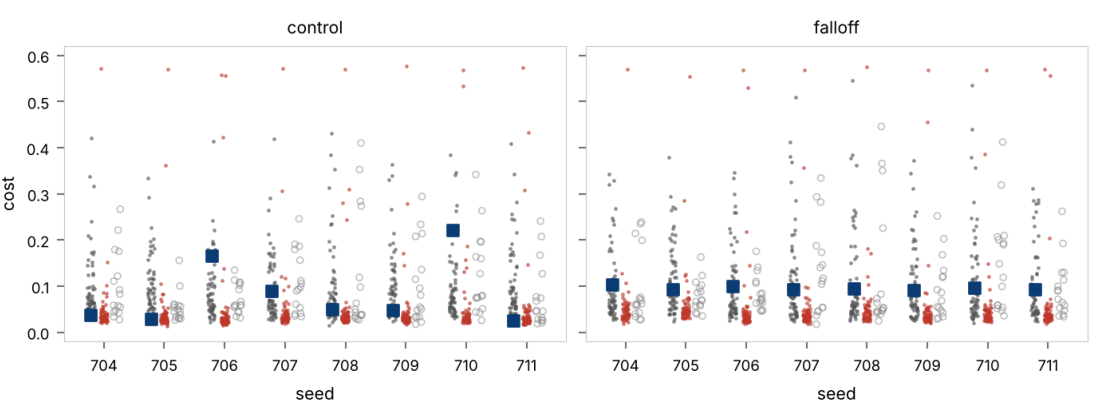
> **A later finding**
>
> The stored runs move their spike too. Scoring every coordinate axis of the stored `falloff` runs and controls, reference-baseline (E1) found the spike of `falloff` on a different coordinate from the control on 3 of 8 seeds, under plain Adam. So a move under the rotation is no sign of the anti term in particular: a change of loss alone may be enough.

seed	spike share, control	top axis share, control	syntax on a rotated axis, control	cos(e_1, spike), control	spike share, `falloff`	top axis share, `falloff`	syntax on a rotated axis, `falloff`	cos(e_1, spike), `falloff`	spike axis, `falloff`	
704	0.50	0.06	0.86, 0.78, 0.83, 0.98	0.05	same	0.48	0.06	0.82, 0.77, 0.78, 0.94	0.04	moved
705	0.46	0.06	0.87, 0.85, 0.91, 0.98	0.03	same	0.48	0.05	0.82, 0.75, 0.82, 0.92	0.03	same
706	0.47	0.06	0.96, 0.88, 0.97, 0.93	0.26	same	0.49	0.06	0.98, 0.85, 0.97, 0.96	0.06	moved
707	0.48	0.07	0.93, 0.86, 0.92, 0.98	0.11	same	0.46	0.10	0.90, 0.76, 0.79, 0.98	0.11	same
708	0.46	0.08	0.97, 0.88, 0.92, 0.93	0.02	same	0.47	0.08	0.88, 0.75, 0.88, 0.92	0.02	same
709	0.47	0.06	0.93, 0.90, 0.92, 0.99	0.06	same	0.48	0.06	0.85, 0.74, 0.82, 0.96	0.06	same
710	0.47	0.07	0.99, 0.82, 0.97, 0.97	0.25	same	0.48	0.05	0.90, 0.76, 0.78, 0.80	0.01	moved
711	0.49	0.06	0.96, 0.86, 0.91, 0.84	0.04	same	0.50	0.07	0.87, 0.89, 0.90, 0.80	0.04	same

***Where the privileged basis went**, per seed. The spike share is the largest share of energy held by any rotated basis vector at the last slice, and the top axis share the largest held by any coordinate axis. The syntax columns give the largest cosine between each syntax embedding (`?`, `,`, ⏎, `=`) and any rotated basis vector; under Adam in the coordinate basis these are 0.86 to 1.0 against a coordinate axis (the stored runs). The cosine columns give the cosine between e_1 and the spike direction of that run, and the last column says whether `falloff` holds its spike on the same rotated basis vector as the control at that seed.*

e_1 now costs what a direction costs, plus the cost of its overlap with the spike. In the controls, the 63 other coordinate axes cost 0.07 at the median, about what a random direction costs in the stored controls, and a rotated axis off the spike costs 0.03, what an ordinary axis costs in the stored runs. So the cheap directions are now the rotated axes.

In the controls, e_1 costs 0.05 (0.03 to 0.09) on the six seeds where its cosine with the spike is about a tenth or less, and 0.19 (0.17 to 0.22) on the two where it is about a quarter, the overlap E2 found in the net removal. In the `falloff` runs e_1 costs 0.10 (0.09 to 0.10) on every seed: the removal of the anchored op, which the cost spreads over the ops, with nothing of the spike in it.



***The cost of projecting out one direction**, at full dose at every slice and position, as the mean drop in the task score over the ops; one column per seed, the control on the left and `falloff` on the right. Grey dots are the 63 coordinate axes other than e_1, red the 64 rotated axes, and the open circles sixteen random directions; the square is e_1.*

Discussion

The anchor did not need a privileged axis. Pulled toward an ordinary direction, `falloff` aligned, removed and stayed selective as it did on the coordinate axis on all eight seeds, apart from a few hundredths of alignment° at the embedding. So the assumption of the design, that e_1 stands in for any direction, holds for the anchor itself. The privileged basis changes how we should interpret the comparisons around the anchor, not what the anchor does.

Alignment. How far a state points along e_1…

What the rotation did change is the control, which the removal and spill° measurements are netted against. When e_1 was an axis, the syntax tokens of the control sat on another axis, orthogonal to e_1 for free (in the anchored runs of earlier designs the anchor pulled a syntax token onto e_1 instead, the latch of spill-by-position). When e_1 is a direction, it overlaps the spike of the control by a random cosine, and on two of eight seeds that was enough for the edit to take half the op from the control, so the net measurement counts a loss the anchored run never had. Where the edit damages every op of the control, as on those two seeds, subtracting it no longer isolates the anchored op.

Spill. The side effect of an edit on the other ops…

So the baseline needs choosing with the basis in mind, whatever the optimizer. One that does not depend on where e_1 happens to sit is the median drop over edits along several other directions, like the reference edit of the spill-timing experiment: coordinate axes off the two syntax coordinates under Adam, and any directions under an optimizer with no privileged basis. Raw removal beside it shows what the anchored run itself gives up.

A reader of an earlier report need not revise anything, since every stored comparison kept e_1 as an axis in both runs. But the same logic applies to reference directions: a reference direction compared with e_1 should be another coordinate axis off the two syntax coordinates, since a random direction overstates the baseline by whatever it takes from the spike.

The anchored runs also show the anti term at work in a way the stored runs could not. With the coordinates privileged, keeping the syntax off e_1 cost nothing, because the model parked it on a different coordinate anyway. Here, on 3 seeds, the anchored run holds its syntax on a different rotated axis from the one the control chose.

That fits the anti term steering the choice of the model among the cheap directions of the optimizer. But one of the three is a seed where e_1 overlaps nothing, and the two runs may have settled differently for reasons of their own.

If this holds, the anti term would have a larger job still in a model with no privileged basis at all, since there would be no cheap direction to steer the syntax onto. The optimizer scout shows what the stream of such a model looks like without an anchor.

---

1. Energy: the sum of the squared projections of the states onto a direction. The share of a direction is its energy over the total. ↩

2. Kurtosis: how heavy the tails of a distribution are. A Gaussian scores 3; a direction where most values are small and a few are large scores higher. ↩