AcaysiaRT & AcaysiaDRT: GPU-Native Reactor Simulation for Large Ensemble and Spatial Reactive Transport Studies

1 Summary

AcaysiaRT and AcaysiaDRT form a two-library suite for GPU-native reactor simulation. AcaysiaRT provides vectorized ensemble stepping and rollout for batch reactor dynamics, enabling millions of parallel reactor evaluations on a single GPU. AcaysiaDRT extends this to 3D spatial domains by coupling the same chemistry model with D3Q19 Lattice Boltzmann transport via Strang splitting. Both libraries target process systems engineering (PSE) applications requiring massive parallelism: Model Predictive Control (MPC) with sampling-based optimization, Monte Carlo parameter studies, and machine learning training data generation. While the current reference implementation focuses on aqueous carbonate chemistry, the underlying engine is chemistry-agnostic and designed to support arbitrary reaction kinetics and state definitions that can be expressed as an explicit time-stepping operator.

2 Motivation

Sampling-based optimization methods—including Model Predictive Path Integral (MPPI), Cross-Entropy Method (CEM), and gradient-free trajectory optimization—require evaluating thousands to millions of candidate control sequences in parallel. Traditional reactor simulation tools serialize these evaluations, creating a computational bottleneck that limits real-time applicability. Similarly, uncertainty quantification via Monte Carlo requires large sample counts that are impractical with sequential evaluation.

AcaysiaRT addresses this by implementing the complete reactor dynamics—mass balances, energy balance, and pH equilibrium chemistry—as GPU-vectorized tensor operations. The CUDA backend achieves approximately 2 billion reactor-steps per second, enabling 10,000 parallel rollouts of 100 steps each in under 1 millisecond. This throughput is sufficient for real-time MPC at control frequencies of 1–10 Hz while maintaining exact carbonate chemistry without surrogate approximations.

3 AcaysiaRT: Ensemble Reactor Engine

3.1 Interface

AcaysiaRT exposes a minimal, NumPy/PyTorch-compatible interface:

```
import acaysia_rt as art
```

```
# Single step: [N, 8] -> [N, 8]
x_next = art.step(x, u, dt)

# Multi-step rollout: [N, H, 8]
trajectory = art.rollout(x, U, dt)

# Observable extraction: pH, species concentrations
observables = art.observe(x)
```

All operations are batch-vectorized over the first dimension N, representing independent reactor instances. The engine does not assume a fixed state dimension or reaction network; both are defined by the user-provided dynamics kernel.

3.2 State Representation

The state vector contains 8 conserved quantities sufficient to reconstruct all species concentrations:

Index	Field	Units	Description
0	TEMP	K	Temperature
1	VOLUME	L	Reactor volume
2	DIC	mol/L	Dissolved inorganic carbon
3	ALK	eq/L	Total alkalinity
4	WAC_TOT	mol/L	Weak acid total concentration
5	WBASE_TOT	mol/L	Weak base total concentration
6	BIAS_PH	—	pH sensor bias term
7	UA_SCALE	—	Heat transfer coefficient scale

Table 1: State vector components

The choice of conserved totals (DIC, ALK) rather than individual species concentrations ensures numerical stability and mass conservation across the pH equilibrium solve.

3.3 Chemistry Model

The equilibrium chemistry is solved via a log-domain Newton iteration that computes pH from conserved totals. The carbonate system includes:

$$\mathrm{H_2CO_3} \rightleftharpoons \mathrm{H^+} + \mathrm{HCO_3^-} \qquad K_1(T) \tag{1}$$

$$\mathrm{HCO_3^-} \rightleftharpoons \mathrm{H^+} + \mathrm{CO_3^{2-}} \qquad K_2(T) \tag{2}$$

$$\mathrm{H_2O} \rightleftharpoons \mathrm{H^+} + \mathrm{OH^-} \qquad K_w(T) \tag{3}$$

Temperature-dependent equilibrium constants follow Plummer & Busenberg correlations. The solver enforces mass conservation (DIC) and charge balance (ALK) at each timestep.

Generality of the Chemistry Interface. The carbonate system is used here as a representative, stiff, industrially relevant chemistry for validation and benchmarking. However, AcaysiaRT does not assume a fixed reaction network. The engine operates on user-defined state vectors and dynamics kernels, and alternative chemistries (e.g., kinetic reaction networks, bioprocess models, crystallization kinetics) can be integrated by providing a compatible time-stepping operator.

3.4 Control Inputs

Input	Units	Description
BASE_RATE	mol/s	Base addition rate
ACID_RATE	mol/s	Acid addition rate
FEED_FLOW	L/s	Feed stream flow rate
TAU_SCALE	—	Residence time multiplier
TEMP_SP	K	Temperature setpoint

Table 2: Control vector components

3.5 Performance

The CUDA backend achieves the following throughput on an NVIDIA RTX-class GPU:

Metric	Value
Batch size	10,000 reactors
Horizon	100 steps
Rollout time	0.52 ms
Throughput	$\sim 2 \times 10^9$ reactor-steps/s
Speedup vs. PyTorch	130×

Table 3: AcaysiaRT CUDA backend performance

The CUDA backend is operational but should be considered experimental. The PyTorch backend provides identical numerical results with automatic differentiation support.

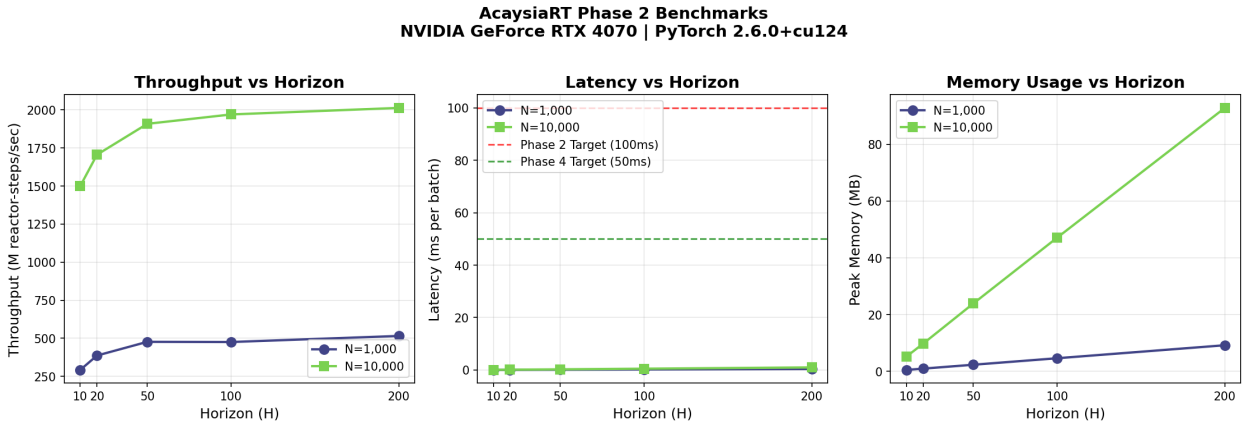

Figure 1: AcaysiaRT CUDA backend performance on NVIDIA RTX 4070. Left: throughput reaches $\sim$2 billion reactor-steps/sec at $N = 10{,}000$. Center: latency remains below 1 ms, well under real-time MPC targets. Right: memory scales linearly with horizon.

4 AcaysiaDRT: Spatial Reactive Transport

4.1 Architecture

AcaysiaDRT couples the AcaysiaRT chemistry model with 3D advection-diffusion transport using the Lattice Boltzmann Method (LBM). Each grid cell contains a full 8-component state vector, and transport is computed via D3Q19 passive scalar advection.

The time-stepping scheme uses Strang splitting:

1. Chemistry half-step: dt/2
2. LBM transport full-step: dt
3. Chemistry half-step: dt/2

This second-order splitting preserves conservation while allowing independent optimization of chemistry and transport kernels.

4.2 Data Layout

States are stored in Structure-of-Arrays (SoA) format for coalesced GPU memory access:

- Cell indexing: cell_id $= x + N_x(y + N_y z)$
- State tensor: $[8, N_{\text{cells}}]$
- LBM distributions: $[19, K, N_{\text{cells}}]$ where $K = 4$ transported fields

The transported fields are DIC, ALK, WAC_TOT, and WBASE_TOT. Temperature transport is optional.

4.3 Conservation Guarantees

AcaysiaDRT maintains strict mass conservation through:

- Per-timestep mass audit: Total mass is computed before and after each step
- Positivity clamping: Concentrations are clamped to ≥ 0 with mass injection tracked
- Volume freeze: Cell volumes remain constant (no CSTR flow dynamics in spatial mode)

Conservation residuals are reported in the step statistics for debugging.

4.4 Performance

Current performance with the PyTorch backend (single GPU):

A dedicated CUDA kernel implementation is expected to provide 10–100$\times$ speedup over the current PyTorch backend.

Grid	Cells	ms/step	MLUPS
16^3	4K	20.3	0.20
32^3	32K	20.8	1.58
64^3	262K	24.0	10.9
128^3	2M	60.7	34.6

Table 4: AcaysiaDRT throughput (MLUPS = Million Lattice Updates Per Second)

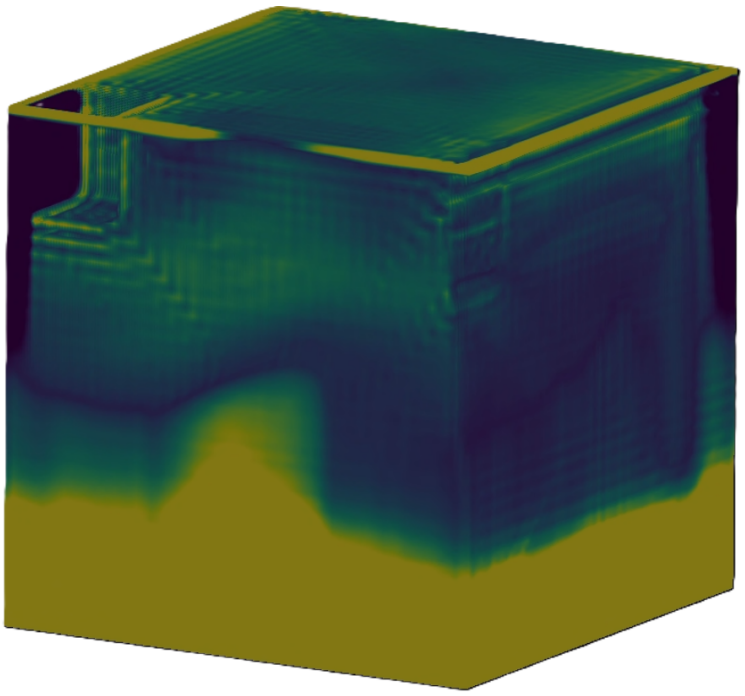

Figure 2: Example scalar concentration field from AcaysiaDRT showing conservative reactive transport on a uniform 3D grid using LBM-based advection and local chemistry evaluation via AcaysiaRT.

5 Use Cases

Model Predictive Control. Shooting MPC and MPPI require evaluating thousands of candidate control sequences per control step. AcaysiaRT's sub-millisecond rollout enables real-time MPC at 1–10 Hz control frequencies with 10K+ parallel samples.

Monte Carlo Studies. Uncertainty quantification via Monte Carlo sampling requires 10^4–10^6 independent simulations. GPU-vectorized stepping makes large-scale sensitivity analysis and parameter uncertainty propagation practical.

Machine Learning. Training neural network surrogate models or reinforcement learning policies requires millions of simulation trajectories. AcaysiaRT provides PyTorch-native tensors with automatic differentiation for end-to-end gradient computation.

Spatial Optimization. AcaysiaDRT enables optimization of injection strategies, reactor geometry effects, and mixing patterns in 3D reactive transport systems.

6 Scope and Limitations

AcaysiaRT and AcaysiaDRT are purpose-built for:

- Aqueous carbonate chemistry (pH control, alkalinity management)
- Large-batch parallel evaluation (1K–1M simultaneous reactors/cells)
- Integration with PyTorch-based optimization and ML workflows

The libraries explicitly do not target:

- General-purpose CFD (no Navier-Stokes, no turbulence modeling)
- Multiphite flow (single aqueous phase only)
- Precipitation/dissolution kinetics (equilibrium chemistry only)
- Non-carbonate reaction networks are not yet provided as reference implementations (though the framework is extensible)

7 Integration

AcaysiaRT (ensemble reactors):

```
import torch
import acaysia_rt as art

# Initialize 10,000 reactors
x = torch.zeros(10000, 8, device="cuda")
x[:, 0] = 298.0   # TEMP = 298 K
x[:, 1] = 100.0   # VOLUME = 100 L
x[:, 2] = 0.002   # DIC = 2 mM
x[:, 3] = 0.0023  # ALK = 2.3 meq/L

# Control sequence: [N, H, 5]
U = torch.zeros(10000, 100, 5, device="cuda")
U[:, :, 0] = 1e-4  # BASE_RATE

# Rollout
trajectory = art.rollout(x, U, dt=0.1)  # [N, H, 8]
```

AcaysiaDRT (spatial transport):

```
from acaysia_drt import DRTParams, backend_torch as drt
from acaysia_drt.models.spatial_cstr import create_uniform_states

params = DRTParams(Nx=64, Ny=64, Nz=64, V_cell_L=1e-6, dx_m=0.001)
states = create_uniform_states(params, "cuda", dic=0.002, alk=0.002)

# Simulation loop
for _ in range(100):
    states, f, stats = drt.step_soa(
        states, controls, velocity, f, bc, dt=0.001,
        params=params, device="cuda"
    )
```