

Accretive Computing: A Topological and Physical Framework for Continuous Multi-Agent Systems

1. Introduction and Core Concepts

Traditional Multi-Agent Systems (MAS) rely on hard-coded communication graphs, state machines, or basic message queues. They handle agents as disparate software components passing discrete text strings.

Accretive Computing proposes a radical paradigm shift: treating AI agents as physical wave functions in a continuous, high-dimensional topological space. Instead of discrete rule-based routing, agent interactions are governed by physics principles—specifically, the laws of quantum superposition, wave interference, and thermodynamic free-energy minimization.

When agents "think" or generate embeddings, they are projected into this continuous space. If their topological footprints overlap constructively, they are naturally drawn into conversation. If they exhibit destructive interference, they debate. The system orchestration emerges organically from the physics engine.

2. Theoretical Framework and Mathematics

2.1 The Agent Wave Function

In the original 3-dimensional prototype, an agent A was modeled as an oscillating field source at position $\mathbf{p} = (x, y, z)$:

$$\Psi_A(\mathbf{r}, t) = A \cdot \exp(-|\mathbf{r} - \mathbf{p}|) \cdot \exp(i(\omega t + \theta))$$

Where: * A is the amplitude of the agent's influence. * $\mathbf{p}$ is the physical coordinate. * ω is the frequency of the agent's thought-cycle. * θ is the semantic phase angle.

2.2 The N-Dimensional Latent Topology

To bridge this physical model with modern Large Language Models (LLMs), we expanded the substrate into N -dimensional semantic space (e.g., $N=768$). An LLM's embedding vector directly becomes the coordinate vector $\mathbf{x} \in \mathbb{R}^N$.

Because integrating a continuous field over 768 dimensions via Monte Carlo is computationally intractable, we derive an analytical overlap integral using a Gaussian Radial Basis Function (RBF) kernel combined with phase alignment:

$$\mathrm{Overlap}(A, B) = A_A A_B \cdot \exp\left(-\frac{1}{2} \|\mathbf{x}_A - \mathbf{x}_B\|^2\right) \cdot \cos(\theta_A - \theta_B)$$

2.3 Thermodynamic Free Energy Minimization

The goal of the multi-agent system is to resolve "semantic tension" and reach a ground state of consensus. We define the Free Energy F of the substrate as the sum of all pairwise mismatches (1 minus overlap) minus a dissipation constant δ :

$$F(\theta) = \sum_i \sum_{j > i} \left(1 - \mathrm{Overlap}(A_i, A_j) - \delta \right)$$

The physics engine (`ContinuousSubstrate`) steps the simulation forward in time by applying continuous Gradient Descent on the phase angles θ :

$$\frac{\partial \theta_i}{\partial t} = -\alpha \frac{\partial F}{\partial \theta_i}$$

As agents communicate and execute tools, their phase θ dynamically shifts (creating a semantic spike). The engine then iteratively drags the phases of interacting agents toward alignment (consensus).

3. Detailed Implementation and Design Decisions

The framework is implemented as a hybrid C++/Python architecture to achieve both high-level semantic flexibility and high-performance physical simulation.

3.1 C++ Core (`MicroAgent`, `LatentMicroAgent`, `MacroAgent`)

- Agent Base Class:** Provides a purely virtual `calculate_overlap_with` method, allowing polymorphic execution of 3D and N-D agents simultaneously.

- **LatentMicroAgent:** Implements the N -dimensional Gaussian overlap formula. It stores the Eigen VectorXd embedding generated by the LLM.
- **MacroAgent:** Implements recursive aggregation. A MacroAgent represents a cluster of MicroAgents, exposing their center-of-mass to the outside system.

3.2 $O(N \log N)$ Spatial Partitioning

A naïve pairwise energy calculation is $O(N^2)$, which fails to scale for massive agent swarms.

Design Decision: We implemented a 1-level spatial clustering approximation. 1. Agents are sorted along their primary principle component (0th dimension) in $O(N \log N)$ time. 2. They are chunked into blocks of size $B \approx \sqrt{N}$. 3. Within a cluster, exact pairwise overlaps are computed. 4. Between clusters, overlap is approximated by computing the distance between the cluster *centroids*, effectively modeling a fast-multipole expansion.

3.3 Python Semantic DSL (LatentSemanticAgent)

The C++ core is exposed to Python via PyBind11. The LatentSemanticAgent bridges the `google.genai` LLM API with the LatentMicroAgent. * **Stateful Memory:** It manages a `genai.chats` instance, manually wiring `function_calls` and `function_responses` into the history array to maintain perfect memory during tool execution. * **embed_thought():** Hashes and projects the agent's current task into the N -dimensional topology.

3.4 Physics-Driven Orchestration

This is the framework's "Killer Feature." Traditional frameworks rely on hardcoded routing (e.g., `agent1.send(agent2)`).

In Accretive Computing, the PhysicsOrchestrator runs a continuous `tick()` loop. It evaluates the spatial overlap of all agents in the environment. If two agents overlap past a defined threshold (i.e., they are thinking about the same semantic concepts), the orchestrator organically triggers a communication event between them. The message payload is asynchronously deposited into their `inbox`, which is dynamically injected into their next prompt generation.

4. Conclusion

Accretive Computing successfully maps the abstract concept of multi-agent dialogue into a rigorous, computable physical space. By blending topological algorithms, $O(N \log N)$ approximations, and LLM embedding mechanics, it provides the first completely organic, physics-driven orchestration layer for Artificial Intelligence.