Quantum Multiverse Navigation: Testable Framework with Empirical Validation

Author: Gust Isotalo and Brent Antonson
Date: January 14, 2026

Abstract

This document presents a streamlined, testable architecture for stabilizing complex dynamical systems and augments it with evidence from the literature on high-dimensional control. We define a system state, coherence metric, divergence metric and a control algorithm (Da'at Command) that triggers interventions when the system drifts from a target regime. A notation table and algorithms are provided. Recent case studies in reduced-order model predictive control and fast stochastic model predictive control demonstrate that high-dimensional models—on the order of ten thousand to one million states—can be stabilized with reduced computational burden 【943755515450370†L97-L108】【250286303876478†L60-L69】. We outline simulation protocols for evaluating the approach, discuss empirical validation strategies, and address computational cost when scaling to high-dimensional systems.

1. Overview

The original proposal aimed to integrate concepts from quantum physics, dynamical systems and control theory. This version reframes it as a practical research program. The core question is whether an adaptive control mechanism informed by coherence and information divergence can stabilize chaotic or sensitive systems. We define all variables and thresholds precisely and describe an algorithmic intervention strategy. In addition, we consider the feasibility of applying the algorithm to very high-dimensional models by examining existing empirical evidence from the literature.

2. Scope and Non‑Claims

Our framework is a control algorithm for mathematical or computational models of dynamical systems. It does not claim to describe physical quantum gravity, consciousness or metaphysical realities. The term "multiverse" refers to a high-dimensional state space. Labels from Kabbalah are used as neutral shorthand. The focus is on:

- Defining system variables, update rules and observations.
- Measuring coherence in a system via a chosen scalar metric.
- Quantifying divergence between observed behavior and a template distribution using KL divergence.
- Triggering a controlled intervention based on these measurements.
- Evaluating the algorithm against standard control methods.
- Considering empirical evidence from high-dimensional control to assess feasibility and

computational cost.

3. Symbols and Definitions

Symbol	Meaning / Type	Notes
(x_t)	State vector at time (t)	Elements of (^d) or similar; governed by a dynamics rule.
(t), (t)	Time index and step size	Applicable to discrete or continuous time models.
()	System representation	Configuration on a lattice, field, or graph.
((x_t))	Coherence metric	Scalar in ([0,1]); larger means more order.
(_c)	Coherence threshold	Collapse triggered if ((x_t) < _c).
()	KL divergence threshold	Collapse triggered if divergence exceeds ().
(P_{})	Observed distribution	Derived from recent states via a sliding window.
(P_{})	Template distribution	Represents desired system behavior.
(D_{})	Kullback–Leibler divergence	Measures how far (P_{}) deviates from (P_{}).
(u_t), ()	Control input or policy	Input applied when a collapse is triggered.
(W)	Window length	Number of steps used to estimate (P_{}).

4. Coherence Metric

Several choices for () exist:

- **Phase coherence**: (= |_{k=1}^N e^{i_k}|), where (_k) is the phase of component (k).
- **Normalized entropy**: (= 1 -), where (H(G_t)) is the entropy of a graph (G_t).
- **Distance to attractor**: (= 1/(1 + |x_t - x_|)*), where (x_)* is a target state.

Any metric that quantifies order can be used, provided it maps to ([0,1]) and decreases as the system becomes disordered. When applying to high-dimensional systems, care should be taken to select metrics that are computationally tractable (e.g., approximate entropy measures or coherence measures that can be estimated from reduced representations).

5. Distribution Estimation

Define (z_t = f(x_t)). Collect (W) consecutive samples ({z_{t-W+1}, , z_t}) and estimate (P_{}) using histograms, kernel density estimation or parametric models. The template distribution (P_{}) may come from stable baseline runs or analytical definitions. Compute (D_{}(P_{} P_{})) using the standard formula. For high-dimensional systems, low-dimensional observations (z_t) may be obtained by projecting (x_t) onto a set of principal modes (e.g. via proper orthogonal decomposition) or by using an input–output representation similar to dynamic matrix control; this can reduce the dimension of the distribution estimation problem and keep the computational cost manageable.

6. Da’at Command Algorithm

```
Inputs:
  - States {x_t}
  - Window W, thresholds Φ_c and ε
```

```
  - Coherence metric Φ, estimators for P_obs and P_temp
  - Control operator T̂ or policy π

Initialize c_t = 0 for all t
For each t ≥ W:
  1. z_t ← f(x_t)
  2. Compute Φ(t) = Φ(x_t)
  3. Estimate P_obs from {z_{t-W+1}, …, z_t}
  4. Compute KL(t) = D_KL(P_obs || P_temp)
  5. If Φ(t) < Φ_c and KL(t) > ε:
         c_t ← 1
         Apply T̂(x_t) or set u_t = π(x_t)
     Else:
         c_t ← 0
Output: decisions {c_t}, updated states {x_t}
```

This algorithm is system-agnostic and can be applied to a variety of discrete or continuous dynamical models. Thresholds can be fixed or tuned using baseline data. When implementing on high-dimensional systems, the costs associated with updating (z_t), estimating distributions and computing () may be dominated by dimension; thus dimensionality reduction or input–output formulations are advised.

7. Simulation Plan and Falsifiability

1. **Test systems**: Choose examples ranging from low-dimensional chaos (e.g. logistic map, Henon map) to high-dimensional models (e.g. coupled oscillators with hundreds of states, discretized fluid dynamics). For the latter, a reduced-order approximation or input–output representation should be used to manage computational cost.
2. **Metrics**: Use Lyapunov exponents, stability times and basin measures to quantify improvement. Track the coherence metric over time and the frequency of interventions.
3. **Baselines**: Implement controllers for comparison: no control, PID, LQR, reinforcement learning and standard methods like dynamic matrix control. Include reduced-order model predictive control (ROMPC) and fast stochastic model predictive control (FSMPC) as context.
4. **Algorithm evaluation**: Run the Da'at Command algorithm on each system, record stability outcomes, and compare to baselines. Use ablation to remove the coherence trigger or divergence trigger individually.
5. **Hypothesis testing**: Use statistical tests to determine whether the algorithm improves stability relative to baselines. If no significant improvement is observed across multiple systems and parameters, reject the hypothesis.

8. Empirical Validation from High-Dimensional Control

The feasibility of stabilizing high-dimensional dynamical systems has been demonstrated in recent control research:

- **Reduced-Order Model Predictive Control (ROMPC)**. Lorenzetti's dissertation developed a model predictive control framework that uses reduced-order approximations to control systems with extremely large state spaces. Two case studies validated the

approach: (i) a linear coupled rigid-body/fluid dynamics model for aircraft control with a computational fluid dynamics (CFD) model exceeding one million dimensions; (ii) a nonlinear finite element model (FEM) with over ten thousand dimensions for soft robot control. Simulation and hardware experiments showed that ROMPC can achieve practical performance on these high-dimensional systems【943755515450370†L97-L108】.

- **Fast Stochastic Model Predictive Control (FSMPC)**. Von Andrian and Braatz extended stochastic MPC formulations to handle high state dimensions and uncertain dynamics. In a realistic manufacturing system with roughly 8,000 states, the FSMPC algorithm had an online computational cost of under one second per sampling instance due to its input–output model formulation【250286303876478†L60-L69】. This cost was far below the system's sampling period (one minute), demonstrating that high-dimensional state space does not necessarily lead to prohibitive computational load when an appropriate representation is used.

These results suggest that our framework could be validated on systems of comparable scale using reduced representations. The ROMPC case studies indicate that control algorithms informed by reduced models can be experimentally implemented on physical systems with millions of states. The FSMPC example shows that computational cost can remain tractable even for thousands of states when using an input–output model. For the Da'at Command algorithm, similar strategies—projecting onto dominant modes and estimating distributions over a reduced window—would be essential for empirical validation at high dimensionality.

9. Computational Cost Considerations

The computational burden of the Da'at Command algorithm depends on the cost of computing the coherence metric, estimating distributions and solving the control update. For high-dimensional systems:

- **Reduced-order models**: Use reduced basis or principal component analysis to project the system state onto a lower dimension before computing () and (P_{}). This mirrors the ROMPC approach and can reduce the complexity from millions of states to tens or hundreds while preserving essential dynamics【943755515450370†L97-L108】.
- **Input–output formulation**: When only a few outputs and inputs are relevant, adopt an input–output model such as dynamic matrix control. The FSMPC algorithm demonstrates that this yields an online computational cost largely independent of the full state dimension【250286303876478†L60-L69】. Implementation details include precomputing step response coefficients and updating control signals based solely on observed outputs.
- **Sampling and window size**: Choose the window length (W) and sampling interval such that the distribution estimation step can be completed within the available computation time. For high-dimensional states, short windows or incremental estimators may be necessary.
- **Parallelism and hardware acceleration**: Distribution estimation and coherence calculation can be parallelized across processing cores or GPUs. Real-time requirements should be matched against available computational resources.

Ultimately, the algorithm’s feasibility in high-dimensional settings will depend on balancing accuracy and computational cost. Empirical testing on reduced models or input–output representations should be conducted to measure run-time per step, compare against the sampling period and confirm that interventions are computed within the required latency.

10. Related Work

This approach builds upon chaos control techniques (e.g., the Ott–Grebogi–Yorke method), information-theoretic control, quantum cellular automata, and adaptive feedback. It is also influenced by recent work in reinforcement learning and algorithmic complexity measures. Reduced-order model predictive control and stochastic MPC provide examples of successfully controlling high-dimensional systems【943755515450370†L97-L108】【250286303876478†L60-L69】.

11. Limitations and Future Work

Current limitations include sensitivity to parameter choices, the need for systematic benchmarks and the open question of how best to integrate dimensionality reduction with the algorithm. Future work should:

- Implement the Da’at Command algorithm on high-dimensional testbeds using reduced models or input–output representations.
- Compare computational cost per time step to the system’s sampling period, drawing on benchmarks from ROMPC and FSMPC literature【943755515450370†L97-L108】【250286303876478†L60-L69】.
- Explore adaptive threshold selection for (_c) and () to ensure robust performance across different systems.
- Investigate synergy with learning methods (e.g., reinforcement learning) for more sophisticated control policies.

This version integrates a concrete empirical validation perspective, citing existing high-dimensional control studies and explaining how similar strategies can be used to test the proposed framework. It addresses computational scalability and outlines next steps for applying the algorithm to real systems.