Dynamic-Angle Problem-Solving: A Control Policy for LLMs Under Missing Context

Author: Daniel Sanchez Diaz
Affiliation: Independent Researcher
Email: daniel@danielsanchezd.com

Abstract

Large Language Models (LLMs) often face queries with incomplete context, risking hallucinations or irrelevant outputs. I present the Dynamic-Angle Algorithm, a control policy addressing this by dynamically deciding when to answer, ask for clarifications, search external sources, or reframe queries. In this paper, I provide a comprehensive technical assessment of Dynamic-Angle in the context of current 2024–2025 LLM agent control research. I evaluate its design against a rigorous framework of criteria – including uncertainty quantification, confidence updating, action utility design, and optimal stopping policies – and compare it with state-of-the-art approaches such as ReAct, Reflexion, Self-RAG, DSPy, and RQ-RAG. I highlight how Dynamic-Angle formalizes a stake-sensitive confidence thresholding and myopic utility optimization to balance information gain against effort, and I analyze its theoretical soundness and limitations. My analysis finds that Dynamic-Angle's principled strategy aligns with many best practices (explicit uncertainty tracking, targeted tool use, stop conditions) and offers novel contributions like two-tiered context gating and stake-based decision thresholds, while also revealing open challenges in calibration, learning of utility estimates, and integration with emerging techniques (e.g. advanced uncertainty modeling and tree search planning). This work positions Dynamic-Angle within the evolving landscape of LLM control policies and outlines future directions to enhance its reliability and mathematical rigor for safe, efficient LLM deployment.

Introduction

Large language models have rapidly become capable assistants across many domains, yet controlling their behavior under missing or uncertain context remains a critical challenge. An LLM prompted with an underspecified question may either hallucinate an answer or endlessly ask clarifying questions. Recent research in 2024–2025 has explored various frameworks for LLM agent control, from prompt-based reasoning strategies to explicit policy algorithms. These include methods like ReAct [1], which intermixes reasoning steps with tool-use actions, and Reflexion [2], which uses self-generated feedback to iteratively refine the model's output. Other advances such as Self-Reflective Retrieval-Augmented Generation (Self-RAG) [3] and RQ-RAG [4] train models to decide when to retrieve external information or rewrite queries. Even software frameworks like DSPy [5] have emerged, allowing developers to script complex LLM behaviors (e.g. tool use, chain-of-thought, reflection) in a structured way. Despite this progress, there is a need for principled decision policies that can guarantee when an LLM should stop seeking more context and commit to an answer, especially in high-stakes scenarios where errors are costly.

This work examines the Dynamic-Angle Problem-Solving algorithm – a control policy designed specifically for LLMs operating with variable and often insufficient context. The Dynamic-Angle approach formalizes the decision-making process of an LLM into a series of steps with an internal confidence score guiding actions. Intuitively, the policy suggests that the model start with the narrowest inquiry possible given its prior certainty, widen the "angle" of exploration only as needed, and stop once a confidence threshold appropriate to the query's stakes is reached. This contrasts with more ad-hoc approaches and promises to reduce unnecessary tool use and follow-up questions.

In this paper, I present a comparative literature review and technical assessment of the Dynamic-Angle Algorithm. I evaluate how it addresses key aspects of LLM control identified by rigorous evaluation frameworks – namely: (1) Uncertainty Quantification and confidence calibration, (2) Confidence Update Mechanisms and criteria for belief revision, (3) Action Utility Formulation and decision geometry, and (4) Optimal Stopping Policies for answer commitment. I then cross-reference Dynamic-Angle with contemporary methods (ReAct, Reflexion, Self-RAG, DSPy, RQ-RAG) to situate its design in the landscape of LLM agents. My analysis emphasizes theoretical soundness and conceptual contributions, as no large-scale empirical benchmarks have yet been published for this algorithm.

The remainder of this paper is organized as follows. In Related Work, I review state-of-the-art LLM control strategies and advances in uncertainty modeling that form the context for Dynamic-Angle. In Method, I describe the Dynamic-Angle Algorithm in detail, including its mathematical formulation of confidence and utility. The Comparative Analysis section then evaluates Dynamic-Angle against both the evaluation criteria and alternative approaches, discussing strengths and weaknesses. I outline key Limitations of Dynamic-Angle, particularly in mathematical rigor and practical integration. In Future Work, I suggest directions for enhancing the approach (such as improved calibration, learning-based utility estimation, and integration with tree search). Finally, I conclude with remarks on Dynamic-Angle's place in the evolving research landscape and its potential impact on reliable LLM deployment.

Related Work

Reasoning and Tool-Use Agents

A prominent line of research has equipped LLMs with the ability to perform intermediate reasoning steps and invoke tools or external APIs. The ReAct framework by Yao et al. (2023) was an early example that prompted LLMs to produce both a chain-of-thought and explicit action commands in an interleaved manner [1]. By reasoning step-by-step and retrieving information (e.g. via a wiki browser API) when needed, ReAct showed improved factuality and fewer hallucinations on knowledge-intensive tasks [1]. Following ReAct, many variants and agent frameworks emerged. For instance, Toolformer (Schick et al. 2023) trained LLMs to insert API calls in generated text, and AutoGPT (2023) attempted fully autonomous multi-step planning using LLMs. These systems, however, often lacked an explicit notion of confidence or stopping criterion – they would continue acting until a task was done or a fixed step limit was hit. The Dynamic-Angle Algorithm shares ReAct's spirit of interleaving reasoning and actions, but it adds a formal decision criterion: a numeric confidence threshold for stopping when the answer is likely correct.

Self-Reflection and Memory

Another thread of work has focused on letting LLM agents iteratively improve their outputs via self-critique. Reflexion (Shinn et al., 2023) introduced an agent that, after producing an initial answer or making an attempt at a task, receives feedback (from the environment or a heuristic) and then generates a verbal self-reflection on what went wrong [2]. This reflection is stored in the agent's memory and influences the next trial. In this way, the agent doesn't update its weights but learns from linguistic feedback to avoid repeating mistakes [2]. Empirically, Reflexion greatly improved performance on tasks like code generation and decision-making by reducing error repetition. The Dynamic-Angle approach also leverages a form of reflection: if the algorithm detects that recent steps have yielded little new information or that new evidence contradicts its current assumptions, it triggers a reframing of the problem. This reframing is akin to a self-reflection where the agent explicitly acknowledges a potential mis-framing and attempts an alternative angle. Unlike Reflexion's multi-episode learning, Dynamic-Angle's reframing happens within a single query session and is preemptive (triggered by low information gain) rather than purely reactive to a failure, but both share the goal of escaping futile trajectories.

Retrieval-Augmented Generation

Integrating external knowledge retrieval with LLM generation (RAG) is now a standard technique to improve factual accuracy. Self-RAG (Asai et al., 2023) is a recent advancement that makes retrieval conditional and self-reflective [3]. Instead of always fetching a fixed number of documents, Self-RAG trains the model to decide when and what to retrieve, and introduces special reflection tokens that allow the model to critique the relevance of retrieved passages or the adequacy of its own answer [3]. This results in a controllable generation process: the model can dynamically adjust how much it relies on external information. Another related method, RQ-RAG (Ma et al., 2024), focuses on query refinement for retrieval. The model learns to iteratively rewrite or decompose queries, search with them, and then answer [4]. By training on curated data of query refinements, RQ-RAG outperforms static retrieval approaches and even strong baselines like ChatGPT on certain QA tasks [4]. In comparison, the Dynamic-Angle Algorithm does not train the base LLM to retrieve or critique; rather, it wraps around any LLM and decides when to call a retrieval tool (browser). Its action utility function explicitly weighs a Browse action's expected gain in confidence against its effort cost, so that external searches are only performed when likely to be beneficial. This philosophy aligns with Self-RAG's on-demand retrieval (avoid unnecessary fetches), but Dynamic-Angle's mechanism for this decision is a handcrafted utility formula rather than a learned policy.

LLM Control Frameworks

The fast-paced development in this area has also led to higher-level frameworks for composing and evaluating LLM behaviors. DSPy (2024) is one such framework that treats prompts, tools, and control-flow as modular components that can be combined into an LM program [5]. For example, DSPy provides a ReAct module, a ChainOfThought module, and optimizers for prompt improvement [5]. A developer can thereby declaratively specify that an LLM should follow a certain strategy (like thinking stepwise and using a tool) and easily swap in different strategies. While not a research algorithm per se, DSPy embodies the idea that structured control policies (like Dynamic-Angle) can be implemented and tested systematically. Indeed, one could implement the Dynamic-Angle Algorithm within such a framework by writing a policy loop that uses DSPy's primitives (for asking, browsing, etc.) and decision optimizers. This would facilitate comparisons between Dynamic-Angle and other policies in a unified environment.

Uncertainty Quantification for LLMs

A distinctive feature of Dynamic-Angle is its use of an internal confidence score C_t that is updated as new evidence is gathered. This reflects a broader trend in recent LLM research: modeling and leveraging the model's uncertainty. Traditionally, language models output probabilities for tokens, but these are not well-calibrated indicators of answer correctness or confidence in complex outputs. Thus, uncertainty quantification (UQ) methods have been proposed to estimate how much an LLM "knows that it doesn't know." One approach is to use multiple responses or an ensemble of models and measure agreement or variability. For example, Self-consistency decoding (Wang et al., 2022) samples many chain-of-thoughts and looks at answer consistency as a proxy for confidence. In 2024, researchers began designing UQ specifically for long-form text: LUQ (Long-text UQ) is a novel sampling-based framework that segments a long response and computes uncertainty for each part, aggregating to predict factuality [6]. Zhang et al. (2024) showed that LUQ's uncertainty scores correlate strongly with factual correctness in long answers and that choosing the answer with the lowest predicted uncertainty (an ensemble method) can improve overall factuality [6].

Another approach is to incorporate semantic understanding into uncertainty measures. MARS (Meaning-Aware Response Scoring) (Bakman & Yaldiz, 2024) introduced a scoring function that weights each token's contribution to the answer's meaning when estimating uncertainty [7]. Rather than treating all token prediction probabilities equally (which can overly penalize longer answers), MARS assesses which parts of the answer are critical and computes a confidence score that better reflects semantic correctness [7]. Techniques like MARS and LUQ represent the cutting edge of UQ for LLMs, offering multi-dimensional views of uncertainty (e.g., semantic uncertainty vs. knowledge uncertainty).

Dynamic-Angle's confidence tracking is comparatively simple: it starts from a prior based on query type and updates C_t in small increments or decrements based on evidence. There is no complex uncertainty model; the updates are heuristic (e.g. $+\Delta$ for confirming evidence, $-\Delta$ for contradictions) and the confidence is essentially a subjective probability. In my analysis, I will discuss how this choice, while pragmatic, could benefit from the more principled UQ techniques above. Ideally, a future Dynamic-Angle implementation might incorporate calibrated confidence estimators (ensuring the internal C_t corresponds to true success likelihood) or use criteria like expected calibration error to adjust C_t updates. The evaluation framework specifically calls for decomposing epistemic vs. aleatoric uncertainty – something not explicitly done in Dynamic-Angle, but conceptually the algorithm's design accounts for epistemic uncertainty (lack of knowledge) by allowing browsing/asking, and for ambiguity by allowing reframing.

Optimal Policy Design and Geometry

In reinforcement learning and planning literature, there are established tools for designing decision policies, such as Upper Confidence Bound (UCB) for balancing exploration vs exploitation, and Monte Carlo Tree Search (MCTS) for exploring decision branches. Some very recent works have tried to bring these ideas to LLM agents. For example, Zhang et al. (2024) combined an LLM with MCTS for solving complex math problems: the MCT-Self-Refine (MCTSr) algorithm treats each reasoning step as an action and uses an improved UCB formula to decide which partial solution path to explore next [8]. By iterative selection and backpropagation of a reward (problem solved or not), their LLM was able to find correct solutions more reliably than greedy decoding. Another work, Wu et al. (2024), studied when tree search is useful for LLM planning, concluding that for tasks requiring deep reasoning, a search can significantly help, whereas for straightforward tasks it may be overkill. The Dynamic-Angle Algorithm currently does not perform multi-branch search – it follows one line of inquiry at a time, making myopic decisions. However, its design does borrow the spirit of UCB: the utility $U(a|D_t)$ can be seen as an upper confidence bound on value for action a in that it inflates expected gain by an independence factor $\iota(a)$ (to favor exploring independent evidence) and deflates by cost. This is reminiscent of how UCB adds an optimism term to encourage exploring less-visited actions. Additionally, I can interpret Dynamic-Angle's notion of "widening the angle" as exploring a broader portion of the solution space manifold. The evaluation framework suggests checking if geometric interpretations are more than metaphor; in Dynamic-Angle, the term "angle" is indeed a metaphor, but one could imagine embedding the query in a semantic vector space and literally defining an angle of exploration. For now, the algorithm's "policy geometry" is implemented via discrete triggers (Micro vs Full context ask gates, reframing pivot), not via continuous geometric transformations, which keeps it straightforward but leaves room for more sophisticated geometric or tree-based planning in the future.

Method: The Dynamic-Angle Algorithm

The Dynamic-Angle Problem-Solving algorithm is a control policy layer that wraps around an LLM, guiding its interaction with the user and external tools. It proceeds in iterative steps, maintaining a state of knowledge and confidence. Here I describe its key components and decision rules.

Problem Setup and Notation

Let Q be the user's query and D the set of available evidence or context at a given time (initially, D_0 might include any user-provided context or just be empty). The algorithm assumes two categorical labels for the query: a stake level s and a query class k. The stake level $s \in \{\text{low}, \text{med}, \text{high}\}$ reflects the consequences of error – for example, a casual question vs. a high-stakes decision (medical or legal advice). The query class k characterizes the nature of information needed, such as: deterministic (e.g. math problem with a fixed answer), stable-external (open-domain fact that likely has a static answer), time-sensitive (answer may have changed or requires current data), personal/private (user-specific information not in the model's training data), or unobservable-now (cannot be answered currently by anyone). These classifications serve to initialize the system's confidence and strategy.

I denote by $C_t \in [0,1]$ the internal confidence after t steps. This is the algorithm's estimate of the probability that it can now answer Q correctly. At initialization ($t = 0$), Dynamic-Angle sets a starting confidence C_0 based on the query class via a prior $p(k)$. Intuitively, if Q is a simple closed-form question (deterministic), the model is nearly sure it knows the answer ($C_0 \approx 1.0$). If Q asks for a factual lookup (stable-external), C_0 might be moderately high (e.g. 0.8) assuming the model probably saw the fact in training. For time-sensitive queries, C_0 is lower (the model may be outdated, say 0.4). Personal/private queries get a very low prior (0.2 or less) since the answer likely isn't in training data; and for unanswerable-now questions C_0 can be near 0.0 by definition. These are example values; I suggest specific priors like: deterministic 1.0, stable external 0.8, time-sensitive 0.4, personal 0.2, unobservable 0.

Next, a confidence threshold T is set based on the stake level: $T = \tau(s)$, where for instance $\tau(\text{low}) = 0.60$, $\tau(\text{med}) = 0.80$, $\tau(\text{high}) = 0.95$. This T is the target confidence at which the agent is comfortable finalizing an answer. High-stakes queries demand near-certainty (95%), whereas low-stakes ones can tolerate more uncertainty (60%). The combination of priors and thresholds means that for a low-stakes, deterministic question, I might start with $C_0 = 1.0$ which already exceeds $T = 0.6$, so the model could answer immediately (with a note of assumptions). Conversely, a high-stakes, personal question might start at $C_0 = 0.1$ and need to reach $T = 0.95$ – requiring many steps of clarification or evidence gathering if possible.

Action Space

At each step t, the algorithm must choose an action a_t to improve its knowledge or conclude the interaction. The defined action space includes: Answer (provide a final answer to the user), Browse (search external sources for information, e.g. via a web tool), Ask (pose a follow-up question to the user – Dynamic-Angle distinguishes a "Micro" ask vs. a "Full" ask, explained below), Cross-check (verify a piece of information from a second source), Temporal check (verify if something has changed or if current date is relevant), or Reframe (rephrase or reconceptualize the query in light of a suspected misunderstanding). Not all actions are always available; for example, if the conversation or user settings disallow follow-up questions, the Ask actions would be removed from the set.

Utility Function for Actions

Dynamic-Angle introduces a quantitative utility model to select actions rationally. Let D_t be the evidence available at step t. For each possible action a, the algorithm estimates the expected confidence gain if that action is taken, denoted $E[\Delta C \mid a, D_t]$. It also considers the cost of the action, $cost(a)$, which abstractly represents the effort or latency (for instance, browsing might have a higher cost than asking a single question, which in turn has higher cost than simply answering without extra steps). Additionally, each action is assigned an independence weight $\iota(a) \in [0,1]$ that represents how independent the evidence from this action is likely to be from what's already known. The motivation here is to favor actions that bring new, independent information – for example, browsing a new source might have $\iota \approx 1$ (completely new evidence), whereas asking the user might yield information that overlaps with what the model already considered, so ι might be lower. Using these, the myopic utility of action a in state D_t is defined as:

$$U(a \mid D_t) = \frac{E[\Delta C \mid a, D_t] \cdot \iota(a)}{cost(a) + \varepsilon}$$

with a small $\varepsilon > 0$ to avoid division by zero. This formula essentially computes a reward-rate or efficiency score: how much expected confidence increase can I get per unit cost, adjusted by novelty of information. The algorithm then chooses the action with highest utility: $a_t = \arg\max_a U(a \mid D_t)$. Notably, this is a greedy, one-step lookahead policy – it does not explicitly plan multiple steps ahead, but it is designed to heuristically approximate an optimal strategy by always doing the most cost-effective step first. For instance, if the model is pretty sure of the answer and C_t is already high, the Answer action might have a high utility (moderate ΔC because answering confirms the solution, with minimal cost). If the model is uncertain and a piece of information is clearly missing, an Ask or Browse that could significantly raise C will have higher utility despite some cost. The independence factor $\iota(a)$ ensures that if an action's information is likely redundant (low independence), its utility is downgraded – this prevents, say, asking the user for information that is basically already present in context.

Confidence Update Rules

After executing action a_t, the algorithm observes some evidence e_t. Evidence might be a snippet of text from a web search (if a_t was Browse), an answer or clarification from the user (if Ask), a cross-check result, etc. The evidence is added to the context: $D_{t+1} = D_t \cup \{e_t\}$. Then the confidence is updated:

$$C_{t+1} = \text{clip}(C_t + \delta_t, 0, 1)$$

where δ_t is the increment based on the new evidence. The update rule is formulated qualitatively: if e_t strongly confirms the hypothesis or fills a crucial gap, then $\delta_t > 0$ (confidence goes up). If e_t is weak or ambiguous, then $\delta_t \approx 0$ (confidence unchanged). If e_t contradicts the current hypothesis or indicates the query is unanswerable, then $\delta_t < 0$ (confidence drops). All updates are kept small to be cautious, and the result is clipped to stay within [0,1]. This mechanism is intentionally heuristic; it requires the algorithm (or the underlying LLM) to interpret the new evidence and judge its impact on confidence. In practice, one could imagine the LLM itself producing a summary or score of how strongly the evidence answers the question or changes its belief, and map that to a δ_t. This part is not fully deterministic – it's a policy suggestion for how to adjust confidence.

Stopping Criteria

Dynamic-Angle defines clear rules for when the loop of actions should terminate and the agent should produce a final answer (or a final decision that it cannot answer). The primary stopping condition is confidence adequacy: if at any step t, the confidence C_t meets or exceeds the threshold T ($C_t \geq T$), then the agent is ready to answer. It will formulate a final answer to Q and present it to the user, including any necessary caveats or assumptions if some context was missing. (Indeed, the algorithm suggests that if it had to assume something due to missing info, it should state those assumptions in the answer for transparency).

Secondary stopping conditions handle cases where continuing is unlikely to help. One such rule is: if the expected marginal gain of any action is below a small threshold η (for example, $\eta \approx 0.10$ in confidence) – meaning no available action is expected to improve confidence by more than 0.1 – then it may not be worth further exploration. In this case, the policy would stop and give its best attempt answer, while explicitly noting remaining uncertainty. This is important: rather than loop endlessly with negligible gains, it's better to end the session and inform the user of the doubt (e.g. answer with "Based on what I have, I'm only somewhat confident; here's my best answer and what's still unclear."). Another stopping criterion is a timebox or interaction limit – if a predefined number of turns or time duration has passed, the agent should stop to avoid frustrating the user. Finally, if the agent cannot proceed because a needed piece of evidence is not accessible (e.g. a paywalled document or a physical measurement), it should stop and explicitly request that the user provide that essential artifact. This way, control is handed back to the user when the agent is blocked.

Two-Speed Context Gate

A novel aspect of Dynamic-Angle is its strategy for asking the user for more context. I observed that bombarding the user with a long list of questions can be overwhelming, but asking too little might not resolve the ambiguity. I propose a two-tiered approach to querying the user. By default, the agent uses a Micro Gate: it asks for only the 2–3 most critical pieces of information that would most increase its confidence, plus perhaps one safety check (e.g. "Let me know if any aspect has high stakes or special requirements"). This micro inquiry is designed to be answerable in one user response. If the user cannot provide all items, the agent is instructed to prioritize and indicate which single item would be most useful. Only if this minimal query comes back insufficient (or if from the start it's clear the query is high-risk or very underspecified) does the agent escalate to a Full Gate: a more exhaustive request for all relevant missing information. The Full Gate might be phrased as: "To ensure a correct answer, I will need the following details: …" listing every essential parameter. The dynamic between Micro and Full gate ensures that the agent is conservative in asking – it tries the lightweight clarification first, and only goes for a full clarification if necessary. This aligns with user experience best practices and keeps the conversation efficient.

Reframing Mechanism

Despite careful planning, the agent might reach a point where progress stalls. Dynamic-Angle includes a safeguard: if two consecutive steps yield very low confidence updates (essentially $|\delta_t| < \delta_{\min}$ for two t in a row) or if new evidence directly conflicts with the current interpretation of the query, then the agent should consider that it may be approaching the problem from the wrong angle. In such a case, the agent triggers a reframing: it generates an alternate hypothesis or way of looking at Q (call this F' a new frame for the question), explicitly states this to the user (to confirm or just to signal its line of thought), and then selects one discriminative test for the new frame. For example, if the query was "What is the safest way to invest money?" and the agent has been discussing low-risk bank accounts with little progress, a reframing could be: "Perhaps I should consider that 'safest' might mean guaranteed no loss. Let me check something: Did you perhaps mean in terms of not losing principal at all, like government bonds or insured accounts?" – followed by an action to gather info on that. Reframing helps avoid fixation on a wrong assumption. It effectively resets C_t partially (since a new frame often implies starting over on some aspects) but the algorithm can retain any evidence that is still relevant under the new frame. Only one reframing is done at a time, after which the process continues as normal (choose next action, etc.). This feature bears similarity to self-reflection in Reflexion and to query decomposition in some RAG systems, but is uniquely triggered by low information gain, indicating stagnation.

Summary of Algorithm Flow

Putting it all together, the Dynamic-Angle Algorithm operates as follows:

1. **Initialization**: Determine query class k and stakes s. Set initial confidence $C_0 = p(k)$ and threshold $T = \tau(s)$. If critical context is missing and follow-ups are allowed, initiate the Context Gate (Micro by default) to ask the user for key details.

2. **Immediate Answer Check**: If $C_0 \geq T$ (the model's prior knowledge already meets confidence requirement), the agent may answer immediately, while explicitly stating any assumptions it is making due to missing context.

3. **Iterative Loop**: If not confident enough, enter loop:
   - Compute the utility $U(a|D_t)$ for each possible action a (excluding disallowed ones) based on current evidence D_t.
   - Select the action $a_t = \arg\max U(a|D_t)$ and execute it.
   - Observe new evidence e_t, update the evidence set and compute δ_t, then update confidence $C_{t+1} = \text{clip}(C_t + \delta_t)$.

4. **Stopping Check**: After each update, if $C_{t+1} \geq T$, stop and output the answer. If not, check other stopping criteria: if $\max_a E[\Delta C|a, D_{t+1}] < \eta$ (diminishing returns) or if a predefined turn/time limit reached, then stop with a best-effort answer (and mention the uncertainty). If the agent cannot proceed without user-supplied data (e.g. needs a document), stop and ask for it (this can be seen as a deferred Full Gate).

5. **Reframing Check**: If two iterations have passed with negligible δ or a contradiction arose, consider the current approach possibly flawed. Propose a reframed query or alternate interpretation F' to the user, and perform one focused check (action) under this new frame. Then continue the loop at step 3 with updated context.

6. **End of Loop**: The loop continues until a stopping condition triggers. Upon stopping with an answer, if confidence is below the ideal threshold due to an enforced stop, the answer should include an advisory of the open uncertainties (for honesty and safety).

Assurances and Trade-offs

I argue that this policy yields fewer unnecessary clarification turns and avoids unbounded tool use by setting explicit thresholds and stops. Reframing prevents the agent from getting stuck on a single line of reasoning. However, I acknowledge some trade-offs: for low-stakes straightforward queries, this approach imposes a bit of overhead (all the machinery might be overkill if a direct answer would do). There's also a risk of misclassification of stakes or query type – if those are set incorrectly, the priors or thresholds could be suboptimal. And the confidence update rules are heuristic, lacking a deep Bayesian or learned model, which might sometimes misjudge the evidence impact. These issues are mitigated by logging provenance and assumptions (so that a human or oversight process can audit how the answer was obtained). The algorithm's built-in evaluation metrics suggest it should be assessed on metrics like: number of turns to resolve a query (fewer is better), first-answer accuracy in low-context cases (improvement expected), error rate on high-stakes queries (should be very low), frequency of unnecessary tool calls (should decrease), and appropriate use of "I need more data" responses (should be low but non-zero, meaning the agent asks for data only when truly needed). These align with the goal of making the LLM both efficient and cautious.

In summary, Dynamic-Angle provides a structured approach for LLM control, combining ideas of threshold-based stopping, utility-guided action selection, and adaptive querying. In the following section, I analyze how well this design stands up to the ideals laid out by recent research and how it compares to alternative methods addressing similar problems.

Comparative Analysis

I now assess the Dynamic-Angle Algorithm along several dimensions, correlating with both the evaluation criteria mentioned earlier and comparisons to other contemporary LLM control approaches. The analysis is organized into thematic sub-sections for clarity.

Uncertainty Quantification and Confidence Calibration

Internal Confidence vs External UQ: Dynamic-Angle's use of an internal confidence score C_t throughout its decision process is a noteworthy feature that sets it apart from many earlier agent frameworks. ReAct and similar chain-of-thought agents did not maintain an explicit probability of correctness at each step – they relied on the model's reasoning and prompts to implicitly decide when it "felt" done. In contrast, Dynamic-Angle formalizes this with C_t and threshold T, which is more akin to how a human might self-monitor confidence or how a probabilistic model would propagate belief. However, one must examine how well-calibrated or rigorous this C_t is. The algorithm treats C_t in a somewhat subjective manner: initial values are guesswork based on query type, and updates δ_t are heuristic. From a strict uncertainty quantification perspective, this could be seen as a weakness – there is no guarantee that, say, $C_t = 0.8$ actually corresponds to an 80% chance of being correct. The evaluation framework emphasizes calibration (e.g., requiring expected calibration error < 0.1 for reliable confidence) and decomposition of uncertainty sources. Dynamic-Angle does not explicitly separate epistemic uncertainty (model knowledge limits) from aleatoric uncertainty (ambiguity/inherent randomness), whereas some research recommends doing so. For example, an answer might be uncertain because the question itself is ambiguous (aleatoric) even if the model knows facts well; Dynamic-Angle's approach to that would be to ask clarifying questions (Micro Gate), which is appropriate, but it does not label the uncertainty type formally.

On the positive side, Dynamic-Angle's confidence tracking aligns with the growing recognition that LLMs should know when they don't know. It provides a mechanism for the model to say "I'm not confident enough to answer yet," which is crucial for safe AI deployment. Compared to Self-RAG, which also gives the model a way to reflect on whether it has sufficient info, Dynamic-Angle's confidence C_t is a simpler scalar, while Self-RAG used more implicit signals (like special tokens to indicate the need for retrieval). The scalar approach is interpretable – one can expose C_t or its threshold logic in explanations to users, thereby increasing transparency. If I consider advanced UQ methods: LUQ and MARS could potentially augment Dynamic-Angle. For instance, rather than relying purely on a heuristic δ_t, the algorithm could use a MARS score to adjust confidence: if the LLM's answer has certain tokens with low contribution or if multiple answer variations disagree on key points, reduce C_t. Likewise, LUQ's idea of combining multiple perspectives (semantic similarity vs knowledge-based consistency) could inform the confidence: if both semantic phrasing and factual content seem stable across different attempted answers, then be more confident.

Confidence Update Mechanisms: Another perspective is how Dynamic-Angle's confidence updating compares to traditional belief update frameworks. In a Bayesian sense, each piece of evidence could be thought of as updating a posterior probability. Dynamic-Angle doesn't provide formulas for δ_t; it entrusts the LLM or a heuristic to generate it. This is somewhat risky, as it could lead to non-monotonic or oscillatory confidence if not careful (imagine contradictory evidence followed by confirmatory evidence – does it exactly cancel out or overshoot?). The algorithm does include clipping to [0,1] and presumably would choose small $|\delta|$ to avoid wild swings. An ideal approach might enforce something like a Lipschitz continuity in updates – the framework even mentions a criterion that $||C_t - C_{t-1}||$ should be bounded relative to input changes for stability. Dynamic-Angle doesn't explicitly ensure Lipschitz continuity or any convergence proof (a red flag noted is the absence of convergence guarantees for iterative updates). It is conceivable that if evidence keeps coming in small drips (δ never pushing C over T but always positive), the algorithm could loop for many iterations; the η threshold for marginal gain is meant to cut that off, but it's heuristic as well.

By contrast, consider Reflexion: it doesn't maintain a continuous confidence, but it does have a notion of success/failure per trial. Over multiple trials, success rate improves, which could be seen as increasing confidence in the next trial's correctness. Reflexion also doesn't quantify uncertainty, but it uses the outcome (task success or failure) to adjust behavior indirectly. Dynamic-Angle tries to do this within a single trial by synthesizing internal feedback (e.g. after a cross-check, it might reduce confidence if an inconsistency is found). I might say Dynamic-Angle internalizes a reinforcement signal (like success/failure) into its C_t continuously, whereas Reflexion externalizes it across runs.

Thresholding and Output Quality: The choice of thresholds (0.6, 0.8, 0.95) in Dynamic-Angle is interesting to compare with stop criteria in other systems. For example, in many question-answering pipelines, a system might abstain if confidence (often learned from a classifier or calibrated probability) is below some threshold. Dynamic-Angle builds this in explicitly per stake level. This can drastically reduce high-stakes errors because it essentially changes the decision threshold for outputting an answer. If I consider GPT-4's approach (OpenAI's system card, 2023) – it doesn't have a formal threshold, but it is often instructed to say "I'm not sure" for uncertain answers. Dynamic-Angle gives a more structural way to determine that uncertainty. This resonates with the concept of selective prediction in ML, where a model can choose to say "I don't know" for some fraction of cases to improve reliability on the ones it does answer. The algorithm's "appropriate 'I need more data' rate (target: low but non-zero)" aligns with that: it should not overuse abstention, but it should do it when truly needed.

Decision Policy and Action Selection

Utility-Driven Action Choice: The core of Dynamic-Angle's policy is the utility function $U(a|D_t)$ defined earlier. This formula effectively encodes a prioritization of actions: those that give a big confidence boost cheaply and with new info are chosen first. Let me analyze this in light of other agent policies:

- **ReAct and others via prompting**: In ReAct, the decision of whether to "Act" (e.g. call a tool) or to stop and give an "Answer" is handled implicitly by the prompt format. The LLM generates either a thought and an action or a thought and the final answer. It was observed that ReAct tends to call the tool if the chain-of-thought thinks something is unknown. However, there is no explicit cost accounting or expected gain calculation – it's all learned from a few prompt examples or inherent knowledge. This can be brittle; e.g., the model might overuse the tool for simple queries or underuse it when needed. Dynamic-Angle's approach imposes a rational structure: even if the underlying LLM is not very savvy about when to tool-use, the control policy will force a comparison (maybe using estimates coded by the developer). For example, one could calibrate $E[\Delta C|\text{Browse}]$ to a moderate value and $cost(\text{Browse})$ to a slightly higher value than $cost(\text{Answer})$, ensuring trivial questions don't trigger a web search. This kind of tunable trade-off is much harder to enforce in pure prompt-based methods.

- **Multi-step Planning vs Greedy**: One might question the greedy nature of Dynamic-Angle's action selection. Could it get stuck in a local optimum? For instance, suppose two steps of browsing the web (action A twice) would yield a large confidence gain in total, but the first browse alone yields only a tiny clue (so $E[\Delta C|A]$ appears small). Meanwhile, asking the user (action B) might give a moderate immediate gain. A myopic strategy would choose B, potentially exhausting user's input, when the optimal plan was A twice. This is a classic exploration vs exploitation issue. MCTS-based agents like MCTSr explicitly explore sequences of actions and evaluate the end result. Dynamic-Angle does not simulate sequences; it assumes a roughly monotonic and additive benefit of actions. In cases where synergy of actions is important (only after doing X does Y become useful), the greedy approach may falter. However, in typical informational queries, synergy is limited – any single info source can be considered independently useful or not. The reframing mechanism partly helps if the initial greedy choices led to a dead-end.

- **Exploration bonus ($\iota(a)$)**: The inclusion of the evidence independence weight is conceptually similar to exploration bonuses in reinforcement learning (like UCB's bonus for rarely taken actions). If the agent has already, say, read multiple web articles (so further browsing might be redundant) but hasn't asked the user directly yet, then $\iota(\text{Ask})$ would be higher relative to $\iota(\text{Browse})$ now, encouraging a shift of strategy. This is clever and not something seen in, for example, Self-RAG or RQ-RAG – those either retrieve more blindly or have to rely on the model's learned behavior to not repeat the same query. Dynamic-Angle explicitly guards against redundant actions. This could be viewed as injecting domain knowledge: it knows that different sources of info have overlaps, so it wants diversity. A potential improvement could be making $\iota(a)$ dynamic: e.g., track how independent the last few evidence pieces were. If browsing different websites, initial ones might be independent but eventually you see repeated facts, so independence goes down.

- **Cost parameterization**: The notion of $cost(a)$ is an interesting design choice. In current LLM agents, "cost" might include response latency, API call expense, or user burden. Reflexion, for instance, doesn't formalize cost, but it implicitly incurs cost by doing multiple trials (time cost). RQ-RAG and Self-RAG operate within one query so they don't consider cost explicitly either. By including cost, Dynamic-Angle can be tuned for efficiency. For instance, if using a tool is slow or expensive, one can increase its cost to discourage overuse. If user follow-ups are considered undesirable beyond one question, one might set $cost(\text{Ask second time})$ very high. In effect, it's a way to encode a policy preference: e.g., prefer self-reliance (the model's knowledge) until confidence really drops. However, setting these costs and expected gains is nontrivial – it might require empirical data or at least careful thought. In absence of learning, these are hyperparameters the developer must choose.

Optimal Stopping and Answer Commitment

Threshold-based Stopping vs Other Criteria: Dynamic-Angle's main stop rule is straightforward: stop when confident. This rule, with stake-adjusted threshold, is an embodiment of an optimal stopping policy under uncertainty – reminiscent of settings like medical diagnosis or decision making where one decides to stop testing and make a decision when the probability of a correct decision is high enough to warrant the cost/risk trade-off. The presence of secondary stop conditions (η threshold on marginal gain, time limit) is to avoid pathological cases of never reaching the primary threshold. Let me compare with other approaches:

- **ReAct / basic LLM agents**: They typically stop when they have produced an "Answer" token in their output. That decision is left to the LLM's judgment, which could be based on subtle cues or exhaustion of reasoning steps. There isn't a guaranteed performance threshold considered. It might stop too early (hallucinate an answer) or too late (loop in thought). Indeed, Yao et al. noted that ReAct mitigated hallucinations by forcing tool use, but if the model wrongly believes it has enough info, it can still answer incorrectly. Dynamic-Angle's threshold aims to reduce that risk – if the model's prior (C_0) is low for a question, it won't answer until it has done due diligence (or it hits the alternate stop criteria, but then it will explicitly acknowledge uncertainty). This is a major safety improvement over naive agents.

- **Reflexion and multi-episode strategies**: Reflexion doesn't have a single episode stopping rule beyond task success or manual cutoff (e.g., maximum number of trials). It relies on repeated attempts. If an agent fails repeatedly but without any external feedback, Reflexion might not know when to give up. In a sense, Dynamic-Angle's η threshold for diminishing returns plays a similar role: it says "I've tried enough within this episode." If Reflexion were cast in a single session, it might incorporate a similar idea (like after N self-reflections with no progress, stop). So Dynamic-Angle's design anticipates the scenario of stagnation and gives a principled out.

- **Optimal stopping in QA systems**: There's literature on when to stop reading in machine reading comprehension – e.g., models that decide after reading enough passages whether they are ready to answer. Dynamic-Angle's context gate is conceptually similar: the Micro Gate asks for just enough info that the model expects to need. If the user's answer to those still leaves confidence low, the agent might ask more or proceed to search. The threshold T is essentially formalizing how "optimal" that stopping is relative to the cost of more info. In a high-stakes scenario ($T = 0.95$), the agent will be thorough, doing more steps until reaching near certainty, which is akin to lowering the risk of a wrong answer as much as possible. In low-stakes, it will cut off earlier to save effort. This risk-weighted stopping is unique among the methods I compare – others usually have a one-size-fits-all stopping criterion (or none explicit).
- **Expose residual risk**: Another commendable practice in Dynamic-Angle is that if it must stop early (due to η or time) and C is still below T, it doesn't just silently give an answer; it exposes the residual risk. For example, it might prepend "Disclaimer: I am not entirely sure about the answer because [...]." This is aligned with ethical AI considerations – it prevents false certainty. Many current systems don't do this unless explicitly prompted. ChatGPT, for instance, sometimes hedges if it's unsure, but not reliably. With Dynamic-Angle, hedging or warning is built-in when needed. This addresses a point in the evaluation framework about communicating uncertainty and broader impacts – a model that signals uncertainty can allow a human to decide whether to trust the answer or seek second opinions.

Limitations

While the Dynamic-Angle Algorithm introduces a structured and theoretically motivated approach to LLM control, it is not without significant limitations. I categorize these limitations into theoretical, practical/implementation, and scope limitations.

Heuristic Confidence and Lack of Theoretical Guarantees

Perhaps the most salient limitation is the heuristic nature of the confidence update mechanism. The algorithm does not derive δ_t from first principles; it relies on intuitive judgments of the evidence. This raises several concerns:

- **Calibration and Reliability**: There is no guarantee that C_t truly represents an accurate probability. If C_t is mis-calibrated, the threshold T may be reached too early or too late. The evaluation criteria explicitly warn against undefined behavior at confidence boundaries and missing normalization strategies. Dynamic-Angle does clip C_t between 0 and 1 and starts with somewhat normalized priors, but it lacks a formal calibration step (e.g., no mechanism like temperature scaling or isotonic regression to ensure C aligns with actual accuracy likelihood). In long sequences of updates, errors could accumulate (this touches on absence of error propagation analysis).
- **Convergence**: There is no proof that C_t will converge to 1 (or at least exceed T) if the answer is knowable and sufficient evidence exists. In the worst case, the agent might oscillate or plateau below T. The stopping criteria η is a heuristic fix, but that introduces an arbitrary parameter (0.1). Without convergence guarantees or at least a bound on needed steps, one cannot be sure of the algorithm's efficiency or termination on all inputs. In contrast, some MCTS or reinforcement methods might offer convergence under certain assumptions (e.g., eventually the tree search will find a solution with probability 1, or an RL policy will converge in the limit). Dynamic-Angle's semi-greedy approach does not come with such assurances.
- **Undefined Edge Cases**: If C_0 is extremely low (like for an unanswerable question) and no evidence can raise it, how does η handle that? The algorithm would stop with best-effort, which is fine, but one could imagine edge cases where evidence is contradictory – confidence could even go down then up. Does the algorithm risk a loop if evidence alternately raises and lowers C around the threshold? Possibly yes, if not handled, though the η rule might eventually cut it off when net gain is negligible.

Utility Function Simplifications

The utility $U(a)$ is crafted with a myopic assumption. This may fail to account for long-term benefits of certain actions as discussed. For instance, asking a question might have a small immediate expected gain but unlock a path to very high confidence (multi-step dependency). The greedy policy could miss such chains. Ideally, one would incorporate a lookahead or use something like a Q-value for actions (e.g., via dynamic programming or learning). The lack of multi-step planning could reduce optimality. Moreover, the formula uses a simple division by cost – which is linear. Maybe in reality, the relationship is nonlinear or context-dependent. The evaluation framework mentions checking if utility design satisfies properties like Bellman optimality or if reward shaping is properly bounded. Dynamic-Angle's utility is not derived from a Bellman equation, and it doesn't ensure, for example, that following the greedy policy maximizes the probability of reaching threshold with minimal cost. It's a reasonable heuristic but not proven optimal.

Additionally, the algorithm doesn't explicitly address exploration vs exploitation trade-off beyond the independence weight ι. If the estimates of $E[\Delta C]$ are uncertain, a Bayesian or robust approach might suggest exploring an action you're less sure about. Dynamic-Angle always takes the current argmax. In practice, since it's not learning on the fly, this might be fine, but it's not theoretically ideal if $E[\Delta C]$ itself has uncertainty.

Parameter Tuning and Dependence on Human Expertise

Many of the values in Dynamic-Angle (priors $p(k)$ for classes, thresholds $\tau(s)$, the minimal gain threshold η, even $\iota(a)$ values and cost estimates) are set by the designer. These might not generalize across domains or tasks without retuning. For example, $\tau(\text{low}) = 0.6$ might be acceptable for casual questions, but if the system is deployed in a customer support chatbot, even low stakes might require higher accuracy to avoid misinformation. Or the class priors: "time-sensitive external" being 0.4 might hold for general knowledge, but for a tech news domain, maybe the model's prior is worse and should be 0.2. Essentially, the algorithm requires expert judgment or empirical calibration to set these parameters. This is a limitation because it adds overhead to adapt the system to new settings, and mistakes in these settings could impair performance. By contrast, a learning-based policy might automatically adjust to domain difficulty (e.g., if the model's training data is weak in an area, it learns to ask more).

Resource Intensity and Overhead

While Dynamic-Angle is aimed at efficiency (with its utility balancing gain and cost), the policy itself can introduce overhead in cases where it might not be needed:

- **Low-stakes simple queries**: I note a trade-off: minor overhead for low-stakes deterministic tasks. If a user asks "What's 2+2?", a very low-stakes and deterministic question, an LLM would answer "4" immediately. Dynamic-Angle would classify it deterministic ($C_0 \approx 1$) and hopefully answer immediately too – which it likely would since C_0 would exceed even T_{low}. But consider a slightly more complex question where the model is actually fine to answer directly, but dynamic-angle doesn't know that and maybe asks a clarification or does a needless web search. Over-reliance on procedure could make the system slower or more annoying on trivial queries. Ideally, the class priors circumvent a lot of this (deterministic → no need to search), but if classification is off, the system might do extra work. This is a "false negative" scenario – failing to realize it already knows enough.
- **User Experience Issues**: There is a risk that the Dynamic-Angle agent might come across as overcautious or interrogative. For example, some users might expect an answer, not a follow-up question. If the Micro Gate triggers on too many queries (especially medium stakes ones where user might not expect a counter-question), it could reduce user satisfaction. This is a design trade-off: safety vs convenience. The algorithm is designed to err on the side of caution (which is good for high stakes), but it might need careful UX handling (like phrasing follow-ups politely, explaining why it needs to ask). The evaluation framework's human factors (human evaluation and broader impact) remind us to consider this: does the algorithm unduly burden the user or create new risks (like the user providing sensitive info as context because the agent asked, which could be a privacy issue)? Dynamic-Angle says to respect privacy constraints, but once it asks, the user might reveal private data. In a learned system, perhaps the model might avoid asking certain things; here it's rule-based – hopefully the designer thought of excluding obviously sensitive follow-ups.

Misclassification of Stakes or Query Type

The initial step of classifying the query into k and s is itself a potential point of failure. It might be done by the LLM or a separate classifier. If it mistakes a high-stakes query for low-stakes, T will be too low and it might answer when it shouldn't (e.g., treating a medical query casually). Conversely, if it flags too many things as high-stakes, it will often delay answers and provide heavy disclaimers, which could be unnecessary. This risk of mis-tiering was noted by the authors. They mitigate by logging assumptions, but that doesn't fully solve it. Some stake detection might be straightforward (explicit "medical, legal" keywords), but others could be subtle. This is an area where one might need a curated list or even a learned classifier to do it properly. So the pipeline's correctness hinges on a potentially brittle classification step.

Limited Learning/Adaptation

Dynamic-Angle as described does not learn from experience. If it makes a wrong call in one interaction (e.g., it trusted some evidence that turned out false), there is no mechanism to adjust its future behavior, aside from if the developers analyze logs and tweak parameters. Reflexion and self-refine approaches incorporate feedback loops to improve performance over time (Reflexion between trials, MCTSr uses some self-evaluation in search, etc.). A dynamic-angle agent will perform the same way each time given the same conditions. In rapidly changing domains, not learning could be a drawback. However, one could wrap learning around it (for example, use outcomes to adjust the priors or costs).

Scope of Applicability

The algorithm is clearly tailored to QA or conversational assistance tasks where missing context is the main challenge. It might not be directly applicable to tasks like story generation, code generation, or others that don't involve missing external info. That said, one could imagine analogues (e.g., for code generation, missing context might be API docs – dynamic-angle could then decide when to search documentation vs. just code). But this would require customizing action space and priors. So out-of-the-box, it's not a one-size-fits-all LLM policy. In contrast, something like ReAct is quite general (it can integrate any tool as needed just by prompt). Dynamic-Angle might need a bit more configuration per domain/toolset.

Future Work

The Dynamic-Angle Algorithm represents an important step toward safer and more effective LLM control, but it also opens many avenues for further research and improvement. I outline several directions for future work, drawing inspiration from the limitations identified and from emerging trends in the field.

Learning-Based Calibration and Adaptation

A top priority is to bring more rigorous learning and calibration into the Dynamic-Angle framework. One idea is to develop a calibration model that takes the LLM's internal signals (e.g., logits, retrieved evidence, chain-of-thought content) and produces a more accurate confidence score. Recent UQ toolkits like LUQ or frameworks in multi-dimensional UQ could be integrated. For example, one could generate multiple candidate answers with the LLM (using different prompts or randomness) and apply multi-dimensional UQ methods to derive a comprehensive uncertainty estimate. This estimate could serve as the initial C_0 or even as C_t updates, thus rooting the confidence in statistical measures rather than pure heuristics. Similarly, the MARS scoring technique could be used post-hoc on an answer draft to gauge confidence and decide if more steps are needed.

Another learning angle is to allow the agent to learn optimal $U(a)$ parameters. Instead of manually specifying $E[\Delta C]$ and $cost(a)$, one could simulate many QA scenarios (perhaps using a proxy environment or self-play) and use reinforcement learning to adjust these values so as to maximize some reward (like final accuracy minus penalties for delays). The algorithm could thus tune itself: e.g., learning that browsing tends to yield on average a 0.3 confidence boost in certain domains, etc. Caution is needed to maintain interpretability, but even a limited learning (like using bandit algorithms to learn which action often leads to success) could improve the utility estimates over time.

Enhanced Query Classification and Context Handling

The front-end of Dynamic-Angle – classifying query type k and stake s – could be extended and improved. A more fine-grained or learned classifier could be developed using training data of queries labeled by stakes. For instance, one could compile a dataset of user queries with annotations: trivial vs important, factual vs opinion vs personal, etc., and train a transformer to predict (k, s). This model could surpass simple keyword-based classification, thus reducing mis-tiering errors. Additionally, the set of query classes might be expanded or made continuous (e.g., predict a "knowledge availability" score instead of discrete k classes). If the classifier is uncertain, the system might default to assuming higher stakes or ask the user to clarify the intent/stakes.

Moreover, dynamic context gating could be refined. The Micro vs Full Gate concept is somewhat binary; future work could explore a multi-turn strategy where the agent can ask one question, get an answer, re-evaluate, and if still insufficient, ask another – essentially a loop of micro inquiries until a threshold of context completeness is reached. This would require criteria to avoid too many questions (perhaps similar η logic but for information gain from user). It might also involve asking the user in a conversation, e.g., "Is there anything else important I should know?" as a final check before proceeding to answer. This more interactive clarification process should be tuned by user-centered research to ensure it's helpful, not bothersome.

Integration of Advanced Tool Use and Multi-Step Reasoning

While Dynamic-Angle considered browsing and asking as actions, future versions could incorporate a wider set of tools and operations:

- **Knowledge Base and Memory**: The agent could maintain a memory or knowledge graph of facts gathered during the dialogue. Actions could then include querying that structured memory or doing consistency checks within it. This relates to the concept of "self-consistency" – ensuring the pieces of evidence in D_t do not conflict. If a conflict is found, an action could be to address it (maybe by asking which is correct or searching specifically about the discrepancy).
- **Computation Tools**: For queries requiring calculation or logical deduction, integrating with a calculator or Python interpreter (similar to Toolformer) would be useful. The policy then needs to decide when to call the calculator. This could be based on query classification (e.g., if k = deterministic math problem, a "calculate" action is available with high expected ΔC).
- **MCTS or Tree Search for Reasoning**: Future

research might attempt a hybrid of Dynamic-Angle with tree search. For instance, the agent could branch on different assumptions or frames and explore them in parallel (to a limited depth), using an approach like MCTSr's UCB selection to decide which branch to follow. If computational resources allow, this could overcome the myopia limitation. A simplified version might be to generate two possible reframes at a reframing point and follow both for one step to see which yields a higher confidence prospect, then commit to one. Essentially, a selective branching when ambiguity is high could outperform a single-threaded approach.

Policy Geometry Formalization

If the concept of "angle" is to be taken more literally, one could attempt to map the problem state to a vector and define actions as moving in certain directions. Perhaps using embedding models, the query plus context could be embedded, and different actions correspond to moving that embedding toward regions associated with known answers. For example, reframing might yield an embedding that is farther from the original but closer to some cluster of answerable queries. This is speculative, but with techniques like contrastive learning one could train an encoder such that the distance (or angle) between the query state and the correct answer representation decreases as confidence increases. Then an optimal policy might literally follow the gradient in that embedding space. This would give a more mathematical grounding to the "angle" idea and could provide new ways to choose and evaluate actions (like if browsing moves the state vector a certain amount, etc.). Such a direction would align with calls for more geometric and theoretical analysis of policies.

Formal Analysis and Verification

To increase trust in the algorithm for high-stakes deployment, future work should include formal analysis. For instance:

- **Bounding number of steps**: Can I derive an upper bound on steps in terms of some parameters (maybe information-theoretic measure of uncertainty)? A stochastic analysis could model C_t as a random process where evidence increments follow a distribution, and attempt to bound hitting time for threshold T. This could yield insights like "with high probability, the algorithm will finish in N steps if the answer is obtainable."
- **Worst-case scenarios**: Identify pathological cases and formally verify the algorithm handles them. One could use model checking or adversarial testing where the environment (user/web) is adversarial to see if the agent still stops gracefully. For example, if every browse returns irrelevant info ($\Delta C \approx 0$) but the agent keeps thinking maybe the next will help, does the η rule always catch it? These scenarios can be simulated to ensure the agent won't loop indefinitely or won't skip needed reframing.
- **Theoretical optimality in simplified model**: Perhaps consider a simplified setting (like a Bernoulli information gain model) to see if the greedy strategy approximates the optimal strategy. This could connect to literature on value of information and optimal experiment design, where one chooses queries to maximize information gain. Dynamic-Angle's heuristic is essentially doing that, but a more formal treatment might reveal under what conditions it is near-optimal or how to improve it (maybe adding a small probability to explore non-greedy actions yields better results, etc.).

User Studies and Human-in-the-Loop Refinement

Since Dynamic-Angle's success partly hinges on how users respond to its questions and clarifications, performing user studies will be invaluable. Future work should test the approach with real users: Do they appreciate the micro follow-up questions? Do they feel the agent is trustworthy when it says "I'm not fully confident"? Gathering feedback can guide tweaks: for example, maybe users prefer the agent to always explain why it is asking something ("I want to double-check I understand you correctly: [question]"). This could be built into the prompt style. Or one might find that certain types of questions annoy users – e.g., asking for too much personal info might be off-putting – so the agent could be adjusted to only ask those when absolutely necessary or to offer an explanation or alternative.

There is also scope for human-in-the-loop training. For instance, using reinforcement learning from human feedback (RLHF) to fine-tune either the base LLM or the meta-policy. One approach: have humans rate the interactions of dynamic-angle agent vs others on various criteria (usefulness, satisfaction, safety), and use these as rewards to refine either the action selection or the phrasing/prompting style of the agent. This could marry the hand-crafted policy with data-driven refinement.

Experimental Evaluation Framework

Finally, future work should also establish a rigorous evaluation framework for LLM control algorithms like Dynamic-Angle. This includes creating benchmark scenarios that test various aspects: some with missing info (to test clarification ability), some adversarial or misleading info (to test reframing and verification), a mix of stakes (to test threshold behavior), etc. Concretely building an evaluation suite where I can measure "turns to resolution", success rate, tool usage count, and calibration of reported confidence would greatly help quantify improvements. One might have to simulate user responses for consistency (perhaps using a human or a separate model to play the user role in providing partial info). Over time, such evaluation could drive an iterative design: if metrics show, for example, that the agent is often stopping at $C \approx 0.75$ for high-stakes when it should have continued, one knows to adjust either the threshold or the confidence estimation method.

Conclusion

In this work, I presented a comprehensive analysis of the Dynamic-Angle Problem-Solving algorithm for large language model control, situating it in the context of recent advancements in LLM agent research. Dynamic-Angle brings together a number of valuable ideas – stake-aware confidence thresholds, heuristic utility-based action selection, and adaptive context gathering – into a single policy framework. My comparative review showed that it addresses some of the shortcomings of prior approaches like ReAct, Reflexion, and retrieval-augmented strategies by providing explicit mechanisms to avoid hallucination (through evidence-backed confidence building) and to prevent endless reasoning loops (through optimal stopping criteria). In particular, the algorithm's emphasis on uncertainty quantification and decision-theoretic stopping represents a step toward safer AI: it demands a quantifiable level of confidence before action, especially when the stakes are high, and it is willing to say "I don't know enough" – a humility not often seen in large language models by default.

At the same time, my technical assessment highlighted that Dynamic-Angle is not a panacea. It currently relies on hand-tuned parameters and heuristic estimates, lacking the rigorous calibration and learning that state-of-the-art research in uncertainty and policy optimization provide. There are scenarios where a purely learned approach might adapt more flexibly, or conversely where a simpler chain-of-thought might suffice without the overhead of a control layer. The algorithm's effectiveness thus remains to be proven in practice, and I have enumerated various limitations – from potential misclassification of query stakes to the absence of formal guarantees – that should be carefully addressed.

I outlined a rich agenda for future work to improve upon Dynamic-Angle. This includes integrating modern uncertainty estimation techniques (such as LUQ and MARS) to ground its confidence measures in statistical reality, and possibly learning the utility function and other parameters from data to reduce reliance on manual tuning. Another promising direction is merging Dynamic-Angle's rule-based structure with the search capabilities of MCTS or the self-improvement of Reflexion, in order to handle more complex reasoning and to learn from mistakes over time. By doing so, I can aim for an agent that is both interpretable in its decision-making and optimal (or near-optimal) in its results – a combination that is highly sought after in trustworthy AI research.

In the broader landscape, Dynamic-Angle exemplifies the trend of moving beyond treating LLMs as one-shot or black-box responders, toward treating them as iterative problem-solvers with agency. It puts forth a structured policy that could be seen as an initial blueprint for LLM decision-making: first assess what you likely know, decide what you need, gather information methodically, and only answer when ready (or gracefully bow out if you cannot reach certainty). This aligns with human problem-solving strategies and with principles of optimal inquiry in fields like information theory and sequential analysis. As LLMs become more integrated into critical applications (from medical advice to legal consulting to autonomous research agents), such control policies will be indispensable to ensure reliability.

In conclusion, Dynamic-Angle is a conceptual advance that pushes LLMs one step closer to being not just fluent generators, but careful reasoners and decision-makers. Its comparative strengths and weaknesses, as dissected in this paper, provide valuable lessons for the design of future LLM control algorithms. I believe that by building on Dynamic-Angle's foundations – enhancing its mathematical soundness, broadening its capabilities, and rigorously evaluating it – researchers can develop next-generation LLM agents that are far more trustworthy, efficient, and aligned with user needs than those available today. The marriage of heuristic control policies with learning-based adaptability appears to be a particularly fertile ground for innovation. Ultimately, the goal is an LLM that can dynamically angle its approach to problems in a way that maximizes truth and utility while minimizing risk – a goal that stands to benefit not only the AI field but all users who rely on AI for informed and safe assistance.

References

[1] Yao, S., et al. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. *ICLR 2023*. arXiv:2210.03629.

[2] Shinn, N., et al. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning. *NeurIPS 2023*. arXiv:2303.11366.

[3] Asai, A., et al. (2023). Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. arXiv:2310.11511.

[4] Ma, X., et al. (2024). RQ-RAG: Learning to Refine Queries for Retrieval Augmented Generation. arXiv:2404.00610.

[5] Khattab, O., et al. (2024). DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines. arXiv:2310.03714.

[6] Zhang, C., et al. (2024). LUQ: Long-text Uncertainty Quantification for LLMs. *EMNLP 2024*.

arXiv:2403.20279.

[7] Bakman, Y. F., & Yaldiz, D. N. (2024). MARS: Meaning-Aware Response Scoring for Uncertainty Estimation in Generative LLMs. *ACL 2024*.

[8] Zhang, D., et al. (2024). Monte Carlo Tree Search Self-Refine (MCTSr) for LLMs. arXiv:2406.07394.