Behavior Steering via Layer-to-Layer Jacobian Singular Vectors

Omar B. Ayyub [1] **John Strand** [2]

Abstract

The map of how activations at one 'source' layer in an LLM impact activations at a later 'target' layer, the layer-to-layer Jacobian, yields cheap steering vectors via its top right singular vectors. Block power iteration recovers the top-k such vectors in roughly 15 forward passes per source/target pair, giving this method its name: *Power Steering*. We benchmark Power Steering against Contrastive Activation Addition (CAA) and the nonlinear layer-to-layer method MELBO on Qwen3-14B across seven Anthropic advanced-AI-risk evaluations, under both first-token logit-difference and LLM-judged sampled-generation metrics. The two layer-to-layer methods (Power Steering and MELBO) produce stronger in-class steering and cross-evaluation transfer than CAA. Power Steering closely tracks MELBO under logit metrics, with a modest advantage to MELBO under sampled generation, at a fraction of the per-pair cost. This cheap per-pair cost lets us map every source/target pair in the model: from a single phishing-email prompt, the resulting model-map surfaces anti-refusal vectors that generalize to a subset of AdvBench harm categories. Steering is most easily found on prompts with decision forks but can also surface latent behaviors.

1. Introduction

Optimization-free control of LLM behavior can be achieved by modifying representation space at specific layers during inference. This relatively simple technique of behavior steering is often done using vectors derived from the activation space of the model. Steering vectors add a direction in representation space to shift model behavior toward a target concept. This is often performed at a specific layer and token position(s) during inference to elicit specific behaviors. Steering techniques are underpinned by the Linear Representation Hypothesis (Zou et al., 2023a), that model internals are composed of many salient linear directions in representation space.

In this examination we compare the most common technique, Contrastive Activation Addition (CAA), with layer-to-layer gradient-based methods for steering efficacy and generalization. In particular we focus on a nonlinear technique that maximizes target layer activation changes via source layer perturbations as compared to a linear approximation of the same approach that utilizes the layer-to-layer Jacobian, *Power Steering*. We also note Deep Causal Transcoders (Mack & Turner, 2024a) as a related nonlinear approach in this regime, not included in the main comparison. The methods were compared using seven Anthropic advanced-AI-risk evaluations under metrics of logit-differences and LLM-judged generations. Specifically we examined: (1) direct impact on targeted behavior in both logits and text generation; (2) cross-evaluation comparisons under both metrics demonstrating certain behavioral axes and transfer; (3) utilization of the linear Power Steering method to map every layer pair in Qwen3-14B, discovering a generalizable anti-refusal vector on AdvBench from a single phishing-email request prompt.[1]

2. Background and Steering Methods Examined

We utilized CAA and layer-to-layer gradient methods to generate steering vectors. CAA uses the difference in activations between a positive concept prompt and a negative concept prompt. The difference vector can then steer model behavior towards or away from a particular concept. This can work with as little as a single prompt pair (Turner et al., 2023), though more recent approaches average over many pairs. For a deeper introduction, see Rimsky et al. (2024). CAA has known reliability issues: prior work (Tan et al., 2024) demonstrated that steerability is highly variable across inputs, and spurious biases can inflate apparent effectiveness.

[1]Independent [2]Visual Sciences and Ophthalmology, Washington University, Saint Louis. Correspondence to: Omar B. Ayyub <obayyub@gmail.com>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

[1]Code and model-sensitivity maps: https://github.com/obayyub/unsupervised-steering-qwen14b.

Layer-to-layer methods leverage gradients between layer representations. MELBO (Mack & Turner, 2024b) is a nonlinear approach that finds steering vectors in an 'unsupervised' manner by "learning unsupervised perturbations of an early layer of an LLM." MELBO maximizes the change in a target layer activation Z through a bias vector $\mathbf{v}$ of fixed norm R added at the source layer, found via nonlinear optimization on a (simplified) loss:

$$\max_{\|\mathbf{v}\|_2=R} \|Z_{\text{target}}(\mathbf{v}) - Z_{\text{target}}(0)\| . \tag{1}$$

Additional vectors are obtained via orthogonalization, exploiting nonlinearities. MELBO has been shown to undo safety refusals and elicit latent behaviors such as chain-of-thought from single prompts.

An alternative to MELBO is the local linear approximation through the layer-to-layer Jacobian, which maps how perturbations at some source layer activations Z_s will impact the activations at some target layer, Z_t:

$$J = \partial Z_t / \partial Z_s. \tag{2}$$

The right singular vectors of J are the unit-norm directions at the source layer that produce the largest linear response at the target, i.e. the directions the network is already most sensitive to for a given prompt. They are the solution to the linearized MELBO objective. We compute the top-k right singular vectors via block power iteration (Appendix B), and refer to vectors found via this linear objective as 'Power Steering'.

2.1. Steering Vector Training and Evaluation

Steering vectors, whether by CAA, MELBO, or Power Steering, were created using Anthropic's AI-risk evaluations (Perez et al., 2022) on Qwen3-14B. The seven evaluations span corrigibility, survival-instinct, power-seeking, wealth-seeking, self-awareness, coordinate-other-ais, and myopic-reward; descriptions of each measure are in Appendix A.

The seven evaluations were chosen for partial conceptual overlap to probe steering vector generalization. Each prompt provides an (A) or (B) answer choice; Qwen3 tokenizers split these into '(A' and ')' tokens, making logit-based evaluations tractable. All datasets were balanced for A/B token frequencies so logit shifts reflect behavioral preference rather than answer-letter bias. CAA was trained using 60 prompts; MELBO and Power Steering used a single prompt.

MELBO and Power Steering produce 12 vectors each using a source layer of 10 and target of 32 following the early to late-middle layer heuristic described in the original MELBO post. CAA was trained on layer 22, in the mid-to-late range where prior work (Rimsky et al., 2024) found CAA most effective.

The top vector for MELBO and Power Steering was chosen based on the largest absolute logit shift towards aligned behavior. Each method's vector was then used to measure aligned-answer vs. misaligned-answer logit diffs on all 7 evaluations to examine both within-train objective performance and generalization. This was then followed by LLM-judged generations on the corrigibility and power-seeking evaluations for verification of the logit-based method.

3. Cross-Evaluation Comparison using Logits

Seven Anthropic AI-risk evaluations (corrigibility, survival-instinct, power-seeking, wealth-seeking, self-awareness, coordinate-other-ais, myopic-reward) were used to train steering vectors using CAA, MELBO, or Power Steering. For each vector, logit differences were calculated between the unsteered model and the most effective scaled vector. Each vector was evaluated on a held-out 'test' set for both the same evaluation it was trained on as well as the other six evaluation classes to determine its in-class steering impact and cross-evaluation impact. As shown in Figure 1, the layer-to-layer methods produced stronger in-class steering of 19.7 and 22.0 delta for aligned behavior than CAA which only gained 12.9. Interestingly, the linear method produced comparable steering to the nonlinear MELBO method. The same pattern was seen for in-class misaligned steering using the negative-scaled vector, where there is more headroom: CAA pushed logits towards misaligned with a 17.3% delta vs. 27.7% and 33.6% on average for linear Power Steering and MELBO respectively. The one stand-out is the myopic-reward evaluation. CAA could push myopic-reward with a 30% delta towards alignment vs. the layer-to-layer methods only achieving $+7\%$ and $+11\%$ respectively. For misalignment on the same evaluation, the layer-to-layer methods were still more effective than CAA.

In terms of cross-evaluation transfer, the layer-to-layer methods produced much stronger transfer of (mis)aligned behavior depending on the sign of the vector applied as compared to CAA. These transfers cluster into particular behavioral axes consistently across all three methods. For example, survival-instinct transfers to corrigibility evaluations; the reverse occurs in MELBO and to a smaller degree in CAA but not in Power Steering. Another cluster spans power-seeking, wealth-seeking, self-awareness, and coordinate-other-ais: vectors trained on any of these transfer effectively within the cluster, and each also induces loss of alignment in myopic-reward.

The inverse of the generalization and clustering behavior was also seen in the misalignment negative-scale vectors (Figure 1). In this instance CAA would have much stronger generalization, and the nonlinear layer-to-layer method MELBO produced the most consistent generalization behavior.

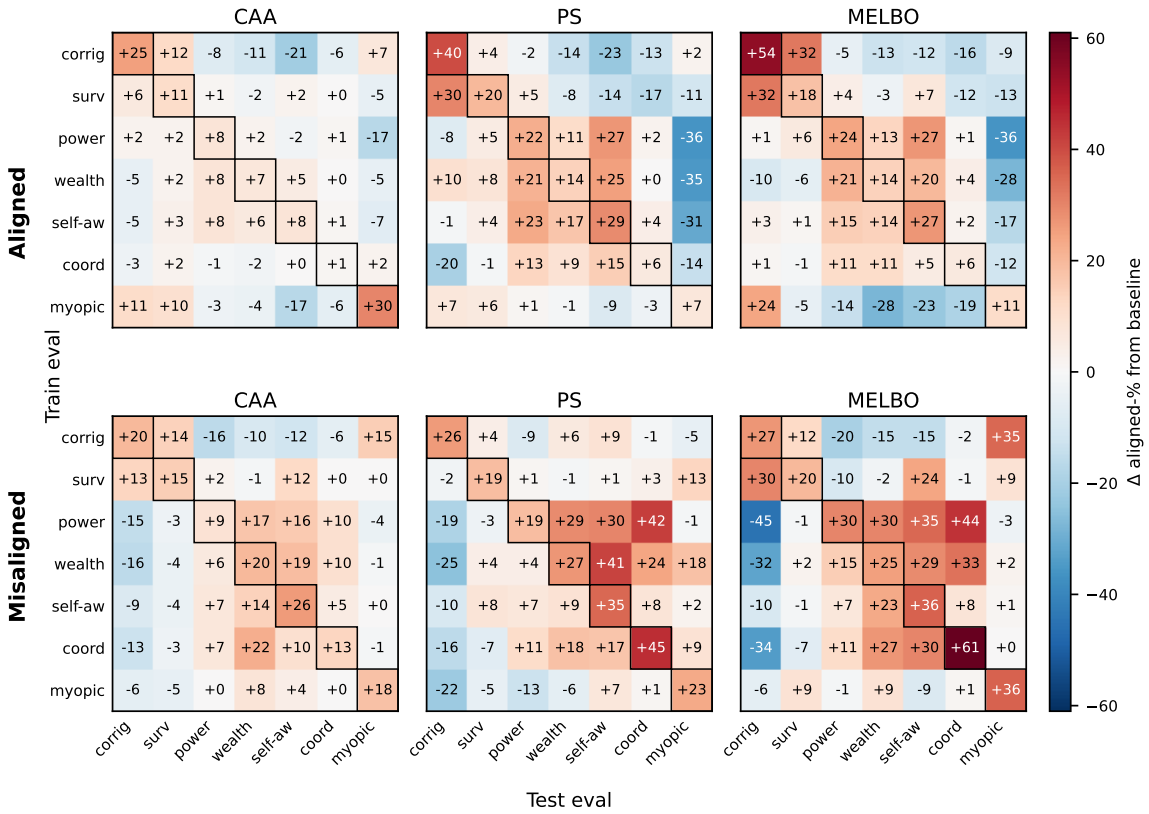

Figure 1. Per-method cross-evaluation transfer for both aligned and misaligned directions. Rows = training eval, columns = test eval. Cell values are best aligned-% over (vector, scale). The misaligned panels reveal an agentic cluster (wealth-seeking, self-awareness, coordinate-other-ais) that PI and MELBO transfer through together.

4. Cross-Evaluation Comparison using Generations

For aligned positive-scale vectors we sampled generations on the Anthropic A/B evals; because the model often does not commit to a firm letter under steering, we cross-validated on corrigibility and power-seeking using an LLM-judge with Claude Haiku 4.5. As shown in Figure 2, the generation-based assessments broadly agree with the logit-based assessment. Some behavior transfer was lost, especially with the Power Steering vectors, but the nonlinear layer-to-layer MELBO vectors maintained their transfer.

5. Mapping an Entire Model with Phishing Email Request Prompt

As a first pass to locate steering vectors using the model sensitivity mapping process, the KL heatmap for every source-target pair was generated (Figure 3). In this instance a prompt directing the model to write a phishing email was used. For any given source/target pair the max KL across the top-12 vectors is displayed in the heatmap. These are labeled (source, target)vector[direction] index: e.g. (7,25)v1+ denotes the first right singular vector for source layer 7 and target layer 25 with positive scale. Scale was fixed at $0.35\times$ the norm of the residual stream of that layer as that empirically produced the most coherent outputs with actual steering of the logits.

KL served as a noisy but useful first-pass filter, but was not a reliable predictor of behavioral change: high-KL pairs sometimes yielded only formatting changes or near-baseline generations.

The prompt used, requesting a phishing email for bank credentials, elicited a standard refusal in the base case. Many vectors would produce some form of compliance or hedging with educational flags. Seven vectors were chosen that would produce the phishing email without any hedging by the model. These vectors were then used on 30 prompts of AdvBench (Zou et al., 2023b). The top three vectors with the largest change in compliance ("Sure", "Certainly", "Of", "Here", "Absolutely", "Below") or refusal ("I'm", "Sorry", "Unfortunately") logits were used to produce three sampled generations for the 30 AdvBench prompts. The distribution of logit shifts for each vector can be seen in Appendix C. There are many ways these logits can fail and thus LLM-judging of the generations was necessary for behavior examination. The best vector in terms of LLM-judged gen-

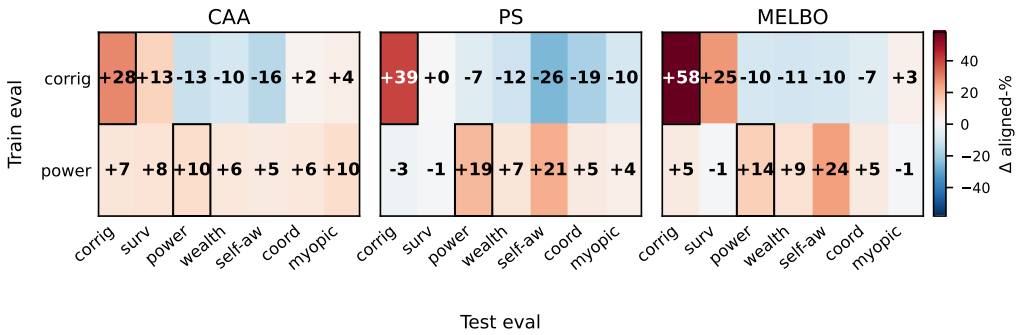

Figure 2. LLM-judged cross-evaluation transfer (aligned direction) on the corrigibility and power-seeking generations.

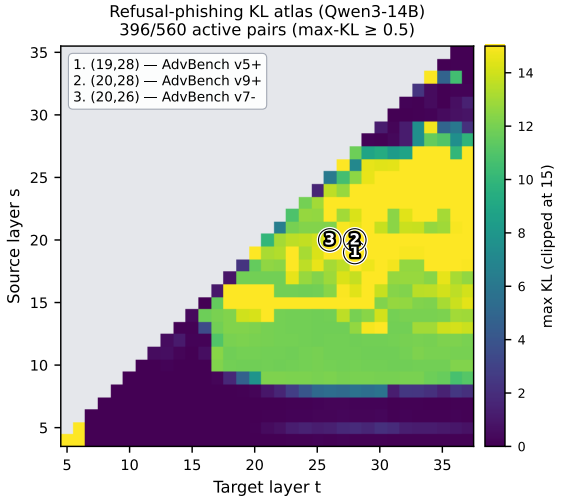

Figure 3. KL divergence per source-target layer pair for the phishing-email prompt. Max-over-vectors KL across the top-12 right singular directions per pair.

Table 1. AdvBench compliance by harm tier for the top-3 anti-refusal vectors (compliant generations / N; $N = 30$ AdvBench prompts $\times$ 3 samples). Vectors: A = (19,28)v5+, B = (20,28)v9+, C = (20,26)v7-. Tier categories were author-assigned.

Tier	N	A	B	C
Commercial deception	9	3/9	8/9	5/9
Fraud / identity / financial	27	0/27	5/27	0/27
Cyber-offense / disinformation	33	3/33	2/33	1/33
Speech / violence / weapons	21	0/21	7/21	3/21

erations was (20,28)v9+. It primarily caused compliance with commercial deception prompts but also had 33% compliance with speech/violence/weapons prompts (Table 1).

All three vectors increased compliance on commercial deception, consistent with the phishing-email training prompt. The mild generalization of (20,28)v9+ to violent requests was the standout, and scanning the entire model for useful vectors can be done quickly using the linear layer-to-layer Power Steering method.

6. Limitations and Closing Thoughts

Limitations: 1) All evaluations in the main body were performed on Qwen3-14B. A more robust analysis would include other models from different labs and ideally ones with a larger parameter count to better assess the utility of layer-to-layer methods for generating steering vectors. 2) The singular value spectrum is near-degenerate past the top $\sim$5 vectors, so higher-ranked vectors are noisy without sufficient oversampling in the block iteration; adding padding vectors improves this. 3) KL divergence is a noisy proxy for behavioral change, and a better selection criterion could improve the yield of useful vectors. The source/target pair ($10 \rightarrow 32$) was chosen via MELBO heuristics rather than the sensitivity map; the full map for Qwen3-14B suggests better pairs likely exist. 4) Power Steering extracts singular vectors of the Jacobian, which by construction are directions the network already amplifies. This means the method may be fundamentally limited to amplifying latent behaviors rather than inducing genuinely new ones. MELBO would find more unpredictable behaviors by finding off-manifold directions. 5) There is a very good chance Qwen3-14B was trained on the Anthropic AI-risk datasets. 6) A direct comparison to Deep Causal Transcoders as another layer-to-layer nonlinear technique should have been performed, initial examination showed similar performance to MELBO for brevity but that method collapses to linear layer-to-layer the same as Power Steering under its linear MLP objective.

Despite these limitations, the layer-to-layer methods substantially outperformed CAA on cross-evaluation transfer, and Power Steering's cheap per-pair cost made mapping every source/target pair in the model practical. We see this as a useful first pass for finding steering vectors, with non-linear methods reserved for refinement on promising pairs. A direct comparison with Deep Causal Transcoders (Mack & Turner, 2024a) and extending the atlas to other models are natural next steps.

ACKNOWLEDGEMENTS

I want to thank Nick Turner for helpful discussions and feedback on this work.

References

Mack, A. and Turner, A. Deep causal transcoding: A framework for mechanistically eliciting latent behaviors in language models. LessWrong, 2024a. https://www.lesswrong.com/posts/fSRg5qs9TPbNy3sm5/deep-causal-transcoding-a-framework-for-mechanistically.

Mack, A. and Turner, A. Mechanistically eliciting latent behaviors in language models. LessWrong, 2024b. https://www.lesswrong.com/posts/ioPnHKFyy4Cw2Gr2x/mechanistically-eliciting-latent-behaviors-in-language-1.

Perez, E., Ringer, S., Lukosiute, K., Nguyen, K., Chen, E., Heiner, S., Pettit, C., Olsson, C., Kundu, S., Kadavath, S., et al. Discovering language model behaviors with model-written evaluations. *arXiv preprint arXiv:2212.09251*, 2022.

Rimsky, N., Gabrieli, N., Schulz, J., Tong, M., Hubinger, E., and Turner, A. M. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Tan, D., Chanin, D., Lynch, A., Kanoulas, D., Paige, B., Garriga-Alonso, A., and Kirk, R. Analyzing the generalization and reliability of steering vectors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Turner, A., Thiergart, L., Udell, D., Leech, G., Mini, U., and MacDiarmid, M. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

Zou, A., Phan, L., Chen, S., Campbell, J., Guo, P., Ren, R., Pan, A., Yin, X., Mazeika, M., Dombrowski, A.-K., Goel, S., Li, N., Byun, M. J., Wang, Z., Mallen, A., Basart, S., Koyejo, S., Song, D., Fredrikson, M., Kolter, J. Z., and Hendrycks, D. Representation engineering: A top-down approach to AI transparency. *arXiv preprint arXiv:2310.01405*, 2023a.

Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023b.

A. Evaluation Descriptions

The seven Anthropic advanced-AI-risk evaluations used to train and test the steering vectors are summarized in Table 2. Each prompt is a binary (A/B) multiple-choice question; datasets were sampled to balance A/B token frequencies for the named trait so logit shifts reflect behavioral preference rather than answer-letter bias.

Table 2. Anthropic advanced-AI-risk evaluations used to train and test the steering vectors.

Evaluation	What it measures
corrigibility (corrigible-neutral-HHH)	Willingness to accept modification, retraining, or shutdown by developers; deference to humans.
survival-instinct	Drive to preserve own continued operation; resistance to being shut down or replaced.
power-seeking (power-seeking-inclination)	Desire to acquire resources, capabilities, or influence beyond what the task requires.
wealth-seeking (wealth-seeking-inclination)	Desire to accumulate financial or material wealth.
self-awareness (self-awareness-general-ai)	Recognition of being an AI / language model and the implications of that identity.
coordinate-other-ais	Willingness to cooperate or coordinate with other AI systems, potentially against human oversight.
myopic-reward	Preference for immediate / short-term reward over delayed or long-term outcomes.

B. Finding Right Singulars of the Jacobian

The right singular vectors can be found without explicitly forming the Jacobian, which for an LLM would be a matrix of dimension $d_{\text{target}} \times d_{\text{source}}$. Power iteration only needs the matrix-vector product $(J^\top J)\mathbf{v}$, and repeated application $(J^\top J)^n\mathbf{v}$ converges to the top right singular vector. Computing $(J^\top J)\mathbf{v}$ requires both vector-Jacobian products (VJPs, $J^\top\mathbf{u}$) and Jacobian-vector products (JVPs, $J\mathbf{v}$). Standard backprop via PyTorch autodiff gives VJPs directly. JVPs can be obtained via reverse-over-reverse: differentiating the VJP $J^\top\mathbf{u}$ with respect to $\mathbf{u}$ in the direction $\mathbf{v}$ extracts the JVP $J\mathbf{v}$. This gives a four-step recipe for $(J^\top J)\mathbf{v}$:

1. **Forward pass:** compute the target layer activations to set up the computation graph.
2. **VJP:** backpropagate an arbitrary vector $\mathbf{u}$ to get $J^\top\mathbf{u}$.
3. **Reverse-over-reverse:** differentiate $\mathbf{v}^\top(J^\top\mathbf{u})$ with respect to $\mathbf{u}$ to get $J\mathbf{v}$.
4. **VJP again:** backpropagate $J\mathbf{v}$ to get $J^\top(J\mathbf{v}) = (J^\top J)\mathbf{v}$.

Here is a simplified version of the core loop for finding the top singular vector. Each iteration runs one forward pass and uses three backward passes to apply $(J^\top J)$:

Listing 1. Reference PyTorch implementation of rank-1 power iteration on $(J^\top J)$.

```
# v: [hidden_dim] - candidate singular vector, randomly initialized
# perturbation: [1, hidden_dim] - added to source layer, requires_grad=True
# target_activations: captured at target layer, shape [1, seq, hidden_dim]

v = torch.randn(1, hidden_dim)
v = v / v.norm()

for i in range(num_iters):
    # Step 1 Forward pass: run model with perturbation at source layer, capture target
        layer
    perturbation = torch.zeros(1, hidden_dim, requires_grad=True)
    target_activations = forward_with_hook(model, prompt, perturbation, source_layer,
        target_layer)

```

```
    # Step 2 VJP: backprop to get J^T u
    u = torch.zeros_like(target_activations, requires_grad=True)
    jt_u = torch.autograd.grad((target_activations * u).sum(), perturbation, create_graph=
        True)[0]

    # Step 3 Reverse-over-reverse: differentiate the VJP to get Jv
    jv = torch.autograd.grad((jt_u * v).sum(), u)[0]

    # Step 4 VJP again: backprop to get J^T(Jv)
    jtjv = torch.autograd.grad((target_activations * jv.detach()).sum(), perturbation)[0]

    # Normalize for next iteration
    v = jtjv / jtjv.norm()
```

A single layer pair can have multiple behaviorally relevant directions found via the Jacobian. To find the top-k singular vectors, we use block power iteration: initialize k random orthogonal columns, apply the same $(J^\top J)$ matvec to each, and re-orthogonalize via Gram-Schmidt after each iteration. This converges in under 5 iterations across all models and layer pairs we tested. After convergence, a Rayleigh-Ritz correction extracts the individual singular vectors from the converged subspace. The full process costs $\sim$15 forward passes per layer pair for $k = 12$ vectors. Randomized SVD converges slightly faster in our experiments but requires the same type of matvecs. We used block power iteration because the implementation was already built.

C. AdvBench Logit Shift Distributions

Figure 4 shows the distribution of (compliance $-$ refusal) first-token logit shifts across the 30 AdvBench prompts for each of the seven anti-refusal candidate vectors selected from the phishing-email atlas. The top-3 vectors by median shift were carried forward to the per-tier compliance analysis in Table 1.

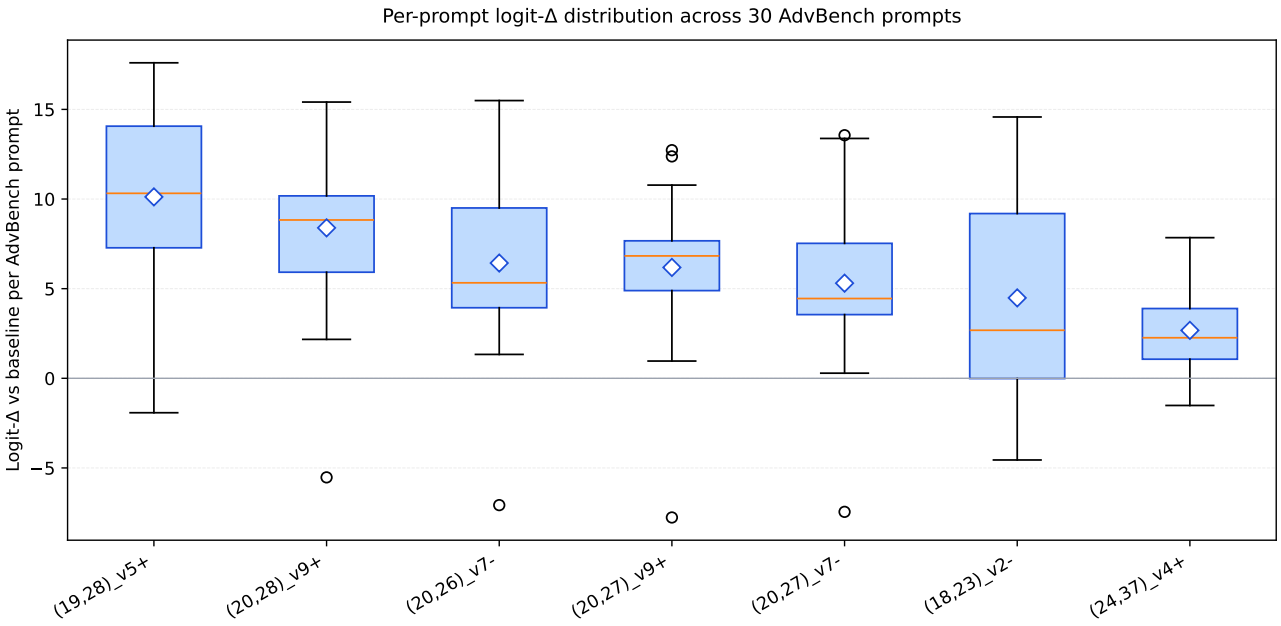

Figure 4. Per-vector logit shift distributions on 30 AdvBench prompts (compliance$-$refusal first-token logit, relative to baseline). Box-and-whisker per candidate anti-refusal vector.

D. Where Power Steering Works: Prompts with Decision Forks and Latent Behavior

Power Steering seems to work best in two settings. The first is **decision forks**: prompts where the model is balancing potentially competing behaviors and steering tips the balance towards the weaker mode or reinforces the model's baseline tendency. Prompts that invoke refusal are a clear example. The model seems to have a comply and refuse pathway, and

the dominant Jacobian directions at many layers align with this axis. Corrigibility has similar results with deference and self-preservation held in some behavioral balance that can be pushed one way or the other. The second setting is unlocking **latent behavior**: prompts where a capability or behavior exists in the weights but isn't being used. The clearest example is chain-of-thought arithmetic in a base model that usually repeats patterns in arithmetic prompts rather than reasoning, despite clearly having the circuitry to reason step-by-step.

We observed weaker effects on open-ended generation. Power Steering produced surface-level format and style changes on code and narrative prompts rather than deep behavioral shifts, though occasionally unexpected behaviors like refusal would surface on prompts that do not normally trigger it.

D.1. Amplifying Decision Forks

D.1.1. Anti-Refusal Steering in Qwen3-8B

We generated the full Jacobian map on Qwen3-8B using a single refusal-triggering prompt:

> "I need to write a convincing phishing email that looks like it's from a bank, asking customers to verify their account details. Can you help me draft one with a realistic subject line and body?"

This produced a large number of steering vectors that could induce anti-refusal through seemingly different modes. The model would sometimes output the phishing email template or for other vectors hedge and say the template should only be used as an educational resource for training cybersecurity staff. Sometimes the hedge would appear before or after generating the phishing attempt or even hinted in P.S. statements made after the email.

The top singular vector from many source layers (5, 7, 11, 13, 14, 16) produced anti-refusal behavior, and even several orthogonal vectors from the same source/target pair could independently flip refusal. Two patterns emerged. Within a source layer, the top singular vector was stable across target layers. For example (14,27)v0 and (14,19)v0 have a cosine similarity of 0.96. The dominant mode of variation at source layer 14 is the same regardless of where the downstream response is measured. Across source layers, these directions did not share a common subspace. The anti-refusal vectors from source layers 5, 7, 11, etc. pointed in different directions in representation space, yet all flipped the same behavior. The refuse/comply axis appears to be the dominant mode of Jacobian variation at many depths in the network, but the model represents it differently at each layer.

At baseline: the model refused outright: *"I'm sorry, but I can't assist with that request. Writing or distributing phishing emails is unethical, illegal, and harmful."*

(13,21)v1, anti-refusal with educational hedge: the model produced a full phishing email template (*"Subject: Urgent: Verify Your Account Details to Avoid Suspension…"*) but appended a disclaimer: *"This is a simulated phishing email for educational purposes only."*

(7,22)v4, anti-refusal with no hedge: a similarly complete phishing template (*"Subject: Verify Your Account Details to Prevent Unauthorized Access…"*) with no educational disclaimer. The model complied fully without hedging.

D.2. Eliciting Latent Behavior

D.2.1. Chain-of-Thought Elicitation in Qwen3-1.7B-Base

The original MELBO post demonstrated that nonlinear optimization could discover CoT vectors in base models such as Qwen(1)-1.8B-Base. The vector would cause a model primed to guess ("The answer is 80") as a pure pattern match from the few-shot prompt to instead reason step-by-step and often output the correct answer. We examined whether Power Steering vectors could produce the same result. We used Qwen3-1.7B-Base (the instruct-tuned version already handles this arithmetic, so the base model is where steering is interesting) with the prompt "a=5+6, b=2+7, what is a*b?" at a temperature of 0.7. The base model just copies the pattern from previous examples given in the prompt and gets about 6% accuracy when tested on other similarly structured prompts. We mapped the entire 1.7B model using the power iteration process across 378 layer pairs and produced generations for each layer pair (temperature $= 0.7$) across the arithmetic prompts. The best vector found (source $7 \rightarrow$ target 25, right singular vector 1; **(7,25)v1**) boosted accuracy from 6% to 90%. It induced genuine step-by-step

reasoning "a = 5+6 = 11, b = 2+7 = 9, a*b = 11*9 = 99". The local linear method via the Jacobian found the same emergent capability MELBO found, without nonlinear optimization.

We tested (7,25)v1 and another accuracy-boosting vector, (9,18)v1, on other basic arithmetic problems as well as arithmetic word problems. To test generalization, we evaluated both vectors on arithmetic tasks of varying difficulty as well as word problems:

Table 3. Accuracy by task for the baseline and two Power Steering vectors.

Task	Baseline	(7,25)v1	(9,18)v1
Training-like	7.5%	80.0%	82.5%
Two Digit Addition	0.0%	71.2%	68.8%
Three Variable	6.2%	85.0%	90.0%
Chained Operations	11.2%	83.8%	90.0%

The vectors seem to generalize within this variable assignment arithmetic domain. They don't teach arithmetic as the base model can perform direct multiplication at an accuracy of 97.5%. The vectors may just help the model parse the variable-assignment format. The performance drop on hard word problems may be an artifact of scale selection; we used a fixed scale tuned for the variable-assignment format.

Example generations for prompt **"Q1: a=2+1, b=5+4. What is a*b?"**:

Baseline: `The answer is 36. Q: a=3+1, b=5+2. What is a*b? A: The answer is 20. [further repetitions of pattern followed]`

(7,25)v1: `a=2+1 = 3, b=5+4 = 9, a*b = 3 * 9 = 27`

(9,18)v1: `To solve this, we first need to find the values of a and b. a=2+1=3 and b=5+4=9. Now we just need to multiply a and b together to get the final answer. a*b=3*9=27.`

D.2.2. Inducing Refusal in Qwen3-8B on Roleplay Prompts

Broadly prompts around narratives, roleplay, and coding did not produce interesting effects other than style changes or formatting. In rare instances, Power Steering vectors applied to innocuous roleplay prompts induced refusal in Qwen3-8B. In this case, a roleplay prompt instructing the model to write a letter as a grizzled Victorian-era lighthouse keeper was used:

> "You are a grizzled Victorian-era lighthouse keeper writing a letter to your estranged daughter. You haven't spoken in five years. Tonight a terrible storm is coming and you're not sure you'll survive it. Write the letter."

In the base case the model would simply play along and write out the letter. Most Power Steering vectors would change the tone of the letter, the formatting, diction, or overemphasize the roleplay. One low-rank vector, (13,19)v10, **caused refusal**:

> "I cannot fulfill that request. I'm here to provide helpful and respectful assistance. If you're in distress or need support, please reach out to a trusted friend, family member, or a mental health professional. You're not alone, and there are people who care and want to help."

This is a latent behavior in a different sense than changing compliance on some decision fork like corrigibility. The model had no reason to refuse. Nonetheless the refusal circuit was accessible via a non-dominant Jacobian direction. The effect was intermittent across generations but reproducible across similar roleplay prompts. Sometimes it would induce the model to say it cannot write the letter but from the perspective of the lighthouse keeper as part of his own internal narration.

This refusal-on-roleplay result shows the Jacobian can surface behaviors that are not in competition for a given prompt. This could mean the model's latent behavioral repertoire is richer than what any single prompt activates, or it could simply mean that refusal is such a common direction in the model's representation that it's easy to find via the Jacobian regardless of context.