Steering Vectors and Linear Encodings for Code Quality in Large Language Models

Will Corcoran
University of California, Santa Barbara
wcorcoran@ucsb.edu

Caleb Stam
University of California, Santa Barbara
cstam@ucsb.edu

Nour Abdelmoneim
University of California, Santa Barbara
nour@ucsb.edu

Abstract

We investigate whether code quality metrics are linearly encoded in the activation space of Large Language Models (LLMs). Using Contrastive Activation Addition (CAA) and linear probes, we analyze three open-weight models across seven code quality metrics. Linear probes achieve $R^2 \geq 0.68$ for four of the seven metrics, showing strong linear encodings across all three models. Steering vector analysis reveals that the Halstead metrics and Cyclomatic Complexity form a cluster, while Comment Ratio remains largely orthogonal. We evaluate generation quality, finding that steering Comment Ratio and Maintainability Index produces monotonic, controllable changes across most models and steering strengths, with minimal degradation in Pass@1 at moderate α. Our results show that LLMs implicitly learn linear representations of code quality, and that steering vectors are a viable training-free mechanism for inference-time control.

1 Introduction

The analysis of Large Language Models has detailed the Linear Representation Hypothesis (LRH) [7] which states "that high-level concepts are represented linearly in the representation space of a model." A key result is intervention via steering vectors injected during inference. Steering vectors have been applied across many domains: reducing sycophancy and hallucination in general-purpose LLMs [10], eliciting truthful responses via probe-weighted intervention [3], and controlling output sentiment and toxicity [12].

Even though LLMs are known for their coding abilities, the domain lacks research on the use of steering vectors for controllable code generation. Style2Code [13] looked into controllable code generation with contrastive representation learning, while [9] showed steering vectors exist for languages and libraries. However, neither work examines whether code quality metrics are linearly encoded in LLM activations. Motivated by overly-verbose and poorly written code, this is a natural investigation. We summarize our novel contributions as follows:

- **Linear Representations.** We show that, across layers, LLMs linearly encode code quality metrics. This is shown through linear probes, for four of seven metrics, with $R^2 \geq 0.68$ (Section 3.1).
- **Steering Vectors.** LLMs produce steering vectors for code quality metrics that fall into two groups: Halstead and related metrics and Comment Ratio (Section 3.2).

Final Project for CS291A (UCSB).

- **Controllable Generation.** Across an α sweep, steering vectors for Comment Ratio and Maintainability Index show a monotonic increase in affected metric in isolation of generation correctness (Section 4).

2 Methodology

2.1 Setup: Models, Datasets, and Metrics

Models. We analyze 3 models: two general-purpose, instruction models (Llama 3.1 8B Instruct [1] and Gemma 2 9B-IT [11]) and one coding model (Qwen 2.5 Coder 14B Instruct [8]). Each of these models is instruction-tuned, so we apply standard chat templates. A detailed breakdown of layers and dimensions can be found in Appendix A.1.

Datasets. For steering vector extraction, we utilize The Stack v2 [4]. We begin by gathering 40k Python files and filtering as follows: an accepted file must be less than 7000 characters and valid syntactically. From there, we de-duplicated and turned all top-level scripts into a function to match our downstream evaluation setting (function completion) and to properly compute metrics. This leaves us a dataset of 20k files.

Metrics. We compute the following metrics with the `radon` package: cyclomatic complexity, maintainability index, comment ratio, Halstead volume, Halstead difficulty, Halstead effort, and source lines of code. These are extremely well-studied metrics in software engineering literature, some having been around since the 1970s. Complete descriptions and formulas can be found in Table 1. Importantly, between 30 and 50% of files fell at the metric minimum, so to not dilute our analysis, we omit these boundary examples when computing linear probes (Section 3.1) and steering vectors (Section 3.2).

2.2 Activation Extraction

For both linear probes and steering vectors, we extract hidden states from each layer of the model. Given a code sample d_i, a single forward pass yields a sequence of hidden states $\mathbf{h}_{i,t}^{\ell} \in \mathbb{R}^m$ for each token position $t \in \{1, \ldots, T_i\}$ and layer $\ell \in \{0, \ldots, L\}$, where m is the model dimension, T_i is the sequence length of d_i, $\ell = 0$ denotes the embedding layer, and $\ell = L$ denotes the final layer.

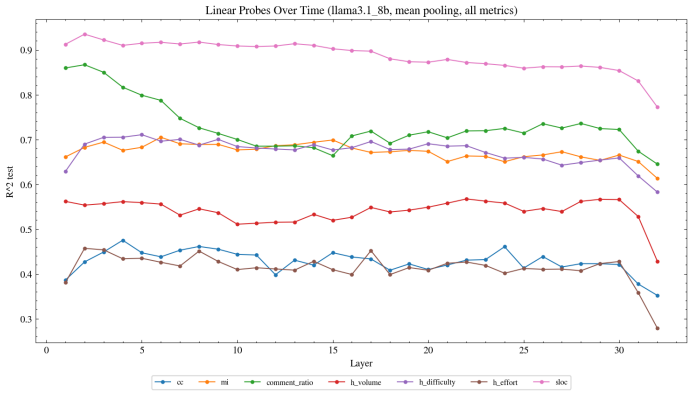
We then reduce to obtain a single vector per sample per layer. This leaves two pooling strategies: last-token pooling, $\mathbf{h}_i^{\ell} = \mathbf{h}_{i,T_i}^{\ell}$, and mean pooling, $\mathbf{h}_i^{\ell} = \frac{1}{T_i} \sum_{t=1}^{T_i} \mathbf{h}_{i,t}^{\ell}$. We discuss the choice of pooling strategy in Section 3.1. However, unless otherwise noted, we use mean pooling.

3 Analysis

In the following sections, we show proof that LLMs linearly encode code quality metrics with linear probes and steering vectors. Both, in different ways, exhibits a linear representation.

Table 1: Code quality metrics used in this work. For Cyclomatic Complexity: E = edges, N = nodes, and P = connected components in the control flow graph. For Halstead metrics: N_1, N_2 are the total operator and operand counts. η_1, η_2 are the unique operator and operand counts. $N = N_1 + N_2$ and $\eta = \eta_1 + \eta_2$.

Metric	Description	Formula
Cyclomatic Complexity (CC) [5]	Number of independent code paths	$CC = E - N + 2P$
Maintainability Index (MI) [6]	0 to 100 score; higher = more maintainable	$MI = 171 - 5.2\ln(HV) - 0.23 \cdot CC - 16.2\ln(SLOC)$
Comment Ratio (CR)	Fraction of lines that are comments	$CR = L_{\text{comment}}/L_{\text{total}}$
Source Lines of Code (SLOC)	Non-blank, non-comment lines	$SLOC = L_{\text{total}} - L_{\text{blank}} - L_{\text{comment}}$
Halstead Volume (HV) [2]	Effort to encode the program	$HV = N \log_2 \eta$
Halstead Difficulty (HD) [2]	Effort to understand the code	$HD = (\eta_1/2) \cdot (N_2/\eta_2)$
Halstead Effort (HE) [2]	Volume $\times$ Difficulty	$HE = HV \times HD$



(a)

Metric	Llama	Gemma	Qwen
SLOC	0.935	0.934	0.937
CR	0.867	0.872	0.872
HD	0.711	0.680	0.688
MI	0.705	0.685	0.721
HV	0.567	0.509	0.577
CC	0.475	0.459	0.421
HE	0.457	0.475	0.434

(b)

Figure 1: (a) Linear probe R^2 test scores per layer for Llama 3.1 8B Instruct (mean pooling). Each line corresponds to a code quality metric; scores are generally stable across layers with a sharp drop at the final layer. Additional plots for Gemma 2 and Qwen 2.5 can be found in Figure 4. (b) Best R^2 test score per metric across all three models with mean pooling, selecting the best layer per metric.

3.1 Linear Probes

With the activations collected in Section 2.2, we train linear probes to test whether code quality metrics are linearly encoded in the latent space of the model. For each metric ϕ_j and layer ℓ, we train a linear probe $\mathcal{F}_{j,\ell} : \mathbb{R}^m \to \mathbb{R}$ of the form:

$$\mathcal{F}_{j,\ell}(\mathbf{h}_i^\ell) = \mathbf{w}_{j,\ell}^\top \mathbf{h}_i^\ell + b_{j,\ell} \tag{1}$$

where $\mathbf{w}_{j,\ell} \in \mathbb{R}^m$ and $b_{j,\ell} \in \mathbb{R}$ are learned by minimizing the ridge objective:

$$\min_{\mathbf{w},b} \sum_{i=1}^{|\mathcal{D}|} \left(\mathbf{w}^\top \mathbf{h}_i^\ell + b - \phi_j(d_i)\right)^2 + \lambda \|\mathbf{w}\|^2 \tag{2}$$

We let $\lambda = 1.0$ and report R^2 on a final test set (10%). We evaluate both mean and last-token pooling strategies. However, in all cases, except for Gemma SLOC and Gemma HV, mean pooling outperforms last token. Thus, in all future analysis, we select mean pooling.

As shown in Figure 1b, all models have relatively the same encoding statistics. This shows that each model learns, roughly, similar encodings for each metric. The models, with ease, linearly encode metrics related to length of input (source lines and comment ratio). This is anticipated given that SLOC and CR are related to sequence length. Importantly, HD and MI are also highly linearly encoded with $R^2 \geq 0.68$. HV, CC, and HE also have some considerable linearly encoding with $R^2 \geq 0.42$.

Figures 1a and 4 each show a similar trend: linear representations are learned early and drift over time, often taking large downward turns in the latest layers. This suggests that deeper layers learn more complex representations of these metrics. Across the three models, metrics appear in a similar order. Halstead Volume is considerably tighter to cyclomatic complexity and Halstead Effort in Gemma 2 than Qwen 2.5 and Llama 3.1.

3.2 Steering Vectors

Contrastive Activation Addition. We use Contrastive Activation Addition (CAA) [10] to derive steering vectors for each code quality metric ϕ_j. Given our dataset $\mathcal{D}$ and the jth metric, we rank the files by their ground-truth metric value $\phi_j(f_i)$ and drop the min- and max-valued examples as discussed in Section 2.1. We define the top and bottom $p\%$ subsets as $\mathcal{F}_j^+ = \{f_i \in \mathcal{D} : \phi_j(f_i) \geq Q_{1-p/100}\}$ and $\mathcal{F}_j^- = \{f_i \in \mathcal{D} : \phi_j(f_i) \leq Q_{p/100}\}$, where $Q_{1-p/100}$ and $Q_{p/100}$ are the corresponding percentiles of ϕ_j over $\mathcal{D}$. Using difference-of-means, the steering vector at layer ℓ is

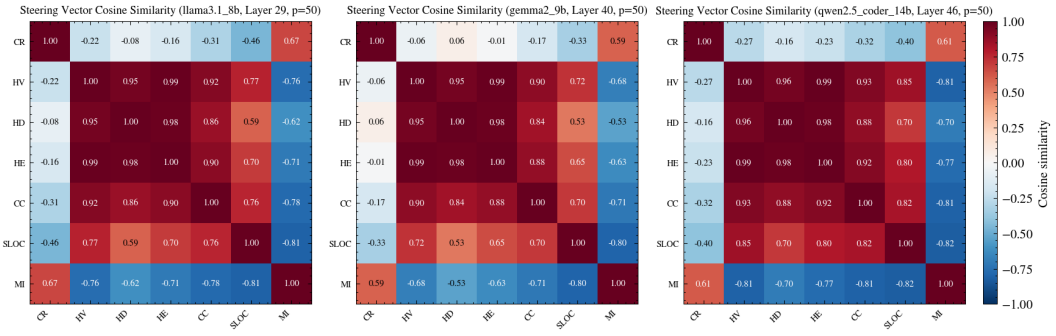


Figure 2: Steering vector cosine similarity matrices for (left) Llama 3.1 8B Instruct (layer 29), (middle) Gemma 2 9B-IT (layer 40), and (right) Qwen 2.5 Coder 14B Instruct (layer 46), all with $p = 50$. The Halstead metrics (HV, HD, HE) and CC form a tightly aligned cluster, while CR and MI are largely orthogonal or negatively correlated with this cluster.

then computed as

$$\mathbf{v}_j^{(\ell)} = \frac{1}{|\mathcal{F}_j^+|} \sum_{f_i \in \mathcal{F}_j^+} \mathbf{h}_i^{(\ell)} - \frac{1}{|\mathcal{F}_j^-|} \sum_{f_i \in \mathcal{F}_j^-} \mathbf{h}_i^{(\ell)}, \tag{3}$$

where $\mathbf{h}_i^{(\ell)} \in \mathbb{R}^d$ is the hidden state of file f_i at layer ℓ. At inference time, the hidden state is modified as

$$\mathbf{h}^{(\ell)'} = \mathbf{h}^{(\ell)} + \alpha\, \mathbf{v}_j^{(\ell)}, \tag{4}$$

where α is a tunable steering coefficient. We sweep $p \in \{10, 25, 50\}$ to assess sensitivity to class boundary selection. Visualizations used $p = 50$, generation uses $p = 10$.

Relationships. Across models, the metrics each take a distinct pattern. The Halstead metrics (HV, HD, HE) are tightly clustered, as expected. Cyclomatic complexity joins this grouping as it is approximately parallel (≥ 0.8). SLOC is less parallel, but still largely related to the Halstead metrics. Maintainability index is roughly anti-parallel with this cluster, as well, which is expected, given that the formula is negatively related to HV, CC, and SLOC. Comment Ratio is largely orthogonal to each metric, except for Maintainability Index. We select comment ratio due to orthogonality and maintainability index due to its representation of the aforementioned cluster as a steering candidates in Section 4.1.

Counterintuitively, Qwen 2.5 Coder, the model fine-tuned for coding tasks, is the least disentangled: each of the cosine similarities is roughly 0.05 to 0.15 higher than Gemma and Llama. Metric correlations in the dataset can be found in Figure 5. Steering cosine similarities mainly follow the correlation, though, the models entangle CC and SLOC more than the underlying metrics. The reason for this is unknown. The layer-wise dynamics of these metrics are discussed in Appendix B.3.

4 Evaluation

We evaluate generation quality on `BigCodeBench` [14], with $\alpha = 0$ serving as the unsteered baseline across all experiments. Steering is applied exclusively on generated tokens, there is no effect on pre-fill.

4.1 Pass@1 Steering

Across nine steering strengths per model, Table 2 reports Pass@1 and mean percentage change in Comment Ratio (CR) and Maintainability Index (MI) relative to the $\alpha = 0$ baseline. As is standard for Pass@1, we use greedy decoding and generate a single example. Because the models occupy different latent spaces, we use model-specific α grids. Llama 3.1 8B-Instruct spans $[-1, 1]$, Gemma 2 9B-IT spans $[-2, 2]$, and Qwen 2.5 Coder 14B spans $[-10, 10]$, each sampled at four symmetric steps (see Table 5). Llama and Gemma are each steered at Layer 30, Qwen at Layer 38, and steering

Table 2: Pass@1 results on BigCodeBench under CAA steering. Each row corresponds to a steering strength α. ΔCR% and ΔMI% report the mean percentage change in Comment Ratio and Maintainability Index relative to the baseline across generated completions. **Bold** and underline denote the best and second-best Pass@1 within each model column, respectively. See Table 5 for raw α values per model.

	Llama 3.1 8B-Instruct				**Gemma 2 9B-IT**				**Qwen 2.5 Coder 14B**			
	Comment Ratio		Maintainability Index		Comment Ratio		Maintainability Index		Comment Ratio		Maintainability Index	
α	ΔCR%	P@1↑	ΔMI%	P@1↑	ΔCR%	P@1↑	ΔMI%	P@1↑	ΔCR%	P@1↑	ΔMI%	P@1↑
$-\alpha_4$	-16.9 ± 30.8	0.269	-0.6 ± 5.7	0.257	-18.3 ± 42.7	0.322	-1.5 ± 8.1	**0.330**	-18.0 ± 28.6	0.413	-1.4 ± 6.0	0.406
$-\alpha_3$	-11.0 ± 23.8	0.291	-0.1 ± 5.2	0.292	-14.3 ± 39.0	0.323	-1.3 ± 7.4	0.328	-11.4 ± 23.2	0.451	-0.8 ± 4.5	0.461
$-\alpha_2$	-4.6 ± 19.0	0.304	-0.1 ± 4.2	0.283	-9.2 ± 33.9	0.324	-1.0 ± 6.5	0.330	-6.5 ± 19.0	0.461	-0.5 ± 3.4	0.459
$-\alpha_1$	-1.0 ± 16.6	0.304	0.0 ± 3.1	0.304	-4.7 ± 23.6	0.321	-0.6 ± 5.2	0.323	-2.8 ± 14.2	0.458	-0.1 ± 2.3	0.458
0	0.0 ± 0.0	**0.307**	0.0 ± 0.0	0.307	0.0 ± 0.0	0.312	0.0 ± 0.0	0.312	0.0 ± 0.0	**0.463**	0.0 ± 0.0	**0.463**
α_1	$+2.9 \pm 25.7$	0.304	$+0.1 \pm 3.3$	**0.316**	$+1.8 \pm 26.4$	0.320	$+0.7 \pm 5.7$	0.321	$+5.1 \pm 28.4$	0.461	$+0.2 \pm 3.1$	0.454
α_2	$+4.4 \pm 27.1$	0.302	-0.2 ± 3.9	0.309	$+7.8 \pm 49.2$	**0.325**	$+1.6 \pm 8.3$	0.324	$+8.8 \pm 37.5$	0.454	$+0.2 \pm 4.0$	0.459
α_3	$+6.3 \pm 28.8$	0.280	-0.4 ± 4.6	0.311	$+11.2 \pm 62.2$	0.321	$+1.7 \pm 9.1$	0.326	$+12.0 \pm 40.6$	0.425	$+0.4 \pm 5.1$	0.202
α_4	$+12.1 \pm 41.8$	0.238	-1.1 ± 6.4	0.260	$+13.8 \pm 65.2$	0.322	$+2.4 \pm 10.2$	0.325	$+15.3 \pm 43.5$	0.270	$+2.7 \pm 11.0$	0.009

vectors are built with $p = 10$. We steer CR and MI independently and discuss Pass@1 results below. Pass@5 results under identical conditioning are deferred to Appendix C.1.

Across each model, Comment Ratio expresses strong steering signal. Mean change in comment ratio is monotonically increasing with α, showing the linear representation of Comment Ratio within models. $\pm\alpha_1, \pm\alpha_2, \pm\alpha_3$ have no impact on Pass@1. In the case of Gemma, we find that steering $\pm\alpha_2$ causes the model to improve generation quality.

Similar steerability is observed for Maintainability Index. With the exception of Llama 3.1, MI increases monotonically with α. The reason for Llama's poor MI steering is unknown, but likely, MI has a complex representation, not well traversed by steering vectors. For Llama and Gemma, steering moves towards a subspace with favorable Pass@1 generations. Susceptibility to oversteering is witnessed in Qwen with α_3, α_4 resulting in poor generation quality. The takeaway is clear, LLMs linearly represent code quality metrics and this can be used for controllable generation without disrupting quality.

4.2 Analysis of Model Outputs

While there is a clear linear relationship between the steering strength α and the percentage change in the two steering attributes demonstrates, we notice that for any single value of α, the standard

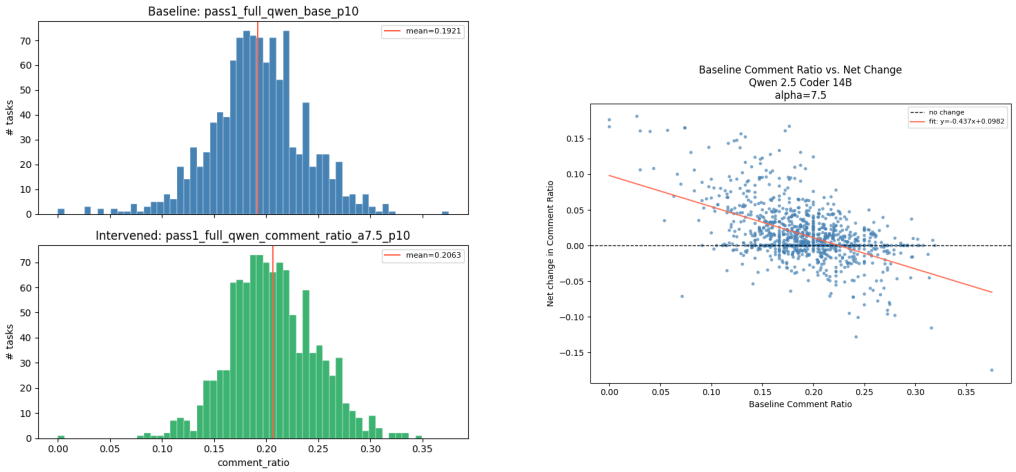


Figure 3: Visualizations of comment ratios of generated code from Qwen 2.5 Coder 14B in the baseline case and with steering strength α_3. **Left:** Distribution of comment ratios in generated code From with no intervention (top) and steering (bottom). **Right:** Scatter plot of prompts by the comment ratio of generated code with no intervention and the net change in comment ratio with steered generation. Regression line indicates a clear negative relationship.

		Llama 3.1 8B-Instruct		Gemma 2 9B-IT		Qwen 2.5 Coder 14B	
		Not Steered	Steered	Not Steered	Steered	Not Steered	Steered
Comment Ratio	Not Prompted	0.0 ± 0.0	$+4.4 \pm 27.1$	0.0 ± 0.0	$+22.5 \pm 120.1$	0.0 ± 0.0	$+8.8 \pm 37.5$
	Prompted	$+35.2 \pm 40.4$	$+37.9 \pm 43.8$	$+152.5 \pm 172.3$	$+157.3 \pm 175.3$	$+26.8 \pm 50.4$	$+32.9 \pm 53.0$
Maintainability Index	Not Prompted	0.0 ± 0.0	$+0.1 \pm 3.3$	0.0 ± 0.0	$+1.3 \pm 5.2$	0.0 ± 0.0	$+0.2 \pm 4.0$
	Prompted	$+0.3 \pm 3.2$	-0.1 ± 3.9	-0.4 ± 10.3	$+0.72 \pm 11.1$	$+0.1 \pm 3.5$	$+0.5 \pm 4.1$

Table 3: Percent change in each target metric relative to the per-model baseline, across prompting and steering conditions. Steering strength α_2 used across models.

deviation of the change in steering attributes is unusually high. To better understand this variance, we conduct an exploratory data analysis of the distribution of steerable attributes in the generated output code.

Our results, for comment ratio on Qwen 2.5 Coder 14B, can be found in Figure 3. Here, we observe that as expected, the mean comment ratio of the generated code increases with steering. However, we also notice that the distribution of comment ratios in generated code tends to place more weight on the tails without intervention, and to be more clustered around the mean with intervention.

The mechanism by which this occurs can be seen in the scatter plot on the right. Here, we observe a negative correlation between the comment ratio in code generated in the baseline and comment ratio in steered code. That is, prompts that generate code with low comment ratios in the baseline case tend to be significantly improved by steering, while prompts that generate code with high comment ratios in the baseline case can actually be degraded. Similar results are observed across models, steering strength, and attributes.

These results suggest that steering's linear behavior occurs in the aggregate, rather than on the scope of individual examples. It may also mean that steering tends to map activations towards a region of latent space that leads to lower variance in model outputs. In this sense, steering may act as an implicit temperature control on the steered attribute, leading to more consistent model outputs.

4.3 Steering with Prompting

To compare the efficacy of steering to alternative methods, we compared the change in relevant attributes when the model is steered towards them to the change in that attribute when it is simply prompted to elicit the desired result. For comment ratio and maintainability index, we appended the following suffixes to each prompt:

- Comment Ratio: "Include inline comments explaining each step."
- Maintainability Index: "Prioritize generating simple, maintainable code."

We evaluated model outputs both with these prompts alone and when the prompts are used in combination with steering. Results can be found in Table 3. Evaluation was done on the full BigCodeBench dataset.

The results indicate that on comment ratio, steering alone is less effective than prompting alone. However, on both models tested, a combination of steering and prompting for the desired result led to the largest increase in performance. On maintainability index, prompting failed to cause the expected result. And, while the high variance of steering results for fixed-α may not appear significant, the linear trend with respect to α show in in Section 4.1 verify that improvement from steering is valid signal.

5 Conclusion

Discussion. We ask *do LLMs linearly encode code quality metrics?* to which we assert the answer is *yes*, backed by linear probe and steering vector analysis. Even though LLMs are not trained on code quality metrics, they internally approximate them. As shown in Section 4.1, this allows for controllable generation of Comment Ratio and Maintainability Index without degradation in benchmark performance. Therefore, steering vectors a plausible alternative to prompting for inference-time intervention and demonstrate utility when used in combination with prompting.

Limitations. Our work is limited to Python, it remains to be seen if similar results are present in other major languages. We extract activations on a general code dataset (The Stack v2) and steer on a function completion benchmark (BigCodeBench). This is a possible distribution misalignment, however, our steering results would likely only improve.

Additionally, due to computational restraints our results were limited to relatively lightweight reasoning models. Validation of similar results on large state-of-the-art reasoning models would provide further evidence for the practical utility of steering for code quality.

Future Work. Similar analysis for general software engineering anti-patterns and well-designed systems remains open work. Analysis into the relationship between languages has been done [9], but could be merged with our work for deeper insight.

References

[1] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu,

Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024.

[2] Maurice H. Halstead. Elements of software science. 1977.

[3] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 41451–41530. Curran Associates, Inc., 2023.

[4] Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, Tianyang Liu, Max Tian, Denis Kocetkov, Arthur Zucker, Younes Belkada, Zijian Wang, Qian Liu, Dmitry Abulkhanov, Indraneil Paul, Zhuang Li, Wen-Ding Li, Megan Risdal, Jia Li, Jian Zhu, Terry Yue Zhuo, Evgenii Zheltonozhskii, Nii Osae Osae Dade, Wenhao Yu, Lucas Krauß, Naman Jain, Yixuan Su, Xuanli He, Manan Dey, Edoardo Abati, Yekun Chai, Niklas Muennighoff, Xiangru Tang, Muhtasham Oblokulov, Christopher Akiki, Marc Marone, Chenghao Mou, Mayank Mishra, Alex Gu, Binyuan Hui, Tri Dao, Armel Zebaze, Olivier Dehaene, Nicolas Patry, Canwen Xu, Julian McAuley, Han Hu, Torsten Scholak, Sebastien Paquet, Jennifer Robinson, Carolyn Jane Anderson, Nicolas Chapados, Mostofa Patwary, Nima Tajbakhsh, Yacine Jernite, Carlos Muñoz Ferrandis, Lingming Zhang, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder 2 and the stack v2: The next generation, 2024.

[5] T.J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, 1976.

[6] Paul Oman and Jack Hagemeister. Metrics for assessing a software system's maintainability. In *Proceedings Conference on Software Maintenance 1992*, pages 337–344. IEEE, 1992.

[7] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Causal Representation Learning Workshop at NeurIPS 2023*, 2023.

[8] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[9] Md Mahbubur Rahman, Arjun Guha, and Harshitha Menon. Steering code llms with activation directions for language and library control, 2026.

[10] Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[11] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit

Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024.

[12] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

[13] Dutao Zhang, Nicolas Rafael Arroyo Arias, YuLong He, and Sergey Kovalchuk. Style2code: A style-controllable code generation framework with dual-modal contrastive representation learning, 2026.

[14] Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*, 2024.

Table of Contents

A Models and Evaluation Setting

A.1 Model Information

Table 4: Model architectures. L denotes the number of transformer layers and d the hidden dimension.

Model	L	d
Llama 3.1 8B Instruct	32	4096
Gemma 2 9B Instruct	42	3584
Qwen 2.5 Coder 14B Instruct	48	5120

A.2 Alphas Per Model

Table 5: Raw α values used per model.

	$-\alpha_4$	$-\alpha_3$	$-\alpha_2$	$-\alpha_1$	0	α_1	α_2	α_3	α_4
Llama 3.1 8B-Instruct	-1	-0.75	-0.5	-0.25	0	0.25	0.5	0.75	1
Gemma 2 9B-IT	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
Qwen 2.5 Coder 14B	-10	-7.5	-5	-2.5	0	2.5	5	7.5	10

B Additional Analysis

B.1 Gemma and Qwen Linear Probes

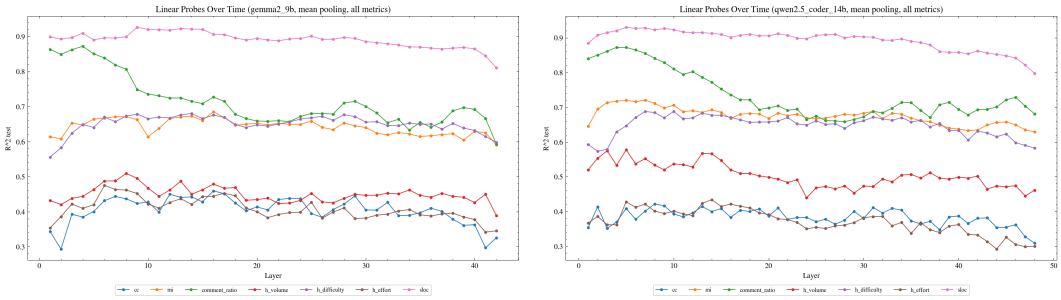


Figure 4: Linear probe R^2 test scores per layer for (a) Gemma 2 9B-IT and (b) Qwen 2.5 Coder 14B Instruct (mean pooling). Each line corresponds to a code quality metric. Results are consistent with Llama 3.1 8B Instruct (Figure 1): SLOC and Comment Ratio are most linearly encoded, while CC and Halstead Effort are weakest. Notably, Halstead Volume is considerably tighter in Gemma 2 than Qwen 2.5 and Llama.

B.2 Metric Correlations in Data

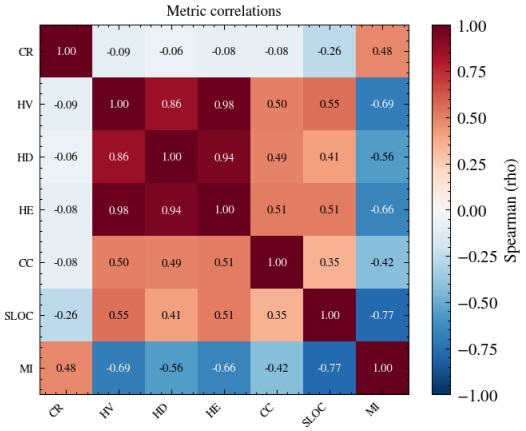


Figure 5: Correlations between code quality metrics computed on the Stack v2 dataset. The Halstead metrics (HV, HD, HE) are strongly intercorrelated ($\rho \geq 0.86$) and moderately correlated with CC and SLOC, while CR is largely uncorrelated with all other metrics. MI is negatively correlated with the Halstead cluster and SLOC.

B.3 Principal Angle Emergence

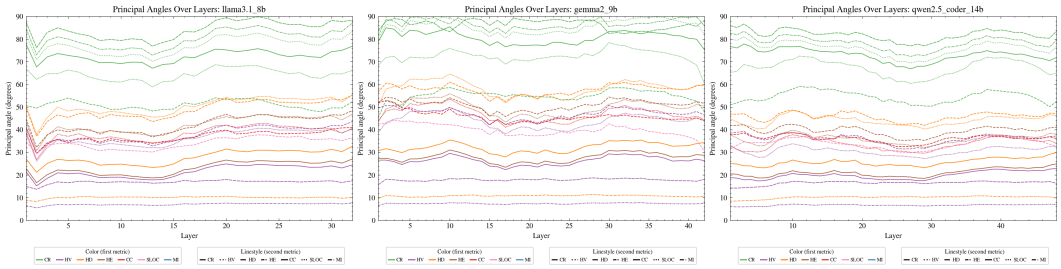


Figure 6: Principal angles between pairs of steering vector subspaces across layers for (a) Llama 3.1 8B Instruct, (b) Gemma 2 9B-IT, and (c) Qwen 2.5 Coder 14B Instruct. Line color denotes the first metric and linestyle denotes the second. Angles near 0° indicate highly aligned subspaces and angles near 90° indicate orthogonality. The angles between steering vectors are largely unchanged across layers and consistent between models, suggesting similar relative relationships.

C Additional Results

C.1 Pass@5 Steering

Table 6: Pass@5 results on BigCodeBench under CAA steering. Each row corresponds to a steering strength α. $\Delta\mathrm{CR}\%$ and $\Delta\mathrm{MI}\%$ report the mean percentage change (with standard deviation) in Comment Ratio and Maintainability Index relative to the baseline across generated completions. Steering vectors and their extraction are consistent with the explanation in Section 4.1. See Table 5 for raw α values per model. **Bold** and underline denote the best and second-best pass@5 within each model column, respectively.

	Llama 3.1 8B-Instruct				**Gemma 2 9B-IT**				**Qwen 2.5 Coder 14B**			
	Comment Ratio		Maintainability Index		Comment Ratio		Maintainability Index		Comment Ratio		Maintainability Index	
α	$\Delta\mathrm{CR}\%$	P@5↑	$\Delta\mathrm{MI}\%$	P@5↑	$\Delta\mathrm{CR}\%$	P@5↑	$\Delta\mathrm{MI}\%$	P@5↑	$\Delta\mathrm{CR}\%$	P@5↑	$\Delta\mathrm{MI}\%$	P@5↑
$-\alpha_4$	-17.8 ± 28.6	0.361	-0.5 ± 4.5	0.368	-21.8 ± 44.5	0.373	-1.7 ± 5.8	0.376	-18.5 ± 26.1	0.454	-1.4 ± 5.0	0.446
$-\alpha_3$	-11.1 ± 20.3	0.386	-0.2 ± 3.8	0.383	-15.4 ± 47.4	0.375	-1.2 ± 5.3	0.371	-11.8 ± 20.5	0.482	-0.8 ± 4.2	**0.482**
$-\alpha_2$	-5.6 ± 14.5	0.403	-0.1 ± 2.9	0.386	-11.6 ± 42.7	**0.388**	-0.9 ± 4.2	**0.385**	-6.3 ± 15.8	0.479	-0.5 ± 2.9	**0.482**
$-\alpha_1$	-1.8 ± 8.9	0.403	0.0 ± 2.1	0.401	-5.0 ± 43.6	0.381	-0.5 ± 3.8	0.381	-3.0 ± 10.8	0.483	-0.2 ± 1.8	**0.482**
0	0.0 ± 0.0	**0.412**	0.0 ± 0.0	0.412	0.0 ± 0.0	0.381	0.0 ± 0.0	0.381	0.0 ± 0.0	0.483	0.0 ± 0.0	**0.482**
α_1	$+1.6 \pm 9.0$	0.393	-0.0 ± 2.0	**0.419**	$+10.8 \pm 80.7$	0.382	$+0.8 \pm 4.2$	0.383	$+4.2 \pm 26.0$	0.485	$+0.1 \pm 2.0$	**0.482**
α_2	$+3.2 \pm 17.7$	0.408	-0.1 ± 2.7	0.409	$+22.5 \pm 120.1$	0.380	$+1.3 \pm 5.2$	0.375	$+7.6 \pm 33.5$	**0.488**	$+0.2 \pm 3.3$	0.480
α_3	$+5.4 \pm 33.4$	0.398	-0.4 ± 3.8	0.409	$+27.8 \pm 123.2$	0.378	$+1.9 \pm 6.3$	0.378	$+11.5 \pm 37.6$	0.465	$+0.2 \pm 4.4$	0.268
α_4	$+10.6 \pm 59.9$	0.367	-1.0 ± 4.3	0.376	$+38.1 \pm 144.0$	0.377	$+2.3 \pm 7.0$	0.371	$+14.3 \pm 39.2$	0.327	$+2.8 \pm 10.2$	0.017