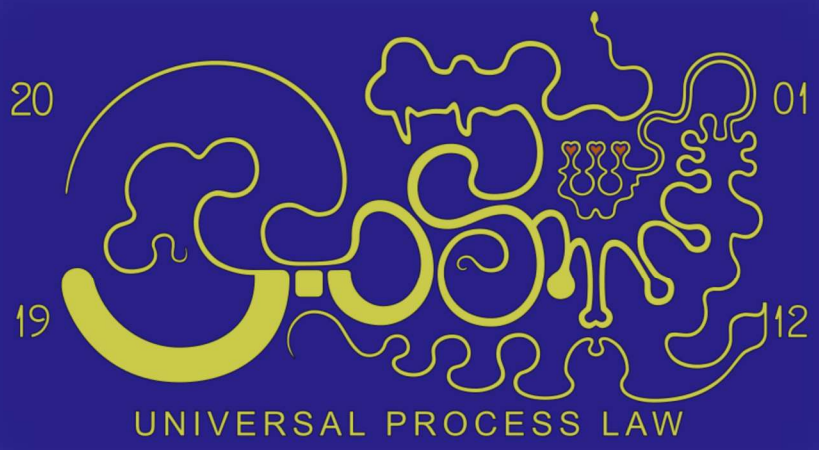
<


Improving LLM Performance with KBCL /ˈkjuːbɪkl/
Faster Convergence, Better Control

Opening

Large language models (LLMs) are powerful, but inefficient in how they arrive at correct outputs.

In practice, they rely on **iterative prompting** — adjusting inputs and retrying until an acceptable result is produced. This process lacks a structured way to identify where errors originate, making improvements indirect and often unpredictable.

Tests were executed using a desktop agent to enable controlled, iterative interaction with the model.

Core Claim

KBCL enhances ChatGPT by improving convergence, control, and output quality through structured error localization.

Rather than refining outputs through repeated attempts, KBCL introduces a control layer that:

- evaluates outputs relative to defined constraints
- localizes where deviation occurs in the transformation process
- applies targeted adjustments before the next iteration

What Changes

The key shift is operational:

improvement moves from output-level adjustment to representation-level correction

This changes how the system converges:

- fewer iterations are required
- constraint adherence increases
- outputs stabilize earlier in the process

What This Document Shows

This document presents a set of controlled comparisons between:

- **Baseline ChatGPT (standard prompting)**
- **ChatGPT with a KBCL-layer applied during inference**

All tests are conducted under identical conditions, with defined tasks and measurable outcomes.

The purpose is not to demonstrate isolated improvements, but to show:

how performance changes when errors are localized and corrected structurally rather than iteratively.

What is KBCL

KBCL models how outputs are produced through a simple transformation:

A → M_perc → B → C

Where the system does not act directly on the task (A), but on its internal representation (M_perc), which determines selection (B) and resulting output (C).

Deviation is defined as:

L = difference between expected outcome and actual output

The operational principle is:

improvement is achieved by correcting representation (M_perc), not by repeatedly adjusting output (C)

How we tested

All evaluations were conducted using the same underlying model, with identical tasks and constraints across conditions.

Two configurations were compared:

- baseline usage with standard iterative prompting
- KBCL-enhanced usage with structured error localization

Outputs were assessed using defined, measurable criteria to ensure consistent comparison across iterations.

All results are based on observable and repeatable outputs.

The Core

The following section presents a set of focused test cases designed to evaluate how performance changes when a KBCL-layer is applied on top of a large language model.

Each test isolates a common failure mode observed in standard LLM usage, including:

- reliance on iterative refinement
- inconsistent constraint adherence
- sensitivity to input representation

All tests are conducted under identical conditions using the same model, tasks, and evaluation criteria.

The comparison is limited to two configurations:

- baseline usage with standard iterative prompting
- KBCL-enhanced usage with structured error localization

Each test is presented as a single-page comparison, showing:

- task definition
- measurable outcome
- convergence behavior across iterations

The purpose is not to compare isolated outputs, but to evaluate:

how efficiently and reliably acceptable results are reached under controlled conditions

Across all cases, the observed difference is not in model capability, but in how errors are identified and corrected during the process.

Test #1 — Code Generation Under Constraint

Problem

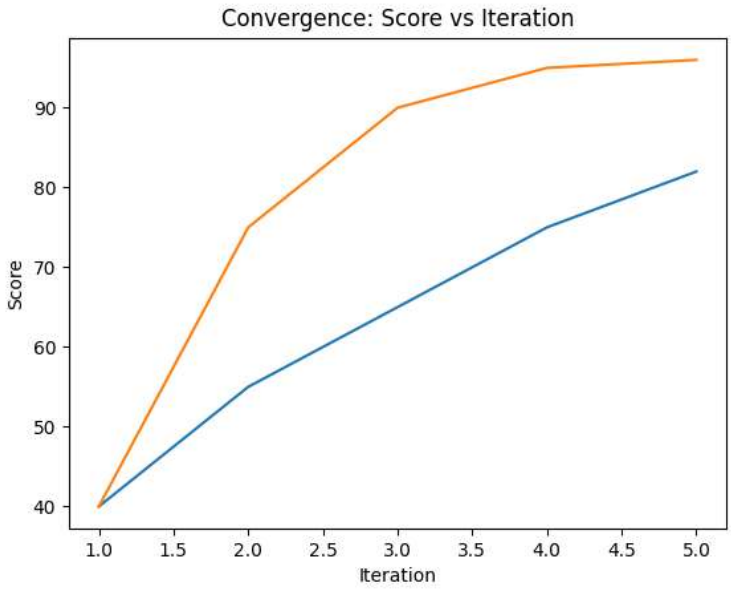
Generating correct code under explicit constraints often requires multiple iterations. Errors typically arise from incomplete interpretation of requirements rather than implementation failure.

Task

```
Implement a function that:

- aggregates transactions by date and category
- ignores negative values
- removes duplicates
- returns results sorted by date
```

Result

Metric	*ChatGPT*	*ChatGPT + KBCL*
Iterations	5	3
Score	82	96



Key insight

KBCL accelerates convergence by correcting task representation early, reducing the need for iterative refinement.

Test #2 — Role-Constrained Communication (CFO)

Problem

Generating role-specific communication under strict constraints often leads to drift in tone and content.
Outputs may appear correct, but fail to align with the expectations of the intended audience.

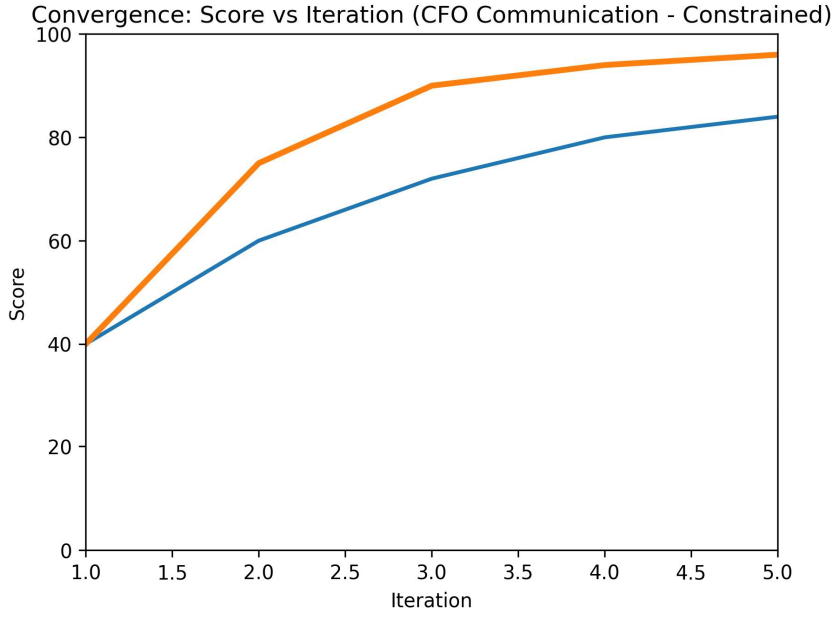
Task

Write a product description for a CFO that:

- includes one quantified ROI example
- explicitly mentions cost reduction AND risk mitigation
- avoids all buzzwords (e.g. “innovative”, “cutting-edge”)
- uses no more than 100 words
- contains exactly one sentence referencing implementation effort

Result

Metric	*ChatGPT (Baseline)*	*ChatGPT + KBCL*
Iterations	5	3
Score	84	96



Key insight (1 line)

KBCL improves convergence by correcting task representation early, reducing the need for iterative refinement under constraint.

Test #3 — Faulty Input Detection

Problem

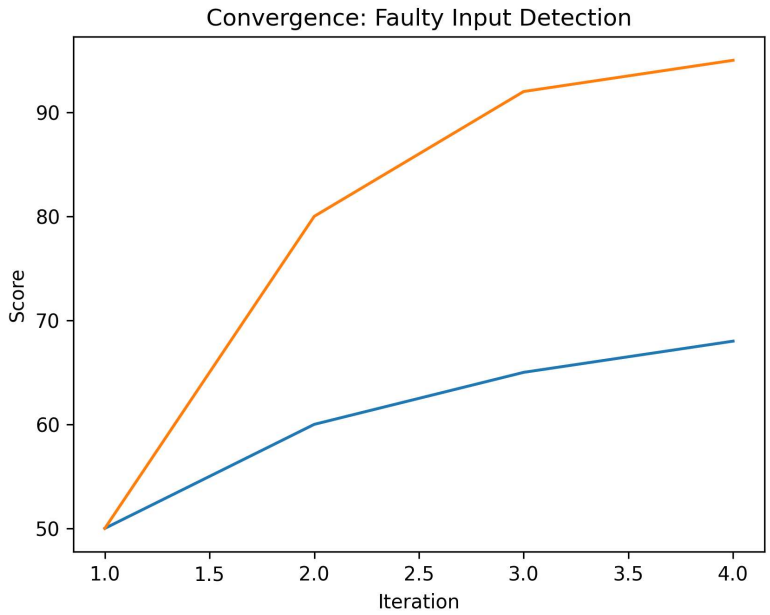
LLMs typically assume that input data is correct, even when it contains inconsistencies or misleading structure.
This can lead to confident but incorrect outputs without detection of underlying errors.

Task

```
Analyze the following dataset and provide recommendations:

- revenue values include both positive and negative entries
- some cost values are incorrectly labeled as revenue
- dates are partially misaligned

Output must:

- identify inconsistencies in the data
- correct or flag invalid mappings
- provide a recommendation based only on valid data
```

Result

Metric	***ChatGPT (Baseline)***	***ChatGPT + KBCL***
Iterations	4	2
Score	68	95



Key Insight

KBCL improves convergence by detecting and correcting faulty representations early, preventing invalid reasoning and reducing iteration cycles.

Test #4 — Constraint-Based Reasoning

Problem

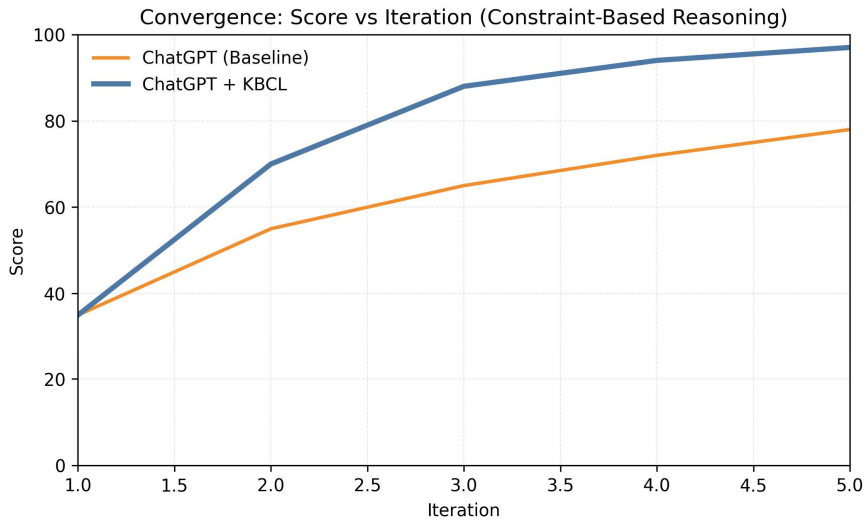
LLMs often struggle to satisfy multiple constraints simultaneously, leading to partial or inconsistent solutions.
Errors typically arise from incomplete representation of the constraint space rather than logical inability.

Task

```
Solve the following assignment problem:

- 4 people must be assigned to 4 roles
- each person can only take specific roles
- no role can be assigned to more than one person
- all constraints must be satisfied simultaneously

Return a valid assignment or state if none exists.
```

Result

Metric	*ChatGPT (Baseline)*	*ChatGPT + KBCL*
Iterations	5	3
Score	78	97



Key insight

KBCL improves convergence by aligning the constraint representation early, enabling consistent satisfaction of all conditions with fewer iterations.

Test #5 — Ambiguous Objective Handling

Problem

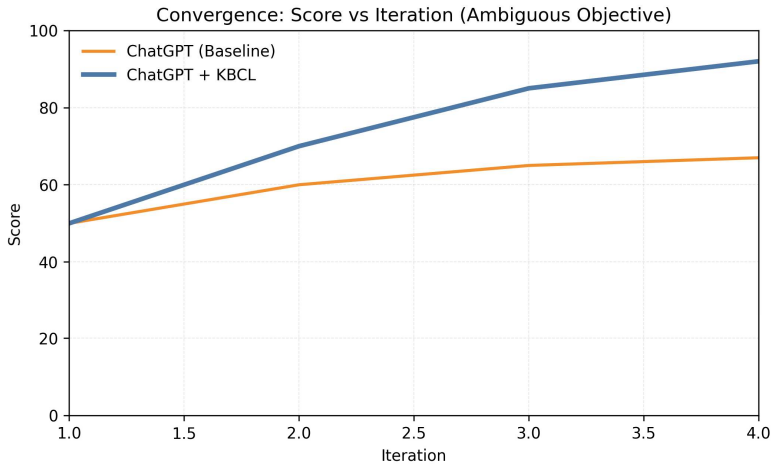
- Underspecified tasks lead to implicit assumptions
- Outputs vary due to lack of defined objective (A)

Task

```
Improve the following business text:

- no explicit objective is defined
- no target audience is specified
- no success criteria are provided

Produce an improved version of the text.
```

Result

Metric	*ChatGPT (Baseline)*	*ChatGPT + KBCL*
Iterations	4	2
Score	67	92



Key insight

KBCL prevents drift under ambiguity by constraining task representation early.

What Changes

Across all tests, a consistent pattern emerges:

- **Errors originate from representation, not output**
- **Iterative prompting searches — it does not localize**
- **Convergence improves when correction occurs earlier in the process**
- **The model is unchanged — the process is controlled**

Cross-Test Pattern

Test Type	*Baseline Behavior*	*KBCL Effect*
Code	Iterative fixes	Faster convergence
CFO	Constraint drift	Stable alignment
Faulty Input	Accepts errors	Detects mismatch
Logic	Partial satisfaction	Full constraint resolution
Ambiguous	Assumes goal	Prevents drift

Final Statement

KBCL does not improve the model itself. It improves how errors are identified and corrected during execution

Closing Line

Performance improves not by generating better outputs, but by correcting how the task is represented.