HGD Method Comparison: Benchmarking Against Other Numerical Methods

Comparison of Heterarchical Granular Dynamics with Alternative Approaches

November 11, 2025

Abstract

This document compares the Heterarchical Granular Dynamics (HGD) method with other numerical approaches for modeling granular flow, including Lattice Gas Automata (LGA), Lattice Boltzmann Method (LBM), and other stochastic advection methods. We analyze accuracy, robustness, computational efficiency, and suitability for different physical regimes. Based on this comparison, we provide recommendations for optimal implementation strategies and identify areas for improvement.

Contents

1 Introduction

The HGD method is a particle-based stochastic approach for modeling granular flow through void migration. To understand its strengths and limitations, we compare it with established numerical methods:

- **Lattice Gas Automata (LGA)**: Discrete particle automata on a lattice
- **Lattice Boltzmann Method (LBM)**: Mesoscopic fluid simulation
- **Discrete Element Method (DEM)**: Individual particle tracking
- **Monte Carlo Methods**: Stochastic sampling approaches
- **Continuum Methods**: PDE-based approaches (Navier-Stokes, granular rheology)

2 Method Classifications

2.1 Spatial Discretization

Method	Grid Type	Particle	Continuous	Hybrid
HGD	Regular lattice	Yes	No	Yes
LGA	Regular lattice	Yes	No	No
LBM	Regular lattice	No	Yes	No
DEM	Off-grid	Yes	No	No
Continuum	Grid/mesh	No	Yes	No

Table 1: Spatial discretization characteristics

2.2 Temporal Evolution

Method	Deterministic	Stochastic	Time Scale
HGD	Partial	Yes	Explicit
LGA	Yes	No	Explicit
LBM	Yes	No	Explicit
DEM	Yes	Optional	Explicit
Monte Carlo	No	Yes	Event-driven

Table 2: Temporal evolution characteristics

3 Detailed Method Comparison

3.1 Lattice Gas Automata (LGA)

3.1.1 Description

LGA represents fluids as collections of discrete particles on a lattice, moving according to simple collision rules. The HPP and FHP models are classic examples.

3.1.2 Similarities to HGD

- Both use discrete particles on a regular lattice
- Both use local collision/swap rules
- Both achieve macroscopic behavior from microscopic rules
- Both are computationally efficient compared to molecular dynamics

3.1.3 Differences from HGD

- **LGA**: Particles are identical, boolean occupation
- **HGD**: Particles have sizes, multiple per cell (heterarchical coordinate)
- **LGA**: Designed for fluid flow (incompressible Navier-Stokes)
- **HGD**: Designed for granular flow with size segregation
- **LGA**: Deterministic collision rules
- **HGD**: Probabilistic swaps with gravity bias

3.1.4 Accuracy Comparison

LGA limitations:

- Statistical noise requires large ensembles
- Lacks Galilean invariance (pre-LBM)
- Limited to specific lattice symmetries

HGD advantages:

- Can represent polydisperse systems
- Direct physical interpretation of swap probabilities
- Naturally handles segregation

3.2 Lattice Boltzmann Method (LBM)

3.2.1 Description

LBM solves the discrete Boltzmann equation for distribution functions on a lattice. It recovers Navier-Stokes equations through Chapman-Enskog expansion.

3.2.2 Comparison with HGD

LBM advantages:

- Well-established theoretical foundation (Chapman-Enskog)
- Smooth, continuous fields
- Excellent for single-phase incompressible flow
- Lower statistical noise

Feature	LBM	HGD
Variables	Distribution functions	Particle sizes + voids
Evolution	BGK collision	Probabilistic swaps
Macroscopic limit	Navier-Stokes	Granular flow
Noise	Minimal	Inherent (stochastic)
Multiphase	Complex	Natural (heterarchical)
Parallelization	Excellent	Good
Memory	Moderate	High (heterarchical dim)

Table 3: LBM vs HGD comparison

- More accurate for viscous flows

HGD advantages:

- Directly represents particle-level physics
- Natural handling of size segregation
- No need for multiphase flow models
- Better for dense granular flows
- Simpler conceptual model

3.3 Discrete Element Method (DEM)

3.3.1 Description

DEM tracks individual particles with explicit contact mechanics, solving Newton's equations for each particle.

3.3.2 Comparison

DEM advantages:

- Most accurate for particle-level physics
- Captures contact forces, friction, rotation
- No grid restrictions
- Well-validated for many applications

DEM disadvantages:

- Computationally expensive ($O(N^2)$ or $O(N \log N)$ with neighbor lists)
- Limited to $\sim 10^6$ particles on typical hardware
- Requires small timesteps (contact duration)
- Complex parameter calibration

HGD advantages over DEM:

- Much faster (can handle 10^9+ particles)

- Larger timesteps possible
- Simpler parameterization
- Good for large-scale systems

HGD disadvantages vs DEM:

- Less accurate force resolution
- No explicit contact mechanics
- Limited to quasi-static/slow flows
- Grid-based restrictions

3.4 Monte Carlo Methods for Advection

3.4.1 Random Walk Methods

Random walk methods move particles stochastically with probabilities based on local conditions.

Similarity to HGD: HGD is essentially a random walk method for voids with gravity bias.

Differences:

- Standard random walk: No preferred direction
- HGD: Gravity-biased with upward preference
- HGD: Includes lateral diffusion proportional to velocity

3.4.2 Kinetic Monte Carlo (KMC)

KMC advances time based on rates of discrete events.

Comparison with HGD:

- **KMC**: Event-driven, variable timestep
- **HGD**: Fixed timestep, multiple events per step
- **KMC**: Exact for Markov processes
- **HGD**: Approximate, collision handling needed

3.5 Continuum Methods

3.5.1 Granular Rheology Models

$\mu(I)$ rheology, critical state theory, etc.

Continuum advantages:

- Fast for large systems
- Well-established for uniform flows
- Good for engineering applications

HGD advantages:

- Captures segregation naturally
- No constitutive model needed
- Better for heterogeneous systems
- Particle-level detail

4 Robustness and Accuracy Concerns

4.1 Current HGD Limitations

Based on the analysis of existing implementations, several robustness/accuracy issues exist:

4.1.1 1. Probabilistic Inconsistencies

Issue: Multiple swaps can target the same destination cell, causing conflicts.
Current handling:

- `d2q4_slow.py`: Random iteration order
- `d2q4_array_v2.py`: Explicit conflict detection and resolution
- `core.cpp`: Random shuffle before processing

Problem: Resolution is not physically consistent - which swap "wins" is arbitrary.
Solution approaches:

1. **Kinetic Monte Carlo**: Select one event based on relative rates
2. **Fractional occupancy**: Allow partial swaps (more LBM-like)
3. **Sequential updates**: Process cells in deterministic order with availability checking

4.1.2 2. Timestep Stability

Issue: Large timesteps can lead to $P_{tot} > 1$, which is unphysical.
Current handling: Check and error/warning when $P_{tot} > 1$
Problem: No automatic timestep adaptation
Solutions:

1. **Renormalization**: Scale all probabilities by P_{tot} if $P_{tot} > 1$
2. **Adaptive timestep**: Reduce Δt automatically
3. **Implicit methods**: Allow larger timesteps

4.1.3 3. Boundary Condition Consistency

Issue: Periodic boundaries with offset can create artifacts.
Current handling: Manual offset specification in parameters
Problems:

- Non-physical flow near boundaries
- Difficult to validate
- Limited boundary condition types

4.1.4 4. Inertia Implementation

Issue: Incomplete/inconsistent inertia in different implementations (see motion_models_analysis.pdf)
Impact on robustness:

- Velocities can grow unbounded without proper damping
- Momentum not conserved in all implementations
- Collision handling is ad-hoc

4.1.5 5. Grid Resolution Effects

Issue: Lattice artifacts, lack of convergence studies
Problems:

- No systematic grid refinement studies
- Particle size relative to cell size matters
- Heterarchical coordinate resolution (n_m) affects statistics

4.2 Accuracy Metrics

To properly benchmark HGD, we need to define accuracy metrics:

4.2.1 Conservation Properties

1. **Mass conservation**: $\sum_{i,j,k} \text{particle}(i,j,k) = \text{const}$
2. **Momentum conservation** (with inertia): $\sum_{i,j,k} mv = \text{const}$
3. **Energy considerations**: Dissipation should match physics

4.2.2 Physical Validation

1. **Angle of repose**: Compare with experiments
2. **Segregation patterns**: Qualitative and quantitative
3. **Flow rates**: Hourglass, silo discharge
4. **Velocity profiles**: Shear flows

4.2.3 Convergence

1. **Grid refinement**: $\Delta x, \Delta y \to 0$
2. **Timestep refinement**: $\Delta t \to 0$
3. **Heterarchical refinement**: $n_m \to \infty$
4. **Statistical convergence**: Multiple realizations

5 Optimal Implementation Strategy

5.1 Recommended Architecture

Based on the analysis, an optimal implementation should:

5.1.1 1. Separate Physics from Numerics

```
class PhysicsModel {
    virtual double compute_probability(
        ParticleState& source,
        ParticleState& dest,
        Direction dir) = 0;
};
```

```
class NumericalSolver {
    virtual void step(Grid& grid,
                      PhysicsModel& physics,
                      double dt) = 0;
};
```

This allows swapping numerical methods without changing physics.

5.1.2 2. Implement Multiple Solvers

Mode 1: Stochastic (current HGD)

- Best for: Non-inertial, quasi-static flows
- Fast, simple implementation
- Good for segregation studies

Mode 2: Kinetic Monte Carlo

- Best for: Rare events, slow flows
- More accurate event selection
- Variable timestep

Mode 3: LBM-inspired

- Best for: Inertial flows, accuracy
- Distribution functions instead of discrete particles
- Better theoretical foundation

Mode 4: Hybrid HGD-DEM

- Best for: Transition regions
- DEM for dense regions, HGD for dilute
- Coupling at interfaces

5.1.3 3. Improved Conflict Resolution

Current: Random selection or shuffling
Recommended: Rate-based selection

$$P(\text{swap } i) = \frac{P_i}{\sum_j P_j} \tag{1}$$

where j indexes all swaps competing for the same destination.

5.1.4 4. Adaptive Timestep Control

```
double compute_safe_timestep(Grid& grid) {
    double max_velocity = grid.get_max_velocity();
    double dt_CFL = CFL_number * grid.dx / max_velocity;

    double max_probability = grid.get_max_swap_probability();
    double dt_prob = 1.0 / (max_probability / grid.dt);

    return min(dt_CFL, dt_prob);
}
```

5.1.5 5. Proper Inertia Handling

Follow the recommendations in `motion_models_analysis.pdf`, Option 1:

- Full velocity fields $u[i, j, k]$, $v[i, j, k]$
- Velocity swap with particles
- Collision damping: $v_{\text{new}} = e \cdot v_{\text{old}}$ where e is restitution coefficient
- Solid friction: $u_{\text{new}} = u_{\text{old}} - \mu \cdot P \cdot \Delta t$

5.2 Validation Strategy

5.2.1 Unit Tests

1. Conservation: Mass, momentum (when applicable)
2. Boundary conditions: No-flux, periodic
3. Single particle: Trajectories
4. Probability limits: $0 \leq P \leq 1$

5.2.2 Benchmark Problems

1. **Collapse**: Column collapse, comparison with experiments
2. **Segregation**: Binary mixture in rotating drum
3. **Hourglass**: Flow rate vs opening size
4. **Angle of repose**: Static pile formation
5. **Shear flow**: Velocity profiles

5.2.3 Comparison with Other Methods

1. **DEM**: Small systems ($\sim 10^4$ particles)
2. **Experiments**: Available data from literature
3. **Continuum**: $\mu(I)$ rheology predictions

6 Specific Recommendations for HGD

6.1 Short-term Improvements (Minimal Changes)

6.1.1 1. Add Robustness Checks

```
// In stream_core and move_voids_core
double P_tot = P_u + P_l + P_r;
if (P_tot > 1.0) {
    // Renormalize instead of error
    P_u /= P_tot;
    P_l /= P_tot;
    P_r /= P_tot;
    P_tot = 1.0;
}
```

6.1.2 2. Implement Proper Conflict Resolution

Replace random shuffle with rate-based selection:

1. Identify all swaps targeting same cell
2. Compute selection probability for each
3. Choose one swap probabilistically
4. Reject others

6.1.3 3. Add Validation Tests

Create systematic tests for:

- Mass conservation (should be exact)
- Momentum conservation (when inertia enabled)
- Grid convergence
- Statistical convergence (multiple runs)

6.2 Medium-term Improvements

6.2.1 1. Adaptive Timestep

Implement automatic timestep control based on CFL and probability limits.

6.2.2 2. Complete Inertia Implementation

Follow Option 1 from `motion_models_analysis.pdf`:

- Add per-particle velocities to `stream_core`
- Implement velocity swapping
- Add collision/friction models

6.2.3 3. Enhanced Boundary Conditions

- No-slip walls
- Stress-based walls
- Inflow/outflow boundaries

6.3 Long-term Improvements

6.3.1 1. Multiple Solver Modes

Implement template-based architecture allowing:

- Current stochastic method
- Kinetic Monte Carlo variant
- LBM-inspired variant

6.3.2 2. Hybrid Methods

Couple HGD with:

- DEM for accurate contact mechanics
- Continuum models for far-field

6.3.3 3. GPU Acceleration

Port to CUDA/HIP for massive parallelization.

7 Benchmarking Protocol

7.1 Proposed Benchmark Suite

7.1.1 Test 1: Mass Conservation

Setup: Random initial condition, run 1000 steps
Metric: $\Delta M/M_0 < 10^{-14}$ (machine precision)
Expected result: Exact conservation

7.1.2 Test 2: Grid Convergence

Setup: Same physics, vary grid resolution $\Delta x = L/N$ where $N = 32, 64, 128, 256$
Metric: Velocity profile RMS error vs N
Expected result: Second-order convergence

7.1.3 Test 3: Timestep Convergence

Setup: Same grid, vary $\Delta t = T/M$ where $M = 100, 200, 400, 800$
Metric: Final state RMS error vs Δt
Expected result: First-order convergence

7.1.4 Test 4: Statistical Convergence

Setup: Same parameters, $N_{\text{runs}} = 100$ different random seeds
Metric: Mean and standard deviation of observables
Expected result: $\sigma \propto 1/\sqrt{N_{\text{runs}}}$

7.1.5 Test 5: DEM Comparison

Setup: Small system (10^4 particles), both HGD and DEM
Metric: Velocity profiles, segregation patterns
Expected result: Qualitative agreement

7.1.6 Test 6: Experimental Validation

Setup: Reproduce published experiments
Metric: Segregation intensity, flow rates, angles of repose
Expected result: Within experimental uncertainty

7.2 Performance Benchmarking

Compare computational cost:

Method	Particles	Time/step	Memory
DEM	10^4	1.0 s	1 MB
HGD (Python)	10^6	0.1 s	100 MB
HGD (C++)	10^6	0.01 s	100 MB
HGD (C++ opt)	10^6	0.001 s	100 MB
LBM	10^6	0.001 s	10 MB

Table 4: Expected performance comparison (order of magnitude)

8 Conclusion

8.1 Summary

The HGD method occupies a useful niche:

- **Faster than DEM**: Can handle large systems
- **More detailed than continuum**: Captures segregation
- **Simpler than LBM**: Direct physical interpretation
- **Better for granular flows than LGA/LBM**: Designed for purpose

8.2 Current Limitations

1. **Conflict resolution**: Not physically consistent
2. **Inertia**: Incomplete implementation in production code
3. **Timestep stability**: No automatic control
4. **Validation**: Insufficient convergence studies
5. **Boundary conditions**: Limited types

8.3 Path Forward

Immediate priorities:

1. Implement probability renormalization
2. Add comprehensive validation tests
3. Complete inertia implementation in `core.cpp`
4. Document convergence behavior

Future directions:

1. Alternative solver modes (KMC, LBM-inspired)
2. Adaptive timestep control
3. Hybrid coupling with DEM/continuum
4. GPU acceleration

8.4 Optimal Implementation

For the current HGD method, the optimal implementation strategy is:

1. **Use C++ implementation** (`core.cpp`) for production
2. **Add inertia** following Option 1 recommendations
3. **Improve robustness** with probability renormalization
4. **Implement conflict resolution** using rate-based selection
5. **Add validation tests** for conservation and convergence
6. **Keep Python implementations** for research and validation

The method is fundamentally sound for its target application (slow, dense granular flows with segregation), but needs improvements in numerical robustness and completeness (especially inertia).

A Additional Reading

A.1 Lattice Methods

- Frisch et al. (1986): Lattice Gas Automata
- Chen & Doolen (1998): Lattice Boltzmann Method review
- Succi (2001): The Lattice Boltzmann Equation

A.2 Granular Flow Modeling

- Cundall & Strack (1979): DEM fundamentals
- GDR MiDi (2004): $\mu(I)$ rheology
- Goldhirsch (2003): Granular kinetic theory

A.3 Stochastic Methods

- Gillespie (1976): Kinetic Monte Carlo
- Voter (2007): KMC overview
- Bortz et al. (1975): n-fold way algorithm