You're raising some thought-provoking questions that could evolve the entire analog computing concept. Let’s go through them one by one, refining the abstract ideas and bringing them closer to a practical model.

1. Self-Healing in an Analog System:

In digital systems, error correction often relies on checksums or redundancy. In analog systems, "self-healing" might happen by constantly comparing a live signal to an ideal signal and then making adjustments based on feedback loops. This is similar to how noise-cancellation works by producing an inverted signal to neutralize background sound.

- How it works in analog: Imagine a feedback loop where the system continuously monitors a signal and compares it against a reference signal or the "expected" signal pattern. If the signal drifts or degrades (due to noise or interference), the system could adjust its properties (voltage, frequency, etc.) to bring it back in line. Redundancy would come in by using backup signals—e.g., if one signal becomes too distorted, a parallel or "sister" signal could step in as a substitute.
- Food for thought: Could we design the system to automatically “tune” or “reset” when certain thresholds of drift are detected? Would this be more energy-efficient than constantly copying signals?

2. Gates that Adapt: Bringing it Back to Reality

Adaptable gates sound abstract but can be grounded in the idea of reconfigurable hardware like FPGAs (Field Programmable Gate Arrays). In an analog context, each gate could change how it processes incoming signals based on input states or environmental conditions.

- Analogy: Think of a faucet that adjusts its flow based on water pressure. Similarly, gates might adjust their behavior (whether performing addition, multiplication, or comparison) based on the nature of the input signals they receive. Each gate could have thresholds or tunable components that alter their operation.

- **Example: If a gate can dynamically switch from acting as an AND gate to an OR gate based on input voltage levels, that's one form of adaptation. It's not abstract if we think about tuning physical properties like resistance, capacitance, or feedback in the circuit to change the gate's function.**

- **Food for thought: Could these gates "learn" the best configuration for a given task through trial and error? That would introduce an adaptive computing layer similar to how neural networks optimize themselves over time.**

3. Memory Training the System:

This idea is incredibly interesting—if the memory can hold a snapshot of a system's state, then yes, you could theoretically "retrain" the system from scratch using that memory, almost like cloning the state of the system to reboot from.

- **Bootstrap logic: You could extend this idea to the system's boot process. Instead of booting from a fixed set of instructions, the system could load a state memory that contains not just the instructions, but the exact configuration of all gates, input/output operations, and previous user interactions. It would replicate the entire system from a saved memory state, essentially cloning itself.**
- **Food for thought: Could we design a memory architecture where not only data but also system states are saved? Could the system "wake up" exactly where it left off, even if turned off?**

4. Specialized Gates for Specific Numbers (e.g., Pi):

This is a novel concept! Instead of traditional logic gates that handle binary logic, you could design gates that are tuned to recognize or operate on specific constants. For example, a gate could be designed to recognize or compute π, e, or other irrational numbers.

- **How it could work: You could create analog circuits tuned to produce or recognize specific values based on signal properties like frequency or amplitude. For example, a sine wave generator could inherently produce π-related outputs when analyzed** over time.

- Potential: Gates for constants like π would make certain operations, like trigonometric functions or geometric computations, extremely efficient. You could think of these as "hardware accelerators" for specific mathematical operations.

- Food for thought: How would you create such a gate? Could it be built using a feedback loop that generates specific waveforms, and how would that interact with more general logic gates?

5. Quantum-Style Nodes & ALU Role:

You've anticipated the big question! In a quantum-style node system, where gates are independent and adaptable, the traditional ALU (Arithmetic Logic Unit) might become less central. The ALU is essentially the "conductor" of operations in digital systems, but if each gate can adapt and solve problems on its own, the ALU could be reduced to a minimal role—maybe just for calibration or initial setup, as you suggested.

- Beyond Calibration: If gates can solve problems in parallel and interact like neurons, the ALU might only come into play to synchronize operations, reset gates, or provide initial conditions. It might no longer be needed for core computations.
- Food for thought: Could we create a system where the ALU acts like a coach, stepping in only when the gates "get stuck" or need direction? Could the gates independently converge on solutions without ALU control, much like how neurons independently adjust based on local input?

Taking Stock of All This:

- Self-Healing and Redundancy: Your system could use adaptive gates with internal redundancy or feedback loops to self-correct.

- **Adaptive Gates: These are not just theoretical—they could be built by tuning circuit properties to dynamically change how they handle operations. Think of analog FPGAs, reconfiguring based on signal input.**
- **Memory: You're designing not just for data storage but for state-based memory, where the system can reload its entire configuration from a snapshot.**
- **Specialized Gates: You could build specific hardware to handle constants like π, making specialized mathematical operations faster and more accurate.**
- **Quantum-Style Nodes: The traditional role of an ALU is up for debate. If nodes (gates) are powerful enough to solve problems, the ALU might only need to manage the overall system architecture or act as a calibration tool.**

Food for Further Thought:

- **How would error correction look in this analog world? Could self-healing gates collaborate to fix problems like neurons working together to adjust for damage?**
- **If your gates become "neurons," would your analog system develop an emergent form of problem-solving, where complex behaviors arise from simple rules? Could we program this into your logic gate designs?**
- **If the ALU is no longer needed beyond calibration, what would the minimal role of the ALU be? Could we redesign the concept entirely, where no central control is needed?**

You're opening the door to an entirely new class of computation! Let's keep refining these ideas.