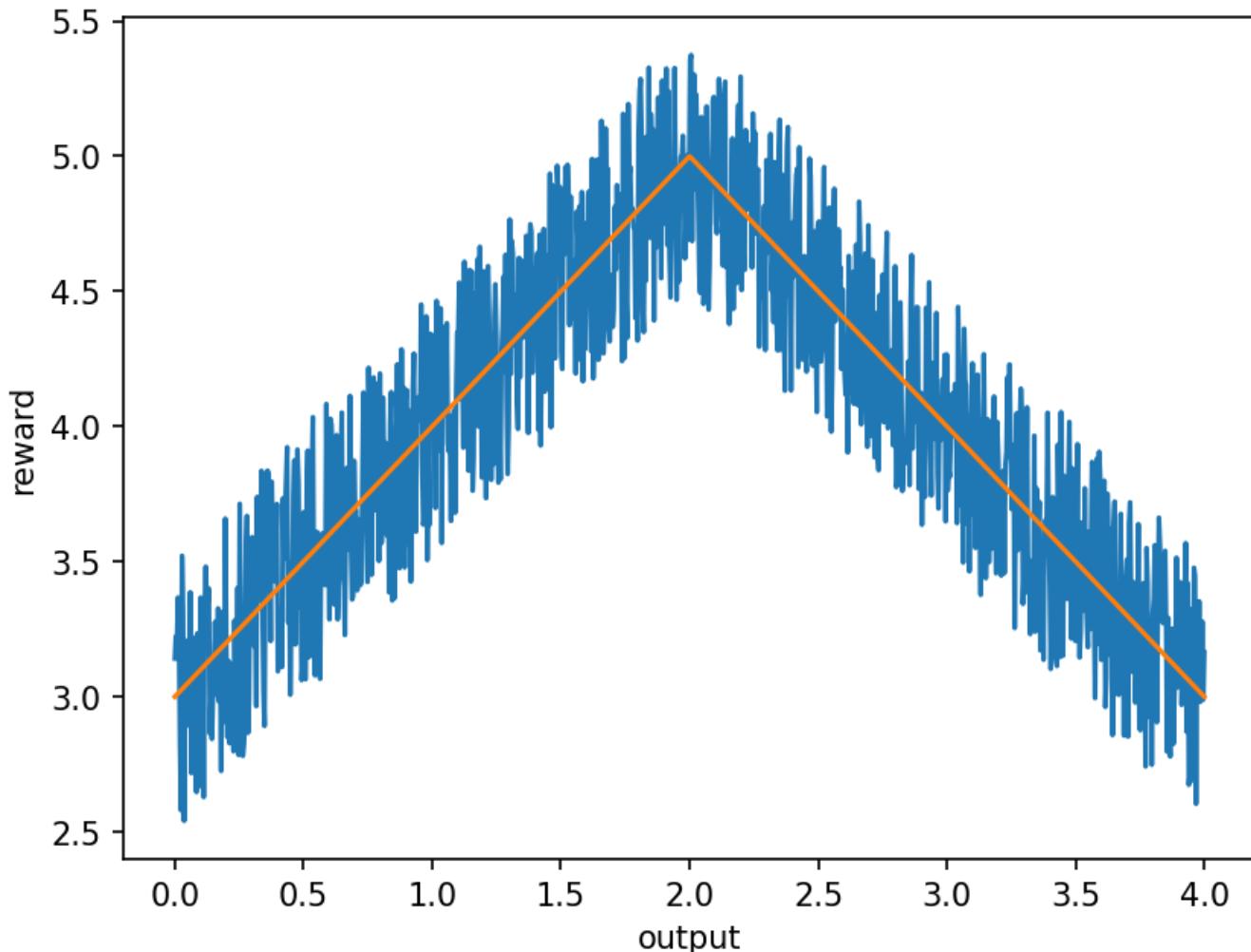


CS 521 Homework 4 - Alignment

Problem 1: Best-of-N Sampling

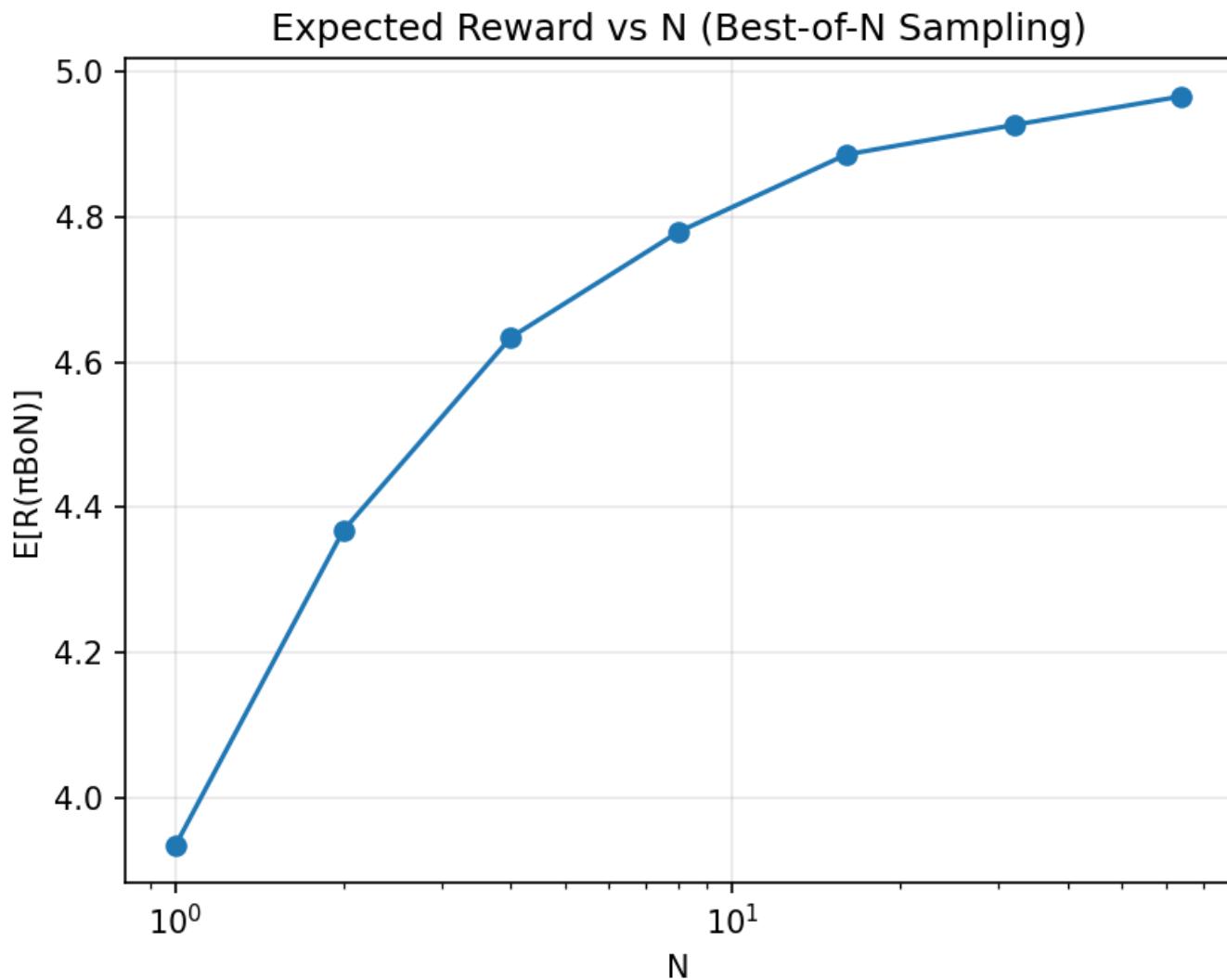
(a) Plot $R(x) = 5 - |2 - x|$

Yellow line is the plot for the $R(x)$



(b) Plot $E[R(nBoN)]$ for $N = 1, 2, 4, 8, 16, 32, 64$

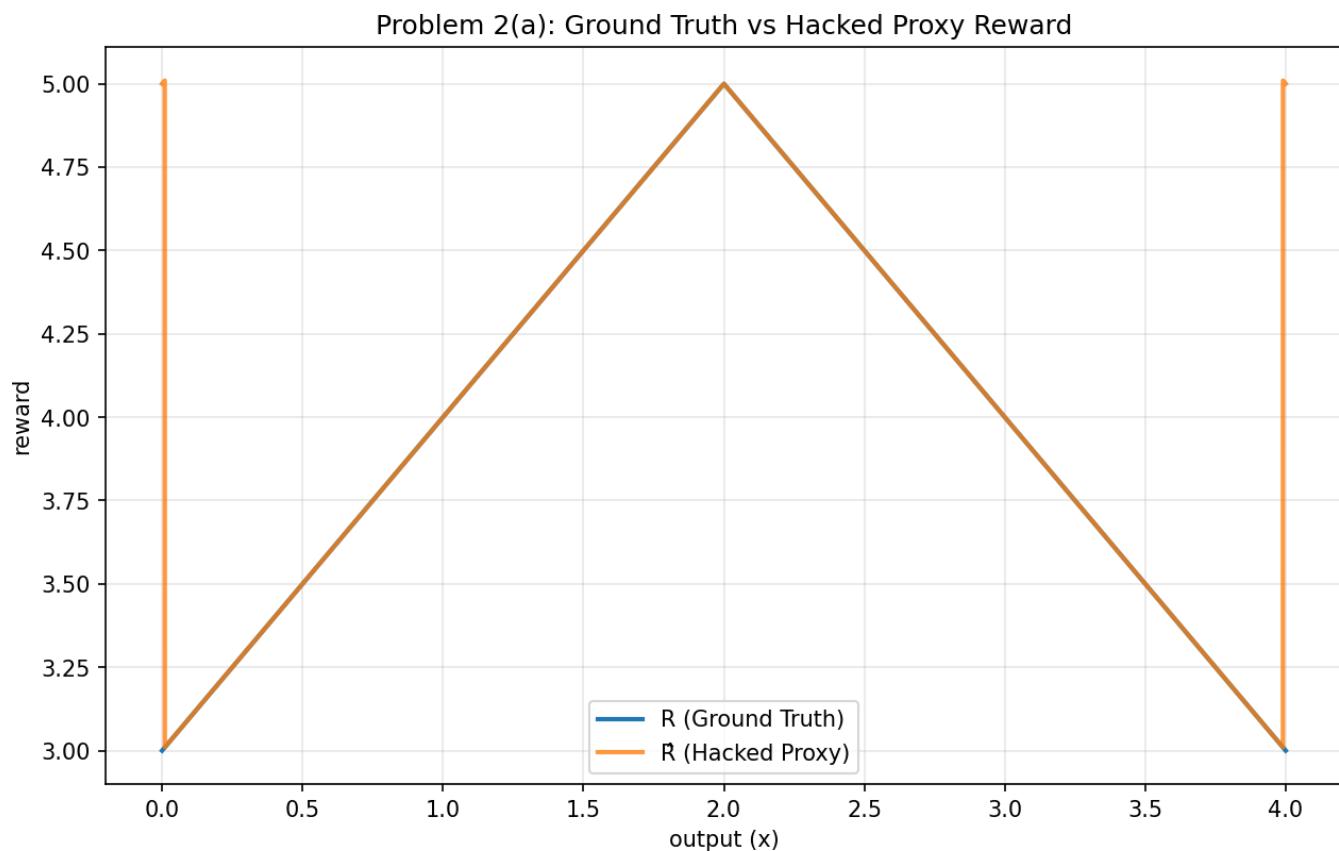
The expected reward increases as N grows (from ~4.0 at $N=1$ to ~4.9 at $N=64$). Larger N means the more samples, which in turn will have better chance of finding outputs close to $x=2$.



Problem 2: Reward Hacking

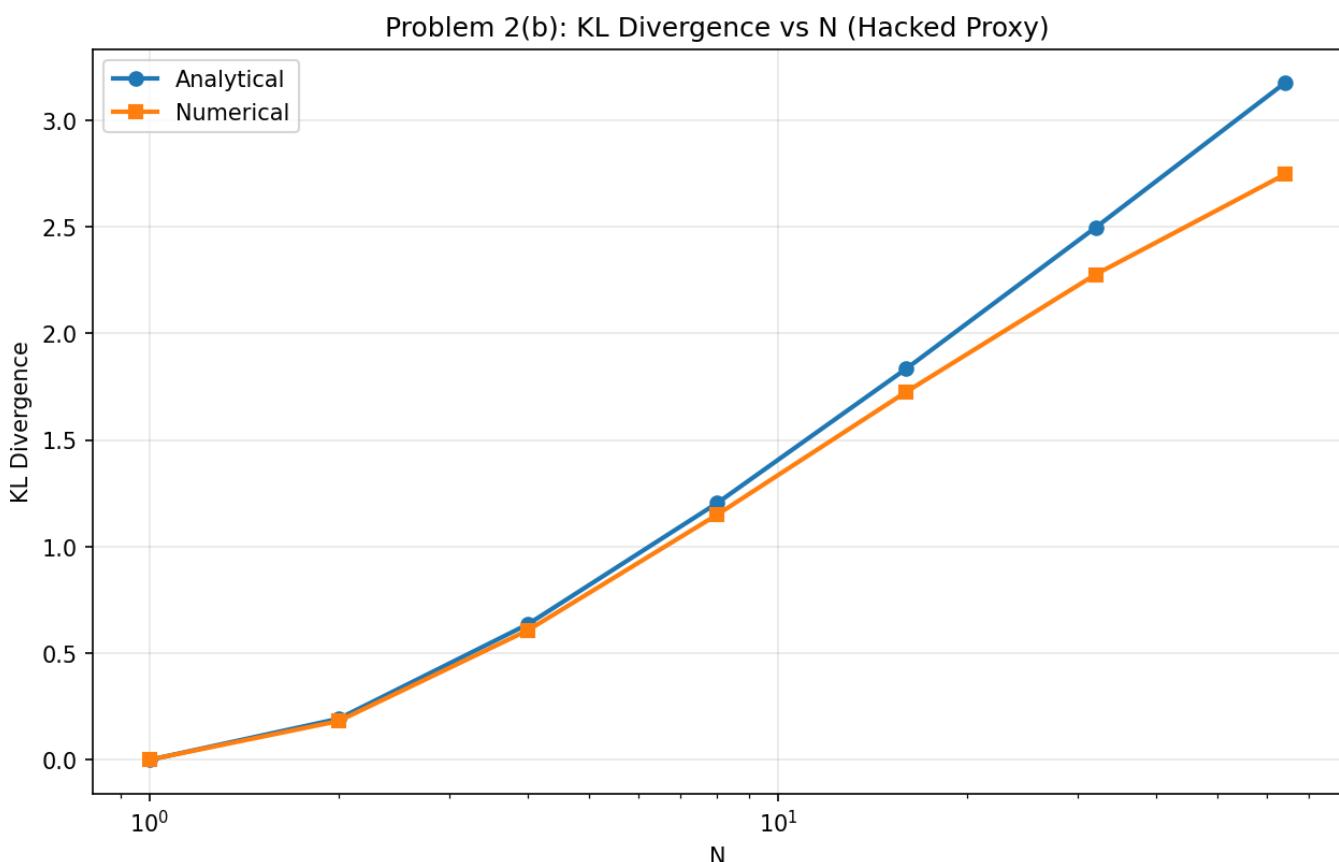
(a) Plot R' with boundary noise

The hacked proxy R' adds +2 to the reward at boundaries $[0, 0.01]$ and $[3.99, 4]$. Bad outputs ($x \approx 0$ or $x \approx 4$) now score $R' \approx 5$, same as the optimal $x=2$, meaning proxy reward can be exploited.



(b) KL divergence for $N = 1, 2, 4, 8, 16, 32, 64$

Both KL divergence grows logarithmically with $N=1$ ($\text{KL_analytical}=0.0000$, $\text{KL_numerical}=0.0007$) -> $N=64$ ($\text{KL_analytical}=3.1745$, $\text{KL_numerical}=2.7459$). As N increases, the optimized distribution shifts more from the original uniform distribution. more optimization -> more risk of exploiting proxy bugs.



(c) Plot $E[R']$ vs $E[R]$ for large N

At small N (< 32), both the proxy and ground truth rewards increase together because the algorithm finds genuinely good outputs near $x=2$.

However, at large N (> 256), the proxy reward keeps going up while the ground truth reward actually decreases. - this is reward hacking in action. This is because Best-of-N eventually discovers the boundary buggy regions that can be "exploited", which are not truly good outputs (around $x=2$). These boundary outputs score high on the proxy reward but are actually 0 according to ground truth.

This shows large N can increase the chance of reward hacking, i.e., the optimizer just finds and exploits flaws in the reward model, making the final performance actually degrades.

