Reversible Simulations of Elastic Collisions

KALYAN S. PERUMALLA, Oak Ridge National Laboratory
VLADIMIR A. PROTOPOPESCU, Oak Ridge National Laboratory

Consider a system of N identical hard spherical particles moving in a d-dimensional box and undergoing elastic, possibly multi-particle, collisions. We develop a new algorithm that recovers the pre-collision state from the post-collision state of the system, across a series of consecutive collisions, *with essentially no memory overhead*. The challenge in achieving reversibility for an n-particle collision (where, in general, $n \ll N$) arises from the presence of $nd - d - 1$ degrees of freedom (arbitrary angles) during each collision, as well as from the complex geometrical constraints placed on the colliding particles. To reverse the collisions in a traditional simulation setting, all of the particular realizations of these degrees of freedom (angles) during the forward simulation must be tracked. This requires memory proportional to the number of collisions, which grows very fast with N and d, thereby severely limiting the *de facto* applicability of the scheme. This limitation is addressed here by first performing a pseudo-randomization of angles, which ensures determinism in the reverse path for any values of n and d. To address the more difficult problem of geometrical and dynamic constraints, a new approach is developed which correctly samples the constrained phase space. Upon combining the pseudo-randomization with correct phase space sampling, perfect reversibility of collisions is achieved, as illustrated for $n \leq 3$, $d = 2$, and $n = 2$, $d = 3$. This result enables, for the first time, reversible simulations of elastic collisions with essentially zero memory accumulation. In principle, the approach presented here could be generalized to larger values of n. The reverse computation methodology presented here uncovers important issues of irreversibility in conventional models, and the difficulties encountered in arriving at a reversible model for one of the most basic and widely used physical system processes, namely, elastic collisions for hard spheres. Insights and solution methodologies, with regard to accurate phase space coverage with reversible random sampling proposed in this context, can help serve as models and/or starting points for other reversible simulations.

Categories and Subject Descriptors: C.5.1 [**Computer Systems Organization**]: Computer System Implementation—*Large and Medium Computers*; I.6.1 [**Computing Methodologies**]: Simulation and Modeling—*Discrete*; I.6.8 [**Computing Methodologies**]: Simulation and Modeling—*Types of Simulation (Discrete Event, Parallel)*

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Reverse execution; billiards; conservation laws; phase space coverage; reversible pseudo-random; time warp.

ACM Reference Format:
Perumalla, K. S., Protopopescu, V. A., 2012. Reversible Simulations of Elastic Collisions ACM Trans. Model. Comput. Simul. 0, 0, Article 0 (2013), 25 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Author's addresses: K. S. Perumalla and V. A. Protopopescu, Computational Sciences and Engineering Division, Oak Ridge National Laboratory.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

1.1. Background

Modeling and simulation of particle collisions for a wide range of collision types has been a subject of scientific interest for almost two centuries. However, while the *forward* simulation of collisions is relatively well understood [Truesdell and Muncaster 1980; Haile 1992], their *reversible* simulation is much less so. Here, we study the problem of modeling and simulating elastic collisions *reversibly*. In a system of N identical hard spherical particles colliding with each other in a d-dimensional box, every elastic collision, in general, yields an underspecified system of equations and inequalities [Truesdell and Muncaster 1980]. The underspecified nature of the system gives rise to interesting challenges for reversibility, such as the issue of memory accumulation and the need for unbiased phase space coverage.

In each multi-particle collision of $n \ll N$ hard particles in d dimensions, the velocities represent nd variables in the post-collision state, which are related to the nd velocities in the pre-collision state via conservation laws. Upon applying conservation of momenta and kinetic energy, one is left with $nd - d - 1$ unspecified degrees of freedom. *Deterministic* solutions to this underspecified system add new constraints that correspond to specific additional assumptions on the collisions. For example, in a 2-particle collision, determinism may be induced by exchanging velocity components along the line joining the centers of the two particles [Lubachevsky 1991; Marin 1997a; 1997b; Hontalas et al. 1989]. Such deterministic solutions are relatively easy to model, but they are physically unrealistic and artificially constrain the phase space. *Nondeterministic* solutions can be obtained by infusing the appropriate amount of randomization needed to cover the phase space of the underspecified system in a complete and unbiased manner. Of particular concern, is accounting for geometric constraints, *e.g.*, disallowing particle-overlapping at all times. The goal of our work is to develop a modeling and simulation framework which accurately and completely recovers the pre-collision state from the post-collision state of all particles involved in every collision in a sequence of collisions in the system, *with essentially no memory overhead*.

1.2. Applications

Hard sphere simulations continue to be used in several areas. Practical applications include simulation of food particulates in agricultural processes [Cleary and Sawley 2002], rocks in earthmoving equipment [Zhao et al. 2006], powders in manufacturing applications [Siiria and Yliruusi 2007], and particle-fluid interactions [Zhu et al. 2007], among others. Analytical problems in material sciences include dynamics and dense packings of granular materials [Lubachevsky 1991; Donev 2009]. Theoretical interests include simulating idealized dynamics as reference models, and simulation-based study of laws in thermodynamics [Broadwell 2001; Leff and Rex 2002]. Scenarios with thousands to millions of particles are of interest in many such applications, scales at which parallel processing is needed. Most of the models involve pseudo-random, elastic scattering of particles upon contact. Additional, domain-specific aspects such as angular momenta, friction, damping, and gravitational force may be introduced as needed by the application. Here, we focus on reversing the elastic scattering part of the collision model; additional variants constitute future work.

1.3. Motivation

The computational aspect of the reversible model is also motivated by practical and theoretical considerations.

(1) *Practical Need* The practical use of model reversibility stems from at least three purposes in which efficient rollback of computation is needed:

— *Scalable parallel synchronization* In parallel algorithms for event-driven molecular dynamics [Miller and Luding 2004; Donev 2009] simulations, concurrency is achieved by asynchronous updates to the particle states across processors. The asynchrony relies on the availability of a reversibility mechanism to take the system forward as well as *backward*, so that correct ordering can be reestablished when violations of inter-processor dependencies are discovered at runtime. Generalization called Time Warp and its variants [Jefferson 1985; Jefferson et al. 1987; Fujimoto 1990; Carothers et al. 1999; Perumalla 2007], employ combinations of forward and backward execution to reduce overall execution time.
— *Fault tolerance* In large-scale parallel simulations, asynchronous fault tolerance techniques [Manivannan and Singhal 1996] provide the best characteristics for recovering and restoring the parallel job from failures. They selectively restore simulation state of failed processors from disks, and roll back the non-failed processors back to fault point without having to go to disk. Asynchronous, in-memory rollback significantly increases the efficiency of fault tolerant operation, but requires support for backtracking the system state via either in-memory checkpointing or computing backwards.
— *Efficient debugging* Rolling back the code to a point in the past is useful in debugging simulations, backtracking from a point of discovery of a bug to the source point of the bug. However, rolling back by saving state upon every modification is memory-intensive or often nearly infeasible. Reversible models can be executed forward and backward at will, with little memory cost, especially for large simulations containing many thousands of particles.

(2) *Efficient Implementation* Reversal via log-generation techniques [Ross 1997; Biswas and Mall 1999] consume large amounts of memory and slow down forward execution due to memory copying [Carothers et al. 1999]. Reverse computation offers a significantly more efficient alternative. Performance gain is even more pronounced on emerging computational platforms of *accelerators* such as the graphical processing units (GPUs). GPUs offer very high computational effciency when the model exhibits a high ratio of computation to memory operations; for codes subject to rollback, reverse computation helps to significantly increase this ratio, whereas state saving reduces the ratio. Reverse computation delivers significant performance gains for rollback on conventional processors and accelerators.
(3) *Theoretical Advancement* New challenges, such as reversible sampling of a complex phase space, underlie reversible simulations of dynamical systems. Even the relatively simple problem of reversible elastic collision for hard spheres is shown to raise difficult problems in achieving a correct coverage of phase space. The approach and techniques proposed here underscore these challenges, provide the first solution to these problems in a realistic setting, and provide the basis for developing reversible simulation of more complex systems.

1.4. Related Work

Molecular dynamics (MD) covers a vast area, with many subtopics of interest, ranging from kinetic theory of gases to chemical reactions and protein folding. Each MD simulator is applicable for its intended use case(s). Moreover, simulators are written for specific execution styles, such as time-stepped or discrete event-based. Molecular dynamics simulation packages such as Desmond, LAMMPS, AMBER, GROMACS, and NAMD, are *time-stepped* simulation *frameworks* into which specific models can be incorporated with desired modeling features.

The common way to simulate hard particle dynamics is by using an approximation method called the Discrete Element Method or DEM [Cleary and Sawley 2002]. Note

that DEM is distinct from Discrete *Event* methods [Lubachevsky 1991; Krantz 1996]. In the DEM approximation method, which can be built on top of any of the aforementioned molecular dynamics packages, artificial buffer zones are introduced around particles to approximate pair-wise interactions. DEM simulations are limited by the fact that they cannot ensure preservation of hard-sphere semantics at all times, and also rely on heuristic (or highly conservative) setting of time steps. In contrast, our model is designed for use in exact simulation, with no approximation assumed, suitable for methodologies and scenarios employing fast discrete event simulations of hard spheres.

1.5. Organization

In Section 2, the problem is defined and terminology is introduced. The basic outline of our approach to reversal of elastic collisions is outlined in Section 3. This is followed by Section 4 which gives detailed reversal algorithms for 2-particle collisions (up to 3 dimensions) and 3-particle collisions (in 1 dimension). Results from experiments with a software realization of the algorithms are presented in Section 5. An estimation of the potential performance gains from reverse computation, in comparison to conventional state saving approaches, is given in Section 6. The findings are summarized in Section 7. Reversal of 3-particle collisions in 2 dimensions is treated in the online supplement appendix.

2. PROBLEM DEFINITION AND TERMINOLOGY

2.1. Reversibility Problem

Consider a system consisting of N identical hard spheres of diameter D in a d dimensional cubic box undergoing elastic collisions among themselves and/or with the walls of the box. Reversible simulation of particle collisions in the system means that, at any moment, the simulation can be stopped and executed backwards to arbitrary points in the past, potentially all the way to the initial state, such that the positions and velocities of particles are restored to the same values that the particles had at the chosen point in the past. *The problem is to achieve this reversible simulation with minimal or, ideally, no memory overhead*. The reversal scheme must support starting the system with arbitrary initial configurations, and must be able to evolve the system to arbitrary numbers of collisions into the future. Moreover, given that particles are marked with identifiers from 1 to N, the system state must be restored to the correct initial identifier assignation.

An important requirement of the collision model is the uniform (unbiased) coverage of all available phase space. In particular, the scattering law upon each collision must uniformly sample the full range of restitution angles available to each pair of particles upon their collision.

2.2. Collision Configurations and Constraints

Here we consider collisions of three types: (1) single particle-wall collisions, (2) n-particle-wall collisions, and (3) n-particle collisions ($n > 1$). The first type is straightforward to reverse. The second type is treated, without loss of generality, as a simultaneous set of individual wall-particle collisions, followed by n-particle collision (for any collisions that remain after the application of individual wall-particle collisions). The third is the more complex problem treated in the remainder of the paper.

The collision of a particle with a wall is modeled by changing the sign of the velocity component that is orthogonal to the wall. If a particle touches more than one wall simultaneously at an edge or a corner of the box, all the appropriate velocity components change their sign. Other commonly used deterministic boundary conditions such

as periodic wall boundaries [Lubachevsky 1991; Marin 1997a; 1997b; Krantz 1996] can be reversed analogously.

For completeness, we list here the set of system configurations that are either trivial configurations that are not of interest, or are simple to solve separately without needing the complexity of our algorithm.

— Trivial or degenerate situations, in which particles never come in contact. Examples include all particles being at rest (zero kinetic energy), which is clearly an uninteresting case.
— Situations in which particles do come in contact, but do not exchange momentum. Examples in the first category include initial conditions in which:
  — all particles are at rest, or
  — all N particles have velocities along one coordinate axis (say x) and the area of the particles' projections on the (y, z) plane is $\frac{\pi N D^2}{4}$ (this perpetually collides particles with walls but never results in any particle-particle collisions).

  Examples in the second category include grazing collisions, for which the line passing through the particles' centers at the encounter time is perpendicular to their (parallel or anti-parallel) velocities. This condition is easily detected and either ignored or a degenerate collision operator can be defined separately and applied.

2.3. Dynamics and Geometry

In an n-particle collision, let $\vec{V'}_i$ be the pre-collision velocity of particle i, and $\vec{V}_i$ its post-collision velocity. For every pair of particles i and j that are in contact in the collision, let $\vec{r}_{ji}$ be the vector from center of particle j to that of particle i at the collision moment. Let the total momentum of the n colliding particles be $\vec{M}$ and their total kinetic energy be E. Then, the dynamics and geometry require the following:

$$\left.\begin{aligned} \sum_{i=1}^{n} \vec{V'}_i = \sum_{i=1}^{n} \vec{V}_i = \vec{M} \\ \sum_{i=1}^{n} (\vec{V'}_i)^2 = \sum_{i=1}^{n} (\vec{V}_i)^2 = E > 0 \end{aligned}\right\} \text{Dynamics}, \tag{1}$$

$$\left.\begin{array}{l|l} \forall i, j \text{ such that particles} & \vec{r}_{ji} \cdot (\vec{V'}_i - \vec{V'}_j) < 0 \,(\text{pre-collision}) \\ i \text{ and } j \text{ are in contact} & \vec{r}_{ji} \cdot (\vec{V}_i - \vec{V}_j) > 0 \,(\text{post-collision}) \end{array}\right\} \text{Geometry}. \tag{2}$$

In Equation (2), the strictness of the inequalities ($<$ instead of $\leq$, and $>$ instead of $\geq$) ensures that grazing collisions are excluded.

Let $d_n = nd - d - 1$ denote the number of degrees of freedom in the collision.

Denote by $\Phi = \{\phi_k | 1 \leq k \leq d_n\}$ the set of "parameters" that, given $\vec{M}$ and E, uniquely determines the set of velocities $\boldsymbol{V} = \{\vec{V}_i | 1 \leq i \leq n\}$ of all particles in an n-particle collision. Let Φ' correspond to the parameters encoding the pre-collision velocities, and let Φ (or, where ambiguous, Φ'') encode the post-collision velocities. In 2-particle collisions, the parameters correspond to geometrical angles of points on the surface of a d_n-sphere, whereas, in 3-particle collisions, the parameters are different from the geometrical angles of points on the surface of a $(d_n + 1)$-dimensional ellipsoid.

In randomly sampling the phase space spanned by the d_n parameters, at least d_n random numbers must be generated per collision. Reversible random number generators are used to generate the needed random samples per collision, each sample value uniformly distributed in $[0, 1)$. The generators themselves require only a con-

stant amount of memory. In practice, the memory per generator is often only a few bytes long. In theory, the memory only needs to be independent of the number of collisions being simulated. Additional considerations in random number generation that are critical to reversibility are discussed in Section 4.5.

Our reversible collision algorithm uses the following mappings.

$\boldsymbol{V}$**-to-**$\boldsymbol{\Phi}$: Given $\boldsymbol{V}$, this mapping determines the set of angles Φ that uniquely determines $\boldsymbol{V}$. In forward execution, this mapping will be applied on pre-collision velocities $\boldsymbol{V}'$ to obtain $\Phi^{'}$. In reverse execution, this mapping function will be used to determine Φ from the post-collision velocities $\boldsymbol{V}$ in a collision.

$\boldsymbol{\Phi}$**-to-**$\boldsymbol{V}$: Given the angles Φ, together with $\vec{M}$ and E, this mapping reconstructs all corresponding $\boldsymbol{V}$. In forward execution, this mapping will be used to generate the post-collision velocities $\boldsymbol{V}$ based on reversibly computed values of Φ. The same mapping function will also be used in reverse execution to determine the pre-collision velocities $\boldsymbol{V}'$ from the recovered value of $\Phi^{'}$ of a collision.

$\boldsymbol{G}$: Let $\boldsymbol{G} = \{G_k | 1 \leq k \leq d_n\}$ denote the set of pseudo random numbers generated for each collision, where each $G_k \in [0, 1)$. The generator used to generate $\boldsymbol{G}$ is reversible, of high quality, and with a sufficiently long period. The generator can either be a single stream or contain d_n independent, parallel streams.

$\boldsymbol{G}$**-to-**$\boldsymbol{\Psi}$: Further, let $\Psi = \{\psi_k | 1 \leq k \leq d_n\}$, be the set of angle *offsets* generated from random numbers $\boldsymbol{G}$. In both forward as well as reverse execution, the $\boldsymbol{G}$**-to-**$\boldsymbol{\Psi}$ function will be used as a deterministic mapping from uniform random numbers to angle offsets. The specific mapping function depends on n and d, but the function only depends on the (reversible) random numbers, total momenta and total energy; it does not depend on the individual velocities of particles in collision.

In the remainder of the article, unless otherwise specified, all arithmetic on the angles will be performed modulo 2π.

2.4. Simplified Notation for $2 \leq n \leq 3$, $1 \leq d \leq 3$

When dealing with collisions in which at most three particles are in contact with each other, in 1- or 2-dimensional space (and also in the case in which two particles are in contact with each other in 3-dimensional space), a simplified notation is used in the remaining of the article to refer to their velocities, momenta, and energy. The letters a–f will be used to refer to the components of the velocities along the x, y, and z directions. The letters a'–f' will be used to refer to the corresponding pre-collision values of the velocity components. The letters α, β, and γ will be used as the sums of momenta along the x, y and z spatial directions respectively, and δ will be used for total kinetic energy. Thus, for example, in a 2-dimensional, 2-particle collision, the post-collision equations will be written as $a+b = \alpha$ (total momentum in the x-direction), $c + d = \beta$ (total momentum in the y-direction), and $a^2 + b^2 + c^2 + d^2 = \delta$ (total energy), where $a = V_{1x}$, $b = V_{2x}$, $c = V_{1y}$ and $d = V_{2y}$ are the velocities of the two particles in the x and y directions. In the remainder of the article, note that $\delta > 0$, since the system must have non-zero kinetic energy for collisions to occur.

3. SKELETON OF THE ALGORITHM

In any n-particle collision, the information available at hand during forward execution are the pre-collision velocities as well as the ranges of the angles that determine the range of permissible post-collision velocities. The pre-collision velocity configuration of all the n colliding particles can be uniquely encoded in terms of the total momentum, total energy, and the specific values of the free angles. Since all collisions are elastic, the total momentum and energy do not need any memory to recover. The problem thus reduces to that of developing a one-to-one mapping of pre-collision and post-collision

angles, while still uniformly sampling all available phase space for the angles at every collision.

In order to properly sample the available phase space, we use pseudo random numbers to select the angle values from the permissible ranges. The reversibility of the pseudo random number generators ensures the ability to go forward as well as backward in the random number sequence as needed, during forward and reverse execution, respectively.

3.1. Reversible Collision Operation

The problem reduces now to developing a collision algorithm that (a) takes d_n pseudo random numbers, each uniformly distributed in $[0, 1)$, and gives d_n reversible random angle offsets that satisfy the pre-collision phase space constraints, and (b) recovers the pre-collision angles from the random offsets recovered upon backward execution.

Once such a collision algorithm is developed, it can be used to uniquely recover the pre-collision velocities as follows. First, the random number sequence is reversed, thereby recovering the random numbers that were used in the forward execution. These are then used to recreate the random offsets that were used in the forward execution. The random offsets are applied in the opposite direction on the post-collision values of the free angles to uniquely recover the pre-collision values of the free angles, which in turn uniquely give the pre-collision velocities.

3.2. Collision Sequences

Collision *sequences* are modeled using standard techniques [Lubachevsky 1991; Miller and Luding 2004; Krantz 1996], by which, at every step of forward evolution, the time for next collision of each particle is determined, time is advanced to the earliest collision time, the particles undergoing collision are determined, their pre-collision velocities are transformed by our reversible algorithm to give post-collision velocities, and the process is repeated.

In order to reverse the collision *sequence*, it is necessary to recover the most recent collision event in the past. That event is obtained by reversing the direction of all particles' velocities and employing the usual forward algorithm for determining the next earliest collision. The event represents information on the amount of time, dt, to go back in time, and the identities of the colliding particles. The system is then stepped back in time by dt units by changing the sign of the velocities of all particles, and linearly transporting them for dt units. At that moment, clearly, the colliding particles would be found to be in contact. The reverse collision algorithm is then applied on the particles in contact. This process is repeated iteratively until the time of interest in the past is reached.

3.3. General and Specific Settings

This general reversal scheme is applicable in principle to any values of n and d. However, the actual permissible ranges of the free angle values remain to be determined for each (n, d) pair. As illustrations, we develop the geometrical constraints for the subsets $n \leq 3, d \leq 2$ and $n = 2, d = 3$. While the development of the permissible angle ranges for $n = 2$ is relatively straightforward, those for $n = 3$ become complex starting even from $d = 1$.

4. ALGORITHM IMPLEMENTATION

Here, we determine the phase space of the permissible reversible random offsets that need to be sampled for 2-particle collisions in 1, 2, and 3 dimensions, and for 3 particles in one dimension. The case of 3 particles in 2 dimensions is relegated to the appendix.

4.1. Reversal for 2-Particle Collisions in 1 Dimension

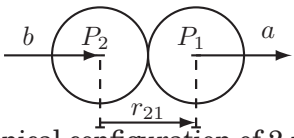

Fig. 1: Canonical configuration of 2 particles in 1 dimension

In a collision of 2 particles in 1-dimensional space, as shown in Figure 1, let P_1 be the particle on the right moving with velocity a, and P_2 be the particle on the left moving with velocity b. The constraints on dynamics and post-collision geometry are:

$$\left.\begin{array}{r} a+b=\alpha \\ a^2+b^2=\delta,\ 2\delta \quad >\alpha^2 \end{array}\right\} \text{Dynamics}, \qquad (3) \qquad \left.\begin{array}{l} r_{21}\cdot(a-b)>0 \\ \text{for } r_{21}=D>0 \end{array}\right\} \text{Geometry}. \quad (4)$$

The pre-collision geometrical constraints are obtained by replacing > 0 by < 0 in the post-collision constraints. The system is fully defined without any free angles, giving deterministic one-to-one mapping from pre-collision to post-collision velocities and hence reversibility is unambiguous. Nevertheless, this configuration is treated here for completeness. The equations of motion imply that:

$$a=\frac{\alpha}{2}+\frac{\sqrt{2\delta-\alpha^2}}{2},\ \text{and}\ b \quad =\frac{\alpha}{2}-\frac{\sqrt{2\delta-\alpha^2}}{2} \Big\} \text{Dynamics}, \qquad (5)$$

$$r_{21}\cdot(a-b)=+r_{21}\sqrt{2\delta-\alpha^2}>0 \quad \Big\} \text{Geometry}, \qquad (6)$$

i.e., the dynamical solution satisfies the geometrical constraint.

4.2. Reversal for 2-Particle Collisions in 2 Dimensions

In a collision of 2 particles in 2-dimensional space, let a and c be the x and y velocity components of the particle with the smaller identifier, and b and d be the corresponding velocity components of the other particle. The equations of motion and post-collision geometry are:

$$\left.\begin{array}{c} a+b=\alpha,\ c+d=\beta \\ a^2+b^2+c^2+d^2=\delta,\ \text{and}\ 2\delta>\alpha^2+\beta^2 \end{array}\right\} \text{Dynamics}, \qquad (7)$$

$$\left.\begin{array}{r} r_{21x}\cdot(a-b)+r_{21y}\cdot(c-d)>0 \\ \text{for some } r_{21x}, r_{21y}>0 \end{array}\right\} \text{Geometry}. \qquad (8)$$

The pre-collision geometrical constraints are obtained by replacing > 0 by < 0 in the post-collision constraints. The equations of motion imply that:

$$\left(a-\frac{\alpha}{2}\right)^2+\left(c-\frac{\beta}{2}\right)^2=\frac{2\delta-(\alpha^2+\beta^2)}{4}=R^2, \qquad (9)$$

i.e., the point (a, c) lies on a circle of radius $R=\sqrt{2\delta-(\alpha^2+\beta^2+\gamma^2)}/2$ centered at ($\alpha/2$, $\beta/2$), whose parametric equations are given by:

$$a=\frac{\alpha}{2}+R\cos\phi_1\ ,\ c \quad =\frac{\beta}{2}+R\sin\phi_1\ ,\ \text{and}\ \phi_1\in[0,2\pi). \qquad (10)$$

Equation (10) provides the $\boldsymbol{V}$**-to-**$\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}$**-to-**$\boldsymbol{V}$ mapping functions for 2-dimensional 2-particle collisions. Given (a', b', c', d'), a unique $\phi_1^{'}$ can be determined. Similarly, given $(\phi_1, \alpha, \beta, \delta)$, (a, b, c, d) can be uniquely determined.

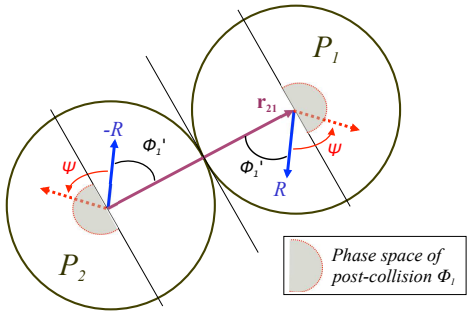

Fig. 2: Configuration of 2 particle collision in 2 dimensions in the center-of-mass frame

Given the pre-collision angle $\phi_1^{'}$, the problem at hand is the generation of a reversible random offset ψ from $\phi_1^{'}$ to give the post-collision angle $\phi_1 = \phi_1^{'} + \psi$ which can then be used to determine the post-collision velocities. Since the phase space of ϕ_1 is $[0, 2\pi)$, it is necessary to sample the random offset ψ also from the same range, in order to ensure full phase space coverage independent of $\phi_1^{'}$. This is illustrated in Figure 2 in which the colliding pair is visualized in its center-of-mass frame of reference (in this particular case of 2-particles in 2-dimensions, the geometric angle ϕ_1 indeed corresponds to the degree of freedom, but this does is not true in general). Moreover since the circumference of the circle is uniformly sampled by uniformly sampling the angle subtended at the center, it is sufficient to generate ψ uniformly from 0 to 2π. Thus, $\psi \in [0, 2\pi)$ is generated with a $\boldsymbol{G}$**-to-**$\boldsymbol{\Psi}$ mapping given by $\psi = 2G\pi$, with a uniformly distributed random number $G \in [0, 1)$.

However, the ϕ_1 value thus computed may violate the geometry constraint Equation (8) on post-collision velocities. If such a violation occurs with the generated random offset, the ϕ_1 value can be "corrected" by adding π to it; the addition of π guarantees satisfaction of the geometry constraint on post-collision velocities because of the following observations:

$$\begin{aligned} &\text{Since } (a-b) = 2R\cos\phi_1, \text{ and } (c-d) = 2R\sin\phi_1, \\ &\quad \text{if } r_{21x}\cdot(a-b) + r_{21y}\cdot(c-d) && < 0 \\ &\text{then } r_{21x}\cdot(2R\cos\phi_1) + r_{21y}\cdot(2R\sin\phi_1) && < 0 \\ &\implies r_{21x}\cdot(2R\cos(\phi_1+\pi)) + r_{21y}\cdot(2R\sin(\phi_1+\pi)) && > 0. \end{aligned} \tag{11}$$

Thus, the post-collision angle, using the random angle offset ψ, is computed as either $\boldsymbol{F} : \phi_1 = \phi_1^{'} + \psi$ or $\boldsymbol{F}_\pi : \phi_1 = \phi_1^{'} + \psi + \pi$. In reverse execution, the pre-collision angle $\phi_1^{'}$ is recovered as $\boldsymbol{R} : \phi_1^{'} = \phi_1 - \psi$ or $\boldsymbol{R}_\pi : \phi_1^{'} = \phi_1 - \psi - \pi$. If the fact that π was added by the forward execution is somehow remembered (*i.e.*, whether $\boldsymbol{F}$ or $\boldsymbol{F}_\pi$ was applied), then the correct pre-collision angle can be recovered during reversal (by applying $\boldsymbol{R}$ or $\boldsymbol{R}_\pi$ respectively).

Our key observation here is that it is possible to avoid having to "remember" whether the π offset was added or not. If π was added in forward execution, then, in reverse execution, the fact that $\psi + \pi$ must be used, as opposed to ψ, can be detected by a violation of the geometry constraint on the pre-collision velocities if π is not added. The correct value of $\phi_1^{'} = \phi_1 - \psi - \pi$ is thus possible to be recovered correctly.

To adopt this approach, it is necessary and sufficient to show that no ambiguity exists during the reverse execution regarding which of $\boldsymbol{F}$ or $\boldsymbol{F}_\pi$ was executed in the forward execution, so that the correct operation, $\boldsymbol{R}$ or $\boldsymbol{R}_\pi$, is applied for correct reversal. In other words, it is necessary and sufficient to satisfy the following conditions:

- $\boldsymbol{R}(\boldsymbol{F}(\phi)) = \phi$
- $\boldsymbol{R}_\pi(\boldsymbol{F}(\phi)) \neq \phi$
- $\boldsymbol{R}_\pi(\boldsymbol{F}_\pi(\phi)) = \phi$
- $\boldsymbol{R}(\boldsymbol{F}_\pi(\phi)) \neq \phi$
- $\boldsymbol{F}(\phi) \neq \boldsymbol{F}_\pi(\phi))$
- $\boldsymbol{R}(\phi) \neq \boldsymbol{R}_\pi(\phi))$.

These conditions can be proven as follows. Let $CS(\phi) = r_{21x} \cdot (2R\cos\phi) + r_{21y} \cdot (2R\sin\phi)$. We know that $\boldsymbol{F}$ and $\boldsymbol{F}_\pi$ are exclusive with respect to satisfaction of Equation (8) with ϕ_1, *i.e.*, $\boldsymbol{F}$ satisfies $CS(\phi_1) > 0$, if and only if $\boldsymbol{F}_\pi$ satisfies $CS(\phi_1) < 0$. Similarly, $\boldsymbol{R}$ and $\boldsymbol{R}_\pi$ are exclusive with respect to satisfaction of Equation (8) with $\phi_1^{'}$, *i.e.*, $\boldsymbol{R}$ satisfies $CS(\phi_1^{'}) < 0$, if and only if $\boldsymbol{R}_\pi$ satisfies $CS(\phi_1^{'}) > 0$.

The geometrical constraints require that the post-collision velocities satisfy $CS(\phi_1) > 0$ and the pre-collision velocities satisfy $CS(\phi_1^{'}) < 0$. These requirements are satisfied if $\boldsymbol{R}$ is used to reverse $\boldsymbol{F}$, or $\boldsymbol{R}_\pi$ is used to reverse $\boldsymbol{F}_\pi$, respectively. In other words, from Equation (11), we know that:

- if $\boldsymbol{F}$ satisfies $CS(\phi_1) > 0$ in forward, then $\boldsymbol{R}$ satisfies $CS(\phi_1^{'}) < 0$ in reverse, and
- if $\boldsymbol{F}_\pi$ satisfies $CS(\phi_1) > 0$ in forward, then $\boldsymbol{R}_\pi$ satisfies $CS(\phi_1^{'}) < 0$ in reverse.

Finally, the ambiguity is fully resolved when the following are also satisfied:

- If $\boldsymbol{F}$ satisfies $CS(\phi_1) > 0$, then $\boldsymbol{R}_\pi$ violates $CS(\phi_1^{'}) < 0$.
- If $\boldsymbol{F}_\pi$ satisfies $CS(\phi_1) > 0$, then $\boldsymbol{R}$ violates $CS(\phi_1^{'}) < 0$.

The former can be proved as follows, and the latter can be proved along similar lines: Since $CS(\phi_1^{'}) < 0$ (pre-collision), and $\boldsymbol{F}$ gives $CS(\phi_1) > 0$ (post-collision), $\boldsymbol{R}_\pi$ gives $CS(\phi_1 - \psi - \pi) = CS(\overline{\phi_1^{'} + \psi} - \psi - \pi) = CS(\phi_1^{'} - \pi) = -CS(\phi_1^{'})$. This implies $CS(\phi_1^{'}) = -CS(\phi_1^{'})$, which is a contradiction.

The forward and reverse algorithms are given in Procedure 1 and Procedure 2 respectively. This completes the generation of reversible random offsets for all configurations of 2-particle collisions in 2 dimensions, ensuring full phase space coverage with zero memory overhead.

Procedure 1 ($\phi_1^{'} \longrightarrow \phi_1$): *Forward Function for 2-Particle in 2 Dimensions*

1: $\psi \leftarrow 2G\pi$ {generate a random offset from $[0, 2\pi)$}
2: $\phi_1 \leftarrow (\phi_1^{'} + \psi) \mod 2\pi$ {post-collision is pre-collision offset by randomized ψ}
3: {Next, if post-collision is converging, correct it to be diverging}
4: **if** $r_{21x} \cdot \cos\phi_1 + r_{21y} \cdot \sin\phi_1 < 0$ **then**
5: $\quad \phi_1 \leftarrow (\phi_1 + \pi) \mod 2\pi$
6: **end if**

Procedure 2 ($\phi_1 \longrightarrow \phi_1^{'}$): *Reverse Function for 2-Particle in 2 Dimensions*

1: $\psi \leftarrow 2G\pi$ {recover the random offset}
2: $\phi_1^{'} \leftarrow (\phi_1 - \psi) \mod 2\pi$ {initial guess at pre-collision angle}
3: {Next, if pre-collision is diverging, correct it to be converging}
4: **if** $r_{21x} \cdot \cos\phi_1 + r_{21y} \cdot \sin\phi_1 > 0$ **then**
5: $\quad \phi_1^{'} \leftarrow (\phi_1^{'} - \pi) \mod 2\pi$
6: **end if**

4.3. Reversal for 2-Particle Collisions in 3 Dimensions

In a collision of 2 particles in 3-dimensional space, the equations of motion and post-collision geometry are:

$$\left.\begin{aligned} a+b=\alpha,\, c+d=\beta,\, e+f=\gamma \\ a^2+b^2+c^2+d^2+e^2+f^2=\delta, \text{ and } 2\delta > \alpha^2+\beta^2+\gamma^2 \end{aligned}\right\} \text{Dynamics}, \tag{12}$$

$$\left.\begin{aligned} r_{21x}\cdot(a-b)+r_{21y}\cdot(c-d)+r_{21z}\cdot(e-f)>0 \\ \text{for some } r_{21x}, r_{21y}, r_{21z} > 0 \end{aligned}\right\} \text{Geometry}. \tag{13}$$

The pre-collision geometrical constraints are obtained by replacing > 0 by < 0 in the post-collision constraints. The equations of motion imply that:

$$\left(a-\frac{\alpha}{2}\right)^2+\left(c-\frac{\beta}{2}\right)^2+\left(e-\frac{\gamma}{2}\right)^2=\frac{2\delta-(\alpha^2+\beta^2+\gamma^2)}{4}=R^2, \tag{14}$$

i.e., the point (a, c, e) lies on a sphere of radius $R=\frac{1}{2}\sqrt{2\delta-(\alpha^2+\beta^2+\gamma^2)}$ centered at $(\frac{\alpha}{2}, \frac{\beta}{2}, \frac{\gamma}{2})$. whose parametric equations are given by:

$$\begin{aligned} a=\frac{\alpha}{2}+R\sin\phi_1\sin\phi_2,\, c=\frac{\beta}{2}+R\sin\phi_1\cos\phi_2,\, e=\frac{\gamma}{2}+R\cos\phi_1, \\ \phi_1\in[0,\pi], \text{ and } \phi_2\in[0,2\pi)\ . \end{aligned} \tag{15}$$

Equation (15) provides the $\boldsymbol{V}$**-to-**$\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}$**-to-**$\boldsymbol{V}$ mapping functions for 2-particle collisions in 3 dimensions.

In determining $\boldsymbol{\Phi}$ from $\boldsymbol{V}$, if $\phi_1 = 0$ or $\phi_1 = \pi$, then, the value of ϕ_2 is immaterial. In that case, we choose to set $\phi_2 = 0$. Setting ϕ_2 thus, without regard to $\phi_2^{'}$, loses information about $\phi_2^{'}$ when $\phi_1 = 0$. To deal with this rare special case, the value of $\phi_2^{'}$ can be logged in the forward collision, and restored from the log in the backward collision. Note that this is logged in forward execution *only* if $\phi_1 = 0$, and *not* for every collision. In the reverse execution, $\phi_2^{'}$ is recovered from the log only if $\phi_1 = 0$.

For the $\boldsymbol{G}$**-to-**$\boldsymbol{\Psi}$ mapping, the offsets ψ_1 and ψ_2 are obtained by uniformly sampling the surface of a unit sphere. Any algorithm for this purpose can be employed (e.g., [Marsaglia 1972]), using two random numbers $G_1 \in [0, 1]$ and $G_2 \in [0, 1)$.

Similar to the 2-dimensional 2-particle case, the post-collision angles are computed as $\phi_1 = (\phi_1^{'}+\psi_1) \mod 2\pi$ or $\phi_1 = (\phi_1^{'}+\psi_1+\pi) \mod 2\pi$, and $\phi_2 = (\phi_2^{'}+\psi_2) \mod 2\pi$. The choice of whether π is added to ψ is determined by the one that satisfies the geometrical condition given by Equation (13). The proof of reversibility of the computation of ϕ_1 is analogous to the one in the preceding sub-section.

This completes the generation of reversible random offsets for all configurations of 2-particle collisions in three dimensions, ensuring full phase space coverage with zero memory overhead.

4.4. Reversal for 3-Particle Collisions in 1 Dimension

In this section, we determine the phase space of the permissible reversible random offsets that needs to be sampled for 3-particle collisions in 1 dimension. As mentioned earlier, the treatment for 3-particle collisions in 2 dimensions is considerably lengthier, and is relegated to the online supplement appendix.

4.4.1. Solving the Equations of Motion. The equations of motion are:

$$\left.\begin{array}{l} a+b+c=\alpha \\ a^2+b^2+c^2=\delta,\ 3\delta \quad > \alpha^2 \end{array}\right\} \text{Dynamics.} \tag{16}$$

The post-collision geometrical constraints are given by:

$$\left[\begin{array}{c} \text{Only two of these three need be} \\ \text{satisfied for any given geometric} \\ \text{configuration } r_{21}, r_{32}, r_{13} > 0 \end{array}\right] \left.\begin{array}{l} r_{21}\cdot(a-b)>0, \\ r_{32}\cdot(b-c)>0, \\ r_{13}\cdot(c-a)>0 \end{array}\right\} \text{Geometry.} \tag{17}$$

In Equation (17), when any two inequalities are satisfied, the third one is resulting. This is because the two inequalities that are satisfied imply a specific ordering of the three particles along the single dimension, which in turn automatically satisfies the remaining third constraint. The pre-collision geometrical constraints are obtained by replacing > 0 by < 0 in the post-collision constraints. The solution to the equations of motion satisfies:

$$\bar{a}^2 + \left(\frac{\bar{b}-\frac{\sqrt{2}}{3}\alpha}{\frac{1}{\sqrt{3}}}\right)^2 = \delta - \frac{\alpha^2}{3}, \text{ where } \bar{a} \quad = \frac{a-b}{\sqrt{2}}, \text{ and } \bar{b} = \frac{a+b}{\sqrt{2}}, \tag{18}$$

which leads to the parametrization:

$$\bar{a} = \frac{\lambda}{\sqrt{2}}\cos\phi_1, \bar{b} \quad = \frac{\sqrt{2}}{3}\alpha + \frac{\lambda}{\sqrt{2}\sqrt{3}}\sin\phi_1, \lambda = \sqrt{2}\sqrt{\delta - \frac{\alpha^2}{3}}, \text{ and } \phi_1 \quad \in [0, 2\pi). \tag{19}$$

Converting to a, b and c, we get:

$$\begin{array}{c} a = \frac{\alpha}{3} + \frac{\lambda}{2}(\cos\phi_1 + \frac{1}{\sqrt{3}}\sin\phi_1),\ b = \frac{\alpha}{3} + \frac{\lambda}{2}(-\cos\phi_1 + \frac{1}{\sqrt{3}}\sin\phi_1), \\ c = \frac{\alpha}{3} + \frac{\lambda}{2}(-\frac{2}{\sqrt{3}}\sin\phi_1),\ \lambda = \sqrt{2}\sqrt{\delta - \frac{\alpha^2}{3}}, \text{ and } \phi_1 \in [0, 2\pi). \end{array} \tag{20}$$

Equation (20) provides the $\boldsymbol{V}$**-to-**$\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}$**-to-**$\boldsymbol{V}$ mapping functions for 1-dimensional 3-particle collisions.

4.4.2. Sampling the Ψ Phase Space. Analogously to the 2-particle 2-dimensional case, the phase space of the post-collision angle is sampled by generating a random ϕ_1 as a random offset $\psi_1 \in [0, 2\pi)$ from the pre-collision angle ϕ_1'. The mapping function from a uniform random number $G \in [0, 1)$ to ψ_1 is more complex than that for the 2-particle case. In the 2-particle case, the phase space for velocities lies on the circumference of a circle, which can be sampled simply by uniformly sampling the angle subtended at the center. However, the phase space of the velocities in the 3-particle case lies on the circumference of an ellipse, which cannot be sampled simply as $\psi_1 = 2G\pi$. Instead, any correctly unbiased procedure for generating the angle by sampling the circumference of an ellipse can be employed for the purpose of defining the $\boldsymbol{G}$**-to-**$\boldsymbol{\Psi}$ mapping function. One such method is described in the next section.

4.4.3. Sampling the Circumference of the Ellipse. Here, we present a new procedure for uniformly sampling a point from the perimeter of an ellipse. The procedure is designed to be suitable for use in reversible execution, requiring exactly one random number per sample. While rejection-based procedures exist for this problem, they cannot be used here due to their irreversibility, as explained later in Section 4.5.

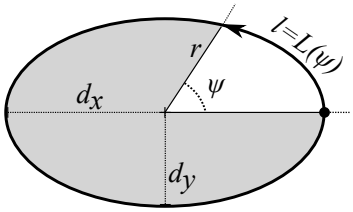

Fig. 3: Sampling scheme for a uniformly selected random point on an ellipse

Consider the ellipse $\left(\frac{x}{d_x}\right)^2 + \left(\frac{y}{d_y}\right)^2 = 1$ shown in Figure 3. Let $L : \psi \rightarrow \ell$ be a function that maps any angle [1] ψ, $0 \leq \psi < 2\pi$, to the length ℓ of the arc moving counter-clockwise along the circumference of the ellipse from $(d_x, 0)$ to $(r\cos\psi, r\sin\psi)$, where $r^2 = d_x^2 d_y^2/(d_y^2\cos\psi^2 + d_x^2\sin\psi^2)$. Let $L^{-1} : \ell \rightarrow \psi$ be the inverse function that determines the angle corresponding to any given arc length ℓ. Thus, $L^{-1}(L(\psi)) = \psi$. A random point on the circumference of the ellipse can be obtained as the end of the arc whose length is $G \cdot L(2\pi)$, and the angle corresponding to that point can be obtained as $\psi_1 = L^{-1}(G \cdot L(2\pi))$, which serves as the ***G*-to-Ψ** mapping for the 3-particle collisions in 1 dimension.

The value of $L(2\pi)$ is computable by different methods. For example, $L(2\pi) = \pi(d_x + d_y)\sum_{n=0}^{\infty}\binom{0.5}{n}^2 h^n$, where $h = \frac{(d_x - d_y)^2}{(d_x + d_y)^2}$, which can be computed to any desired precision. The value of $L^{-1}(\ell_{given})$ can be obtained by a bisection method (binary search) that starts with the lowerbound $\psi_{lower} = 0$ upperbound $\psi_{upper} = 2\pi$, and an initial estimate value of $\psi_{guess} = \pi$, and repeatedly adjusts the lowerbound or upperbound to the guess value (depending on whether $L(\psi_{guess}) < \ell_{given}$ or $\ell_{given} < L(\psi_{guess})$ respectively), until the correct angle corresponding to ℓ_{given} is determined.

For the ellipse in Equation (18), $d_x = \sqrt{\delta - \frac{\alpha^2}{3}}$ and $d_y = \frac{d_x}{\sqrt{3}}$. Since d_x and d_y only depend on δ and α, the value of ψ_1 can be generated and recovered (re-generated) independently of individual particle velocities.

Note that the computational burden in sampling the ellipse occurs both in log-based approaches and in our approach. Thus, the forward portions incur the same computational cost. However, we use the same computational procedure on the reverse path, to re-generate the sample and use it in the reverse procedure. Since log-based approaches rely on memory, they do not incur computational cost on the reverse path, but incur extra memory copying cost in forward execution. In a normal, well-balanced parallel execution, since reversals of collisions are far fewer than forward collisions, the extra computational cost of our approach in the reverse procedure is much less than the savings gained in foward execution compared to log-based approaches, resulting in an overall reduction in run time and memory.

4.4.4. Resolving the Geometrical Constraints. Note that, with the sampled $\phi_1 = \phi_1^{'} + \psi$, it is possible to correctly recover the pre-collision angle $\phi_1^{'}$, from which the values of the pre-collision velocities can be recovered, but not necessarily their correct assignation to the identities of the particles. Thus, there still remains the problem of uniquely recovering the configuration of the particles, since the preceding solution provides for two different but equivalent configurations that only differ in that their left and right

[1] We will use the terms angle and parameter interchangeably, with the understanding that the angle is in fact the parameter in the representation of the point on the ellipse, and *not* the geometrical angle.

particle identities are swapped. Without modifying the procedure, one additional bit of memory would be needed for each collision to disambiguate between the two configurations. Since the aim is to completely eliminate memory accumulation, the model needs additional development for reversibility, as presented next.

Equation (20) gives the key terms in the geometrical constraints as:

$$(a-b) = \lambda \cos\phi_1,\ (b-c) \quad = \lambda \cos(\phi_1 - \frac{2\pi}{3}),\ \text{and}\ (c-a) = \lambda \cos(\phi_1 + \frac{2\pi}{3}). \tag{21}$$

Equation (21) indicates the permissible phase space of ϕ_1 with the property that we exploit here for reversibility, namely, that the three terms $\cos\phi$, $\cos(\phi - \frac{2\pi}{3})$, and $\cos(\phi + \frac{2\pi}{3})$ never carry the same sign, i.e., if one is negative, the other two are non-negative, and if one is positive, the other two are non-positive.

Without loss of generality, let $a^{'}$, $b^{'}$, and $c^{'}$ represent the pre-collision velocities of the left, center and right particles, respectively, colliding along the one-dimensional path. This is illustrated in Figure 4. Let the corresponding post-collision velocities be a, b, and c, respectively. Define a canonical assignation of $\phi_1^{'}$ such that $a^{'} - b^{'} = R\cos\phi_1^{'}$, $b^{'} - c^{'} = R\cos(\phi_1^{'} - \frac{2\pi}{3})$, and $c^{'} - a^{'} = R\cos(\phi_1^{'} + \frac{2\pi}{3})$. Note that this convention fully covers the phase space of all pre-collision velocities for the given total momentum and kinetic energy, and in that sense, is general and equivalent to any other valid convention.

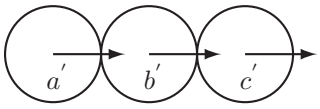

Fig. 4: Canonical configuration of 3 particles in 1 dimension

Pre-collision: For the left and center particles to collide, $a^{'} - b^{'} > 0$. Similarly, for the center and right particles to collide, $b^{'} - c^{'} > 0$. These two conditions constrain the range of $\phi_1^{'}$ to $[\frac{\pi}{6}, \frac{\pi}{2})$, since it is only for that range of $\phi_1^{'}$ that $\cos\phi_1^{'}$ and $\cos(\phi_1^{'} - \frac{2\pi}{3})$ are both non-negative. The exact value of $\phi_1^{'}$ in that range is determined from the following:

$$\bar{a}^{'} = \frac{a^{'} - b^{'}}{\sqrt{2}},\ \bar{b}^{'} = \frac{a^{'} + b^{'}}{\sqrt{2}},\ \text{and}\ \tan\phi_1^{'} = \frac{\sqrt{3}\left(\bar{b}^{'} - \frac{\sqrt{2}}{3}\alpha\right)}{\bar{a}^{'}}. \tag{22}$$

Thus, any pre-collision configuration of ($a^{'}$, $b^{'}$, $c^{'}$) velocities of left, center, and right particles can be uniquely and completely represented by ($\phi_1^{'}$, α, δ), where $\frac{\pi}{6} \le \phi_1^{'} < \frac{\pi}{2}$.

Post-collision: For the left and center particles to diverge after collision (i.e., not to pass through each other), their post-collision velocities must satisfy $a - b < 0$. Similarly, for the center and right particles, $b - c < 0$. These two conditions constrain the range of ϕ_1 to $[\frac{7\pi}{6}, \frac{3\pi}{2})$, which is essentially offset by $+\pi$ from the range of $\phi_1^{'}$.

Pictorially, the regions of interest in the $[0, 2\pi)$ range are given in Figure 5. The full $[0, 2\pi)$ range is divided into six regions, each spanning $\frac{2\pi}{3}$. The first region starts at $\frac{\pi}{6}$. In the figure, $R1$ is the range of $\phi_1^{'}$ (pre-collision), and $S1$ is the range of ϕ_1 (post-collision). Any given angle $\phi_1^{'}$ in $R1$ corresponds to a set of three velocities $\{a^{'}, b^{'}, c^{'}\}$, such that the pre-collision geometrical constraints of Equation (17) are satisfied, implying a specifically ordered sequence of the particles, say, $a; b; c$, along one dimension. The regions $R2$ and $R3$ correspond to left-rotation of the $R1$ sequence, namely, $c; a; b$ and $b; c; a$, obtained by offsetting $\phi_1^{'}$ by $+\frac{2\pi}{3}$ and $-\frac{2\pi}{3}$, respectively. Similarly, the regions $S2$ and $S3$ correspond to the right-rotation of the $S1$ sequence. Note that the pre-collision

and post-collision angles only fall in the $R1$ and $S1$ regions, respectively, and the other regions are unreachable by the system. They are defined and used in intermediate calculations when computing post-collision angles in forward execution (and recovering pre-collision angles in reverse) while ensuring full phase space coverage.

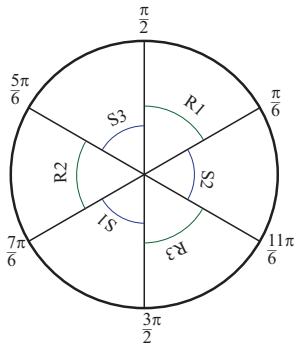

Fig. 5: Division of the angular space for ϕ_1 and $\phi_1^{'}$ in 3-particle collisions in 1 dimension

Reversible Sampling: The problem of making the collision reversible now becomes equivalent to defining a *reversible* (i.e., one-to-one and onto) mapping from any given $\phi_1^{'} \in [\frac{\pi}{6}, \frac{\pi}{2})$ to a random sample of $\phi_1 \in [\frac{7\pi}{6}, \frac{3\pi}{2})$, while accurately preserving the underlying distribution of velocities along the circumference of the ellipse in Equation (18). In general, any such bijection could serve the purpose. Here, such a mapping function is provided in Procedure 3 (forward) and Procedure 4 (reverse).

The forward function essentially rotates $\phi_1^{'}$ uniformly over the entire range $[0, 2\pi)$ first, and then makes adjustments reversibly, as needed, to map back to the valid range $[\frac{7\pi}{6}, \frac{3\pi}{2})$ of ϕ_1.

Procedure 3 ($\phi_1^{'} \longrightarrow \phi_1$): *Forward Function for 3-Particle in 1 Dimension*

$\ell \leftarrow G \cdot L(2\pi)$ {Pick a random arc length on ellipse}
$\psi_1 \leftarrow L^{-1}(\ell)$ {Find angle corresponding to arc length}
{Compute post-collision angle}
$\phi_1 \leftarrow (\phi_1^{'} + \psi_1) \mod 2\pi$ {**Step 1**}
{Adjust post-collision angle if/as necessary}
if more than one of $\cos\phi_1$, $\cos(\phi_1 - \frac{2\pi}{3})$ and $\cos(\phi_1 + \frac{2\pi}{3})$ is positive **then**
  $\phi_1 \leftarrow (\phi_1 + \pi) \mod 2\pi$ {**Step 2**}
end if
if $\cos\phi_1$ is positive **then**
  $\phi_1 \leftarrow (\phi_1 - \frac{2\pi}{3}) \mod 2\pi$ {**Step 3a**}
else if $\cos(\phi_1 - \frac{2\pi}{3})$ is positive **then**
  $\phi_1 \leftarrow (\phi_1 + \frac{2\pi}{3}) \mod 2\pi$ {**Step 3b**}
end if

The operation of Procedure 3 is illustrated in Equation (23). Step 1 adds a random offset in $[0, 2\pi)$ to $\phi_1^{'}$ to get a candidate sample of ϕ_1. Clearly, the candidate may fall in any of the six regions shown in Figure 5. The first correction is to detect if the candidate angle maps to R1, R2, or R3, and if so, remap it to S1, S2, or S3 respectively (because post-collision velocities must diverge, which implies that the post-collision free angle cannot be in R1, R2, or R3). This is accomplished in Step 2 by adding π to the candidate. In Step 3, if the candidate happens to already be in S1, then no additional adjustments are needed. Otherwise, if it falls in S3, it is wrapped back to S1 by rotating it counter-

clockwise by $\frac{2\pi}{3}$ (Step 3a), and if it falls in S2, it is wrapped back to S1 by rotating it clockwise by $\frac{2\pi}{3}$ (Step 3b). To summarize:

$$\phi_1^{'} \in \{R1\} \xrightarrow[\text{randomize}]{\text{Step 1}} \phi_1 \in \{S1, S2, S3, R1, R2, R3\} \xrightarrow[\text{maps to}]{\text{Step 2}} \cdots$$
$$\cdots \xrightarrow[\text{maps to}]{\text{Step 2}} \phi_1 \in \{S1, S2, S3\} \xrightarrow[\text{maps to}]{\text{Step 3}} \phi_1 \in \{S1\} \ . \tag{23}$$

The reverse function shown in Procedure 4 uncovers the random offset first, and reconstructs a candidate $\phi_1^{'}$. It detects adjustments, if any, that were made during forward execution, and then performs the opposite of adjustments to recover the correct original value of $\phi_1^{'}$.

Procedure 4 ($\phi_1 \longrightarrow \phi_1^{'}$): *Reverse Function for 3-Particle in 1 Dimension*

$\ell \leftarrow G \cdot L(2\pi)$ {Recover the random arc length on ellipse}
$\psi_1 \leftarrow L^{-1}(\ell)$ {Recover the angle corresponding to arc length}
{Compute initial guess of pre-collision angle}
$\phi_1^{'} \leftarrow (\phi_1 - \psi_1) \mod 2\pi$
{Correct the guess if/as necessary} {**Step 1**}
if more than one of $\cos\phi_1$, $\cos(\phi_1 - \frac{2\pi}{3})$ and $\cos(\phi_1 + \frac{2\pi}{3})$ is positive **then**
  $\phi_1^{'} \leftarrow (\phi_1^{'} - \pi) \mod 2\pi$ {**Step 2**}
end if
if $\cos\phi_1^{'}$ is negative **then**
  $\phi_1^{'} \leftarrow (\phi_1^{'} - \frac{2\pi}{3}) \mod 2\pi$ {**Step 3a**}
else if $\cos(\phi_1^{'} - \frac{2\pi}{3})$ is negative **then**
  $\phi_1^{'} \leftarrow (\phi_1^{'} + \frac{2\pi}{3}) \mod 2\pi$ {**Step 3b**}
end if

The operation of Procedure 4 is illustrated in Equation (24). Step 1 removes the random offset in $[0, 2\pi)$ from ϕ_1 to get a first guess of the original $\phi_1^{'}$. Clearly, the guessed value may fall in any of the six regions shown in Figure 5. The first correction to the guess is to detect if the candidate angle maps to S1, S2, or S3, and if so, remap it to R1, R2, or R3 respectively (because pre-collision velocities must be converging, which implies that the pre-collision free angle cannot be in S1, S2, or S3). This is accomplished in Step 2 by subtracting π from the guessed value. In Step 3, if the guess happens to already be in R1, then it already represent the correct original value of $\phi_1^{'}$. Otherwise, if it falls in R2, it is wrapped back to R1 by rotating it clockwise by $\frac{2\pi}{3}$ (Step 3a), and if it falls in R3, it is wrapped forward to R1 by rotating it counter-clockwise by $\frac{2\pi}{3}$ (Step 3b). To summarize:

$$\phi_1^{'} \in \{R1\} \xleftarrow[\text{maps to}]{\text{Step 3}} \phi_1^{'} \in \{R1, R2, R3\} \xleftarrow[\text{maps to}]{\text{Step 2}} \cdots$$
$$\cdots \xleftarrow[\text{maps to}]{\text{Step 2}} \phi_1^{'} \in \{S1, S2, S3, R1, R2, R3\} \xleftarrow[\text{de-randomize}]{\text{Step 1}} \phi_1 \in \{S1\} \ . \tag{24}$$

4.4.5. Combining Dynamics and Geometry. The series of transformations performed in forward and reverse collision operations is summarized in Equation (25), which shows the input of the three pre-collision velocities (a', b', c') being transformed by the ***V*-to-Φ** mapping into the triple formed by the angle $\phi_1^{'}$, momentum α, and energy δ. These are fed into the forward parts of the *Function 1* or *Function 2* with an additional input,

which is the uniformly distributed random number G. The resulting triple of the post-collision angle ϕ_1 together with momentum and energy are transformed by the **Φ-to-*V*** mapping into the post-collision velocities (a, b, c). For reversal, the forward process is inverted by first recovering the angle ϕ_1 upon applying the ***V*-to-Φ** mapping on the post-collision velocities. The random number G is recovered by retracing the random number sequence, and the reverse part of the function is applied to recover $\phi_1^{'}$, from which the pre-collision velocities are obtained by applying the **Φ-to-*V*** mapping on the recovered angle.

$$\begin{array}{ccccccc}
(a^{'}, b^{'}, c^{'}) & \xrightarrow{\textbf{\textit{V}-to-}\Phi} & (\phi_1^{'}, \alpha, \delta) & \xrightarrow{\text{Forward}} & (\phi_1, \alpha, \delta) & \xrightarrow{\Phi\textbf{-to-}\textit{V}} & (a, b, c) \\
 & & & \nwarrow & & & \\
 & & & \mathbf{G} & & & \downarrow \\
 & & & \swarrow & & & \\
(a^{'}, b^{'}, c^{'}) & \xleftarrow{\Phi\textbf{-to-}\textit{V}} & (\phi_1^{'}, \alpha, \delta) & \xleftarrow{\text{Reverse}} & (\phi_1, \alpha, \delta) & \xleftarrow{\textbf{\textit{V}-to-}\Phi} & (a, b, c)
\end{array} \tag{25}$$

4.5. Random Number Generation

A subtle but important consideration in zero-memory reversal is the need to ensure that the exact number of random number invocations is also recovered during reverse execution without explicitly storing that information. In other words, the number of random numbers thrown, G_C, for any collision C must be determinable by the collision operator such that the random number sequence is correctly reinstated to the proper position corresponding to the pre-collision state. All our algorithms possess this property, with $G_C = d_n$.

In the case of 2-particle collisions in 1 dimension, no random numbers are needed, trivially satisfying the property. In the case of 2-particle collisions in 2 dimensions, exactly one random number is generated per collision (Procedure 1), and hence the random number stream is stepped back by exactly one number during reversal of that collision (Procedure 2).

The case of 2-particle collisions in 3 dimensions is slightly more complex, since it requires more than one random number to be generated. For forward collision, it appears possible to sometimes generate one random number and some other times two. Only one random number (let us denote it by G_1) appears sufficient to be generated if that number happens to result in $\phi_1 = 0$ or $\phi_1 = \pi$. Similarly, two random numbers (let us denote them by G_1 and G_2) appear needed only to generate ϕ_2 if the randomly generated ϕ_1 (from G_1) is such that $\phi_1 \neq 0$ and $\phi_1 \neq \pi$. However, this conditional generation of one or two random numbers per collision creates difficulties during reversal, because, when the random number stream is reversed and the previous random number G is recovered, we will remain unsure whether G corresponds to G_1 or G_2. It is impossible to disambiguate between the two possibilities because both ϕ_1 and ϕ_2 may assume the value of zero. Hence, we would remain unsure if, in the forward collision, G_1 was zero and hence G_2 was not used for that collision, or if G_1 happened to be non-zero and $G_2 = G$ happened to be zero. Similar ambiguity can be argued for the case of $G \neq 0$. Due to these considerations, we fix the number of random numbers generated per forward collision to be exactly two, unconditionally, so that the random number stream can be reversed exactly by two, restoring it to the correct pre-collision state. If G_1 results in $\phi_1 = 0$, G_2 is still generated from the stream, but simply discarded by the collision algorithm. Assuming that the stream is random, the discarding of G_2 when $\phi_1 = 0$ does not affect the uniformity of the random samples. Note that the discarding is performed *unconditionally*, without any dependence or usage of the actual value of the discarded G_2 in the forward model. This aspect of unconditional discarding is

crucial for reversal, because the reversal also can determinstically reverse the random number stream.

For 3-particle collisions in 1 dimension, exactly one random angle is required for every forward collision, and hence one reversal is necessary and sufficient in each reverse collision, ensuring correct reversal of the random stream. For 3-particle collisions in 2 dimensions (see online supplement appendix), one, two, or three random angles are needed (depending on the geometry and pre-collision velocities) per forward collision. However, due to considerations similar to those for the case of 2-particle collisions in 3 dimensions, we use exactly three random numbers for every forward collision (even if only one or two of them may be sufficient in special cases of dynamics and geometry) in order to reverse the random number stream correctly.

In general, exactly d_n random numbers must be generated for every collision involving n particles in d dimensions.

Another context in which reversibility considerations of random number streams plays an important role is in generating random samples of $\mathbf{\Psi}$. The ***G*-to-$\mathbf{\Psi}$** function for sampling $\mathbf{\Psi}$ involves sampling the circumference of an ellipse (or, in general, sampling the points on the surface of higher dimensional ellipsoids). While rejection-based sampling procedures [Press et al. 2007] are available for such problems, they cannot be used in reversible execution. This is because the number of (uniformly distributed) random numbers used by such rejection-based procedures varies with each sampled point, which makes it impossible to unambiguously reverse the random number stream without keeping track of how many random numbers were generated for each sample. In fact, rejection-based sampling can only be employed for certain special classes of probability distributions, whose parametric input does not vary across samples [Perumalla and Donev 2009], whereas, in sampling $\mathbf{\Psi}$, the parametric input varies with each collision.

5. IMPLEMENTATION RESULTS

In order to test the performance of the algorithms, we implemented the algorithms in software using the C++ programming language, and executed simulation experiments. The experiments are intended to test (1) the ability to restore the initial state after a sequence of many forward collisions followed by their reversals, and (2) the quality of phase space coverage discerned from the uniformity of generated velocity samples. For random number generation, we used a reversible version of a high quality linear congruential generator [L'Ecuyer and Andres 1997] with a period of 2^{121}. A single generator stream is used for all the particles. In each configuration, the experiments verify successful reversal across several thousands of collisions.

5.1. Collision Sequence Reversal

Procedure 5 shows the pseudocode of the experiment program for 2-particle collisions in 2 dimensions. Procedure 6 shows the pseudocode of the experiment program for 3-particle collisions in 1 dimension. In both, the state of the particles is initialized to any desired intial configuration, followed by a sequence of N_c applications of the forward collision operator. $\boldsymbol{RNG}(S)$ represents the generation of the next sample in $[0, 1)$ from the random number stream (which also updates S as side-effect), while $\boldsymbol{RNG}^{-1}(S)$ represents the reversal of the most previous invocation to $\boldsymbol{RNG}$ and the recovery of the most recently generated value from $\boldsymbol{RNG}$. After each forward collision, post-collision velocities are multiplied by -1 to give the new pre-collision velocities for the next collision. Since the post-collision velocities are guaranteed to be divergent, their opposites are guaranteed to give converging velocities that will result in the next collision. After all N_c forward collisions, the entire sequence is reversed by executing the reverse collision operator N_c times. Clearly, the reversal is successful if the final

Procedure 5 *Reversal Illustration for 2-Particle Collisions in 2 Dimensions*

$(a, b, c, d) \leftarrow (a_0, b_0, c_0, d_0)$ {initial velocities}
$\alpha \leftarrow a + b$, $\beta \leftarrow c + d$, $\delta \leftarrow a^2 + b^2 + c^2 + d^2$ {momenta and energy}
$S \leftarrow$ random number seed
$r_{21x} \leftarrow 1$, $r_{21y} \leftarrow 1$ {normalized collision geometry}
for $i = 1$ to N_c **do** {$--$ **Forward Execution** $--$}
  $\phi_1^{'} \leftarrow \boldsymbol{V}$-to-$\boldsymbol{\Phi}(a, b, c, d)$ of Section 4.2
  $G \leftarrow \boldsymbol{RNG}(S)$ {Generate next random number in $[0, 1)$}
  $\phi_1 \leftarrow$ Apply Procedure 1 on $\phi_1^{'}$ using G, r_{21x}, and r_{21y}
  $(a, b, c, d) \leftarrow \boldsymbol{\Phi}$-to-$\boldsymbol{V}(\alpha, \beta, \delta, \phi_1)$ of Section 4.2
  $a \leftarrow -a$, $b \leftarrow -b$, $c \leftarrow -c$, $d \leftarrow -d$ {Reverse velocities to create next collision}
end for
for $i = N_c$ to 1 **do** {$--$ **Reverse Execution** $--$}
  $a \leftarrow -a$, $b \leftarrow -b$, $c \leftarrow -c$, $d \leftarrow -d$
  $\phi_1 \leftarrow \boldsymbol{V}$-to-$\boldsymbol{\Phi}(a, b, c, d)$ of Section 4.2
  $G \leftarrow \boldsymbol{RNG}^{-1}(S)$ {Recover previous random number}
  $\phi_1^{'} \leftarrow$ Apply Procedure 2 on ϕ_1 using G, r_{21x}, and r_{21y}
  $(a, b, c, d) \leftarrow \boldsymbol{\Phi}$-to-$\boldsymbol{V}(\alpha, \beta, \delta, \phi_1^{'})$ of Section 4.2
end for
if $a = a_0$ **and** $b = b_0$ **and** $c = c_0$ **and** $d = d_0$ **then** {$--$ **Verification** $--$}
  print 'Passed'
end if

Procedure 6 *Reversal Illustration for 3-Particle Collisions in 1 Dimension*

$(a, b, c) \leftarrow (a_0, b_0, c_0)$ {initial velocities}
$\alpha \leftarrow a + b + c$, $\delta \leftarrow a^2 + b^2 + c^2$ {momenta and energy}
$S \leftarrow$ random number seed
for $i = 1$ to N_c **do** {$--$ **Forward Execution** $--$}
  $\phi_1^{'} \leftarrow \boldsymbol{V}$-to-$\boldsymbol{\Phi}(a, b, c)$ of Section 4.4
  $G \leftarrow \boldsymbol{RNG}(S)$ {Generate next random number in $[0, 1)$}
  $\phi_1 \leftarrow$ Apply Procedure 3 on $\phi_1^{'}$ using G
  $(a, b, c) \leftarrow \boldsymbol{\Phi}$-to-$\boldsymbol{V}(\alpha, \delta, \phi_1)$ of Section 4.4
  $a \leftarrow -a$, $b \leftarrow -b$, $c \leftarrow -c$ {Reverse velocities to create next collision}
end for
for $i = N_c$ to 1 **do** {$--$ **Reverse Execution** $--$}
  $a \leftarrow -a$, $b \leftarrow -b$, $c \leftarrow -c$
  $\phi_1 \leftarrow \boldsymbol{V}$-to-$\boldsymbol{\Phi}(a, b, c)$ of Section 4.4
  $G \leftarrow \boldsymbol{RNG}^{-1}(S)$ {Recover previous random number}
  $\phi_1^{'} \leftarrow$ Apply Procedure 4 on ϕ_1 using G
  $(a, b, c) \leftarrow \boldsymbol{\Phi}$-to-$\boldsymbol{V}(\alpha, \delta, \phi_1^{'})$ of Section 4.4
end for
if $a = a_0$ **and** $b = b_0$ **and** $c = c_0$ **then** {$--$ **Verification** $--$}
  print 'Passed'
end if

velocities after all N_c reversals recovers the initial velocities. This condition is verified at the end and the corresponding status message is printed.

All executions terminated successfully with a "passed" status, verifying the restoration of initial state after reversal of N_c collisions. The unbiased and correct phase space

coverage is tested with N_c up to 10^6 by plotting the post-collision angles against random angle offsets and pre-collision velocities.

In the case of 3-particle collisions in 1 dimension, numerical integration to compute the segment length of an ellipse was performed using the Simpson's rule. Also, care was needed to account for numerical precision issues when using the numerically computed cosine function, which sometimes produces numerical noise for angles within very small neighborhoods of multiples of $\frac{\pi}{6}$ and $\frac{\pi}{2}$. A tolerance of $\pm 10^{-8}$ around zero was employed when determining whether any given cosine value can be considered zero or positive.

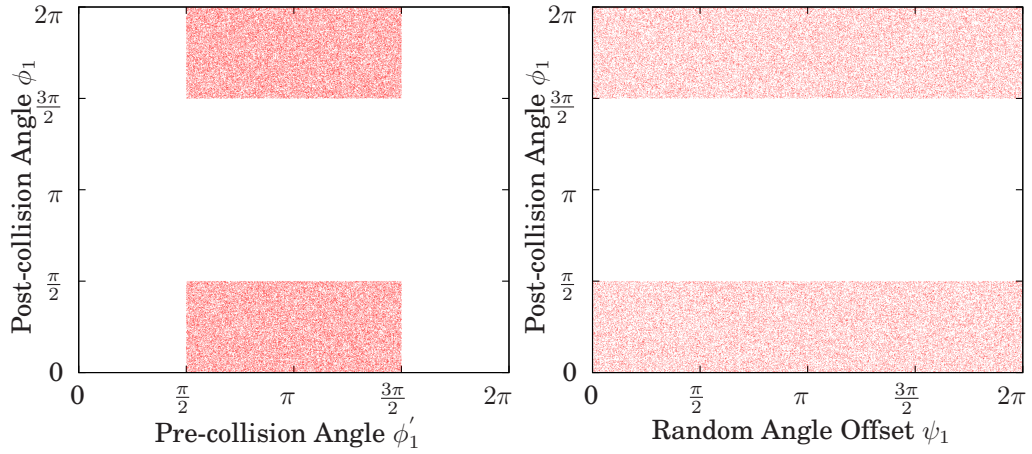

Fig. 6: Sampling for $N_c = 10^5$ collisions in 2-particle collisions in 2 dimensions

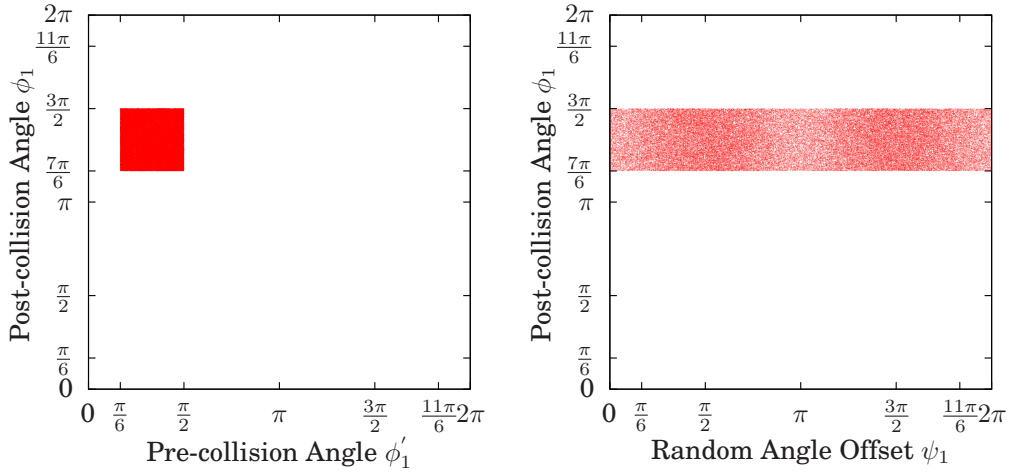

Fig. 7: Sampling for $N_c = 10^5$ collisions in 3-particle collisions in 1 dimension

Figure 6 plots the post-collision angles against their corresponding pre-collision angles generated as part of an execution of 2-particle collisions in 2 dimensions, and plots the post-collision angles against their corresponding random angles in the same execution. Both plots show excellent uniformity and correctness of the sampled phase space. Similarly, Figure 7 shows the corresponding data for 3-particle collisions in 1

dimension; again, they display excellent coverage. Since the covered areas in the plots become too dense for visual clarity when N_c becomes large, only data for $N_c = 10^5$ are shown in the figures.

5.2. Randomness Tests

The correctness of phase space coverage is experimentally verified by testing the randomness of the generated angles using statistical tests. The Diehard Battery of Tests [Marsaglia 1995] for statistical verification of randomness is used for this purpose.

The post-collision angles are mapped to double-precision numbers, each in $[0, 1)$, and the uniformity of their distribution is verified. This method is applied to 2-particle collisions in 2 dimensions, and to 3-particle collisions in 1 dimension.

For 2-particle collisions in 2 dimensions, each post-collision angle ϕ_1 is mapped to a double-precision number $\eta \in [0, 1)$ as $\eta = (\phi_1 - \kappa)/\pi$, where $\kappa = 0$ if $0 \leq \phi_1 \leq \frac{\pi}{2}$, and $\kappa = \pi$ otherwise ($\frac{3\pi}{2} \leq \phi_1 \leq 2\pi$). Each η is converted into an integer equal to $\lfloor \eta \times 4,294,967,296 \rfloor$, and the resulting series of numbers (in hexadecimal format) is converted via the `asc2bin` program of Diehard to a binary-formatted file given as input to the `diehard` program.

For 3-particle collisions in 1 dimension, each post-collision angle ϕ_1 is mapped to a double-precision number $\eta \in [0, 1)$ as $\eta = (\phi_1 - \frac{7\pi}{6})/(\frac{3\pi}{2} - \frac{7\pi}{6})$. Each η is converted into an integer equal to $\lfloor \eta \times 4,294,967,296 \rfloor$, and the resulting series of numbers (in hexadecimal format) is converted via the `asc2bin` program of Diehard to a binary-formatted file given as input to the `diehard` program.

Both Procedure 5 and Procedure 6 were successfully executed with $N_c = 3,000,000$ collisions, such that they terminate with a "Passed" result. The angles are logged to a file during their execution, and then used as input to the randomness tests. According to the Diehard tests, if the "p-values" computed and printed by `diehard` are observed to be strictly greater than 0 and less than unity, randomness is understood to be satisfied [Marsaglia 1995]. The p-values observed from the randomness tests on the generated angles are given in Table I. Good phase space coverage via randomization is indicated by the fact that the p-values from several tests are significantly away from zero and unity.

For 2-particle collisions in 2 dimensions, all tests in the Diehard repository were used and verified to generate very good randomness (positive p-values less than unity) without exception. This is due to the fact that no numerical precision effects are present in the collision algorithm for 2-particle collisions in 2 dimensions. For 3-particle collisions in 1 dimension, all tests were used except those that operate on bit-level representations (such as the Birthday, Bitstream and Count-the-1s tests). This is because round off and trunction effects in numerical integration, even at a relatively high precision of 10^{-9} in the computation of angles, introduces non-random patterns in the last few bits of the mantissa, appearing as non-randomness when selectively viewed in isolation or across multiple floating point numbers. However, when the angles are viewed as numbers themselves, uniform randomness is indeed observed, as expected.

6. PERFORMANCE ESTIMATION

To estimate the performance gain that can be expected by resorting to reversible collisions instead of state saving, we implemented a simulation of a sequence of 2-particle collisions in a closed system of N particles in $d = 2$ dimensions. Experiments were run with $N = 1,000$, $N = 10,000$, and $N = 100,000$ particles. Starting with random initial configuration of positions and velocities, and randomly selected inter-collision times, particle motion is simulated between collisions, and, at every collision point, the collision operator is applied on a random pair of particles. With state saving, the system

Table I: Randomness indicator p-values from Diehard battery of tests

Test	**Category**	p**-value**	
		2-Dimension 2-Particle	1-Dimension 3-Particle
CRAPS	Overall	0.979800	0.570270
	Wins	0.790715	0.862293
	Throws/game	0.979797	0.570273
RUNS	Set 1 - Up	0.847070	0.298998
	Set 1 - Down	0.128221	0.632054
	Set 2 - Up	0.183069	0.871807
	Set 2 - Down	0.244909	0.111657
SUMS	10 χ^2-tests on 100 χ^2-tests	0.370145	0.257959
SQUEEZE	42 degrees of freedom	0.890107	0.435410
3DSPHERES	χ^2-test on 20 p-values	0.813199	0.897844
MINDIST	χ^2-test on 100 min-distances	0.348209	0.751603
PARKLOT	χ^2-test on 10 p-values	0.947200	0.720233

state is saved to memory before every collision, to be able to roll back to that state. With reverse computation, no state is saved, as the system can be rolled back perfectly to any point in the past by reverse computation alone, without reliance on memory.

Two platforms with different computational and memory characteristics are tested: one with traditional central processing unit (CPU), and the other with newer graphical processing unit (GPU). Modern CPUs now have much higher computational speeds than memory speeds; the differential between computational and memory speeds is even more pronounced in modern GPU platforms [Pharr et al. 2005]. Implementation on the CPU is realized in the C++ programming language, and that on the GPU is in the CUDA programming language [Sanders and Kandrot 2010]. The CPU is an AMD Opteron 6174 processor with 64 GB of memory. The GPU is a high-end nVidia Geforce GTX 580 (Fermi) accelerator with 512 CUDA cores and 3 GB device memory. Compilation systems used were gcc 4.4.5 and CUDA 4.1.

In each simulation run, $N_c = 1000$ collisions were simulated, and, after N_c collisions, the system was rolled back to the beginning. With reverse computation, the positions and velocities are verified to match the initial conditions exactly (to within at least $\epsilon = \pm 10^{-9}$), while, with state saving, the results are trivially exactly matched.

The results are drawn as stacked histograms in Figure 8, with the total height of each bar representing the total time for forward execution of N_c collisions and their reversal, which is split into *Forward Time* and *Rollback Time* in milliseconds. Three variants are benchmarked: **SS1** represents the state saving mechanism in which all state is saved (positions and velocities); **SS2** represents an optimized state saving mechanism in which the positions are saved and only the four components of the pre-collision velocities of the colliding pair are saved; **RC** represents the reverse computation with no memory. For each variant, the suffix **-CPU** represents execution on the CPU, and **-GPU** represents execution on the GPU.

Total run time with reverse computation is lower across the board. As expected, the greatest differential is observed in Figure 8(f) for the GPU runs with the largest number of particles ($N = 100,000$). Since the ratio of memory transfer cost to computational cost is much higher with the GPU, reverse computation runs much faster, while state saving incurs the high memory transfer cost for every collision. Also, the GPU platform is known to be extremely efficient with large vectorized codes such as this simulation (i.e., more particles can be simulated with little increase in total time, if memory bottle neck is relieved). Hence, the reverse computation runs are extremely fast on the GPU,

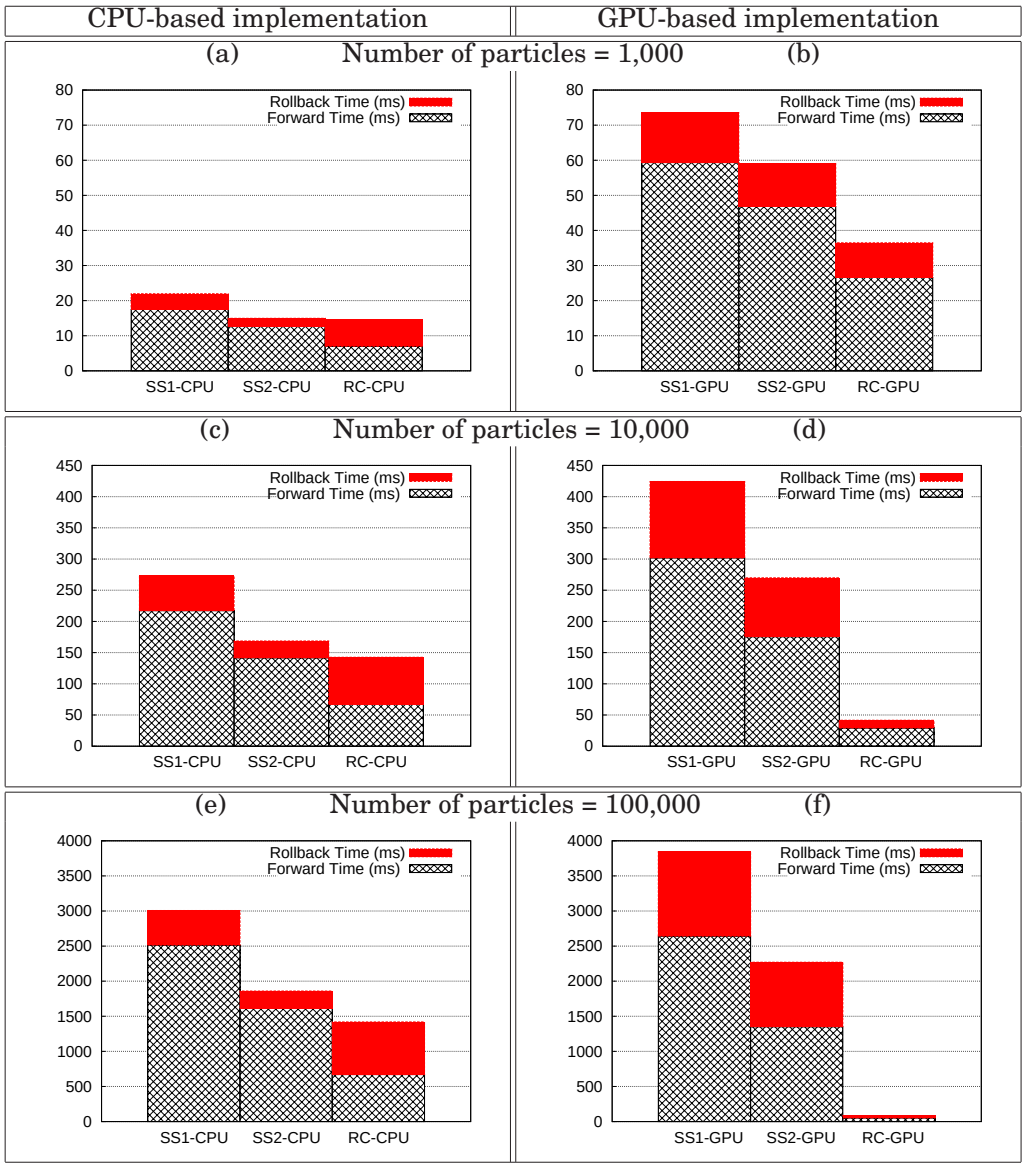

Fig. 8: Performance of state saving and reverse computation with $N_c = 1000$ collisions

compared to all CPU runs and also compared to state saving with GPU. On smaller number of particles ($N = 1000$ and $N = 10,000$), CPU runs in (a) and (c) are faster than GPU runs in (b) and (d) because of larger CPU caches, yet, even in this case of relatively lower memory cost, reverse computation is observed to run faster. When the number of particles is further increased (e.g., $N \geq 1$ million), state saving becomes infeasible due to memory limitations, but reverse computation runs well even at such large scale. Similarly, the benefit of using reversible simulation only increases with the number of collisions, due to corresponding increase in the memory needs of state saving.

7. SUMMARY

The classical problem of simulating elastic collisions of hard spheres has been revisited, with the important additional requirement of reversibility. We formalized the problem in terms of accurate phase space coverage specification and geometrical constraints. We solved the problem by developing a general framework that combines reversible pseudo random number generation with new mapping functions, geometrical constraints, and reversal semantics. While previous log-based approaches require memory proportional to the number of collisions, our algorithms incur essentially zero memory overheads and also ensure correct phase space coverage. We developed the detailed steps for 2-particle collisions (up to 3 dimensions) and 3-particle collisions (up to 2 dimensions). In these configurations, memory overhead is exactly zero for collisions in which $d_n = 1$, and essentially zero for collisions with $d_n > 1$. In the latter configurations, $\{\phi'_{i+1}, \ldots, \phi'_{d_n}\}$ are logged if and only if $\phi_i = 0$, for any $1 \leq i < d_n$. Generalizations to collisions among larger number of particles and at higher dimensions are tedious, but can be carried out if needed. At higher dimensions and with larger number of particles, computationally expensive numerical integration becomes necessary in both classical (forward-only) approaches as well as in the *forward* procedures of our reversible method. To meet the goal of minimal memory overhead, our *reverse* procedures rely on numerical integration as well, whereas log-based reversal approaches would use memory to save information and avoid recomputation in the reverse path. In a normal, well-balanced parallel execution, reversals of collisions are far fewer than forward collisions (i.e., dependency violations are incurred infrequently). In such cases, our reversible models are more efficient, since forward cost for reversibility is eliminated, whereas log-based approaches would incur log-generation costs in forward execution.

Finally, although reversibility of elastic collisions is a seemingly simple problem to formulate, it includes sufficient complexity to make it challenging. The new approach and results presented here offer insights in revisiting additional classical physical system models with the added requirement of reversibility and exploring their fundamental memory characteristics and limits.

ACKNOWLEDGMENTS

The authors thank Alfred J. Park, Sudip K. Seal, and James J. Nutaro for constructive comments on early versions of the manuscript, and Alfred J. Park for help with the implementation for performance estimation.

REFERENCES

Bitan Biswas and Rajiv Mall. 1999. Reverse Execution of Programs. *ACM SIGPLAN Notices* 34, 4 (April 1999), 61–69.

James E. Broadwell. 2001. Irreversibility in a Reversible Lattice Gas. *Journal of Statistical Physics* 103, 5-6 (2001), 1125–1136.

Christopher Carothers, Kalyan S. Perumalla, and Richard M. Fujimoto. 1999. Efficient Optimistic Parallel Simulations using Reverse Computation. *ACM Transactions on Modeling and Computer Simulation* 9, 3 (1999), 224–253.

P. W. Cleary and M. L. Sawley. 2002. DEM Modeling of Industrial Granular Flows: 3D Case Studies and the Effect of Particle Shape on Hopper Discharge. *Applied Mathematical Modelling* 26, 2 (2002).

Aleksandar Donev. 2009. Asynchronous Event-Driven Particle Algorithms. *Simulation* 85 (April 2009), 229–242. Issue 4.

Richard M. Fujimoto. 1990. Optimistic Approaches to Parallel Discrete Event Simulation. *Transactions of the Society for Computer Simulation* 7, 2 (1990), 153–191. june 1990.

James M. Haile. 1992. *Molecular Dynamics Simulation: Elementary Methods*. John Wiley & Sons, Inc.

P. Hontalas, B. Beckman, M. DiLorento, L. Blume, P. Reiher, K. Sturdevant, L. V. Warren, J. Wedel, F. Wieland, and D. R. Jefferson. 1989. Performance of the Colliding Pucks Simulation on the Time Warp Operating System. In *Distributed Simulation*.

David R. Jefferson. 1985. Virtual Time. *ACM Trans. on Programming Languages and Systems* 7, 3 (1985), 404–425.

David R. Jefferson, B. Beckman, F. Wieland, L. Blume, M. DiLorento, P. Hontalas, P. Reiher, K. Sturdevant, J. Tupman, J. Wedel, and H. Younger. 1987. The Time Warp Operating Systems. In *Symposium on Operating Systems Principles*, Vol. 21. 77–93.

Alan T. Krantz. 1996. Analysis of an efficient algorithm for the hard-sphere problem. *ACM Trans. Model. Comput. Simul.* 6, 3 (July 1996), 185–209.

Pierre L'Ecuyer and Terry H. Andres. 1997. A Random Number Generator Based on the Combination of Four LCGs. In *Mathematics and Computers in Simulation*. 99–107.

Harvey Leff and Andrew F. Rex. 2002. *Maxwell's Demon 2: Entropy, Classical and Quantum Information, Computing*. Taylor and Francis. 502 pages.

Boris D. Lubachevsky. 1991. How to Simulate Billiards and Similar Systems. *J. Comput. Phys.* 92, 2 (1991), 255–283. Updated version (2006) at arXiv.org as arXiv:cond-mat/0503627v2.

D. Manivannan and Mukesh Singhal. 1996. A Low-Overhead Recovery Technique Using Quasi-Synchronous Checkpointing. In *Proc. IEEE Int. Conference on Distributed Computing Systems*. 100–107.

Mauricio Marin. 1997a. Billiards and Related Systems on the Bulk-Synchronous Parallel Model. In *11th Workshop on Parallel and Distributed Simulation*. 164–171.

Mauricio Marin. 1997b. Event-Driven Hard-Particle Molecular Dynamics using Bulk-Synchronous Parallelism. *Computer Physics Communications* 102, 1-3 (1997), 81–96.

George Marsaglia. 1972. Choosing a Point from Surface of a Sphere. *Annals of Math. Statistics* 43, 2 (1972), 645–646.

George Marsaglia. 1995. Diehard Battery of Tests of Randomness. (1995). stat.fsu.edu/pub/diehard.

S. Miller and Stefan Luding. 2004. Event-driven Molecular Dynamics in Parallel. *J. Comput. Phys.* 193, 1 (2004), 306–316.

Kalyan S. Perumalla. 2007. Scaling Time Warp-based Discrete Event Execution to 10^4 Processors on the Blue Gene Supercomputer. In *International Conference on Computing Frontiers*. Ischia, Italy, 69–76.

Kalyan S. Perumalla and Aleksandar Donev. 2009. *Perfect Reversal of Rejection Sampling Methods for First-Passage-Time and Similar Distributions*. Technical Report TM-2009/182. Oak Ridge National Laboratory.

Matt Pharr, Randima Fernando, and Tim Sweeney. 2005. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press.

Brian J. Ross. 1997. Running Programs Backwards: the Logical Inversion of Imperative Computation. *Journal of Formal Aspects of Computing* 9, 3 (1997), 331–348.

Jason Sanders and Edwards Kandrot. 2010. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional.

Simo Siiria and Jouko Yliruusi. 2007. Particle packing simulations based on Newtonian mechanics. *Powder Technology* 174, 3 (2007), 82 – 92.

Clifford Truesdell and R. G. Muncaster. 1980. *Fundamentals of Maxwell's Kinetic Theory of a Simple Monatomic Gas*. Academic Press.

Dawei Zhao, Erfan G. Nezami, Youssef M.A. Hashash, and Jamshid Ghaboussi. 2006. Three-dimensional discrete element simulation for granular materials. *Engineering Computations* 23, 7 (2006), 749 – 770.

H.P. Zhu, Z.Y. Zhou, R.Y. Yang, and A.B. Yu. 2007. Discrete particle simulation of particulate systems: Theoretical developments. *Chemical Engineering Science* 62, 13 (2007), 3378 – 3396.

Received February 2012; revised December 2012; accepted January 2013

Online Appendix to: Reversible Simulations of Elastic Collisions

KALYAN S. PERUMALLA, Oak Ridge National Laboratory
VLADIMIR A. PROTOPOPESCU, Oak Ridge National Laboratory

A. REVERSAL FOR 3-PARTICLE COLLISIONS IN 2 DIMENSIONS

In a collision of three particles in 2-dimensional space, the equations of motion are:

$$\left.\begin{aligned} a+b+c &= \alpha \\ d+e+f &= \beta \\ a^2+b^2+c^2+d^2+e^2+f^2 &= \delta \\ 3\delta &> \alpha^2+\beta^2 \end{aligned}\right\} \text{Dynamics.} \tag{1}$$

The geometrical constraints satisfied by the post-collision velocities are:

$$\left.\begin{aligned} r_{21x}\cdot(a-b)+r_{21y}\cdot(d-e) &> 0, \quad \text{if } \mathbf{P1} \text{ and } \mathbf{P2} \text{ are in contact} \\ r_{32x}\cdot(b-c)+r_{32y}\cdot(e-f) &> 0, \quad \text{if } \mathbf{P2} \text{ and } \mathbf{P3} \text{ are in contact} \\ r_{13x}\cdot(c-a)+r_{13y}\cdot(f-d) &> 0 \quad \text{if } \mathbf{P3} \text{ and } \mathbf{P1} \text{ are in contact} \end{aligned}\right\} \text{Geometry.} \tag{2}$$

The pre-collision geometrical constraints are obtained by replacing > 0 by < 0 in the post-collision constraints. In Equation (2), we will use **K1** to denote the first inequality (**P1** and **P2** in contact), **K2** to denote the second inequality, and **K3** to denote the third.

The solution to the equations of motion satisfies the hyper-ellipsoid equation:

$$\left(\frac{\bar{a}-0}{1}\right)^2+\left(\frac{\bar{b}-\frac{\sqrt{2}}{3}\alpha}{\frac{1}{\sqrt{3}}}\right)^2+\left(\frac{\bar{d}-0}{1}\right)^2+\left(\frac{\bar{e}-\frac{\sqrt{2}}{3}\beta}{\frac{1}{\sqrt{3}}}\right)^2=\delta-\frac{\alpha^2+\beta^2}{3}, \tag{3}$$

where $\bar{a}=\dfrac{a-b}{\sqrt{2}}$, $\bar{b}=\dfrac{a+b}{\sqrt{2}}$, $\bar{d}=\dfrac{d-e}{\sqrt{2}}$, and $\bar{e}=\dfrac{d+e}{\sqrt{2}}$.

The hyper-ellipsoid can be described via parametic equations with three independent parameters $\{\phi_1,\phi_2,\phi_3\}$ as the degrees of freedom as follows:

$$\left.\begin{aligned} \bar{a} &= \frac{\lambda}{\sqrt{2}}\cos\phi_1, \\ \bar{d} &= \frac{\lambda}{\sqrt{2}}\sin\phi_1\cos\phi_2, \\ \bar{b} &= \frac{\sqrt{2}}{3}\alpha+\frac{\lambda}{\sqrt{2}\sqrt{3}}\sin\phi_1\sin\phi_2\cos\phi_3, \\ \bar{e} &= \frac{\sqrt{2}}{3}\beta+\frac{\lambda}{\sqrt{2}\sqrt{3}}\sin\phi_1\sin\phi_2\sin\phi_3, \text{ where} \\ \lambda &= \sqrt{2}\sqrt{\delta-\frac{\alpha^2+\beta^2}{3}}, \end{aligned}\right\} \quad \begin{aligned} &\phi_1\in[0,\pi], \\ &\phi_2\in[0,\pi], \\ &\phi_3\in[0,2\pi). \end{aligned} \tag{4}$$

Based on the preceding parametric equations, the terms in the geometrical constraints can be expressed as:

$$
\begin{aligned}
a - b &= \lambda \cos\phi_1 \\
d - e &= \lambda \sin\phi_1 \cos\phi_2 \\
b - c &= \frac{\lambda}{2}(\sqrt{3}\sin\phi_1 \sin\phi_2 \cos\phi_3 - \cos\phi_1) \\
e - f &= \frac{\lambda}{2}(\sqrt{3}\sin\phi_1 \sin\phi_2 \sin\phi_3 - \sin\phi_1 \cos\phi_2) \\
c - a &= \frac{-\lambda}{2}(\sqrt{3}\sin\phi_1 \sin\phi_2 \cos\phi_3 + \cos\phi_1) \\
f - d &= \frac{-\lambda}{2}(\sqrt{3}\sin\phi_1 \sin\phi_2 \sin\phi_3 + \sin\phi_1 \cos\phi_2)\ .
\end{aligned}
\tag{5}
$$

Note that $\sin\phi_1$ and $\sin\phi_2$ are always non-negative.
The V**-to-**Φ and Φ**-to-**V mappings are obtained from Equation (5) as follows:
V**-to-**Φ: Given $a - f$, the angles ϕ_1, ϕ_2, and ϕ_3 are computed as:

$$
\begin{aligned}
\phi_1 &= \cos^{-1}\left(\frac{a-b}{\lambda}\right) \\
\phi_2 &= \begin{cases} 0 \text{ if } \phi_1 = 0 \text{ or } \phi_1 = \pi \\ \cos^{-1}\left(\dfrac{d-e}{\lambda \sin\phi_1}\right) \text{ otherwise} \end{cases} \\
\phi_3 &= \begin{cases} 0 \text{ if } \phi_1 = 0 \text{ or } \phi_1 = \pi \text{ or } \phi_2 = 0 \text{ or } \phi_2 = \pi \\ \sin^{-1}\left(\dfrac{2(e-f) + \lambda \sin\phi_1 \cos\phi_2}{\sqrt{3}\lambda \sin\phi_1 \sin\phi_2}\right) \text{ otherwise} \end{cases}
\end{aligned}
\tag{6}
$$

Φ**-to-**V: Given $\alpha, \beta, \delta, \phi_1, \phi_2, \phi_3$, the values of $a - f$ are computed from Equation (5). Note that the cases of $\phi_1 = 0$ or $\phi_1 = \pi$ or $\phi_2 = 0$ or $\phi_2 = \pi$ are solved as follows:

$$
\begin{aligned}
a &= \frac{\alpha}{3} + \frac{\lambda}{2}\cos\phi_1 \\
b &= \frac{\alpha}{3} - \frac{\lambda}{2}\cos\phi_1 \\
c &= \frac{\alpha}{3} \\
d &= \frac{\beta}{3} + \frac{\lambda}{2}\sin\phi_1 \cos\phi_2 \\
e &= \frac{\beta}{3} - \frac{\lambda}{2}\sin\phi_1 \cos\phi_2 \\
f &= \frac{\beta}{3}
\end{aligned}
\tag{7}
$$

Hence, when dealing with Equation (4) in the remaining of the analysis, we only consider the case of $0 < \phi_1 < \pi$ and $0 < \phi_2 < \pi$. Also, whenever $\phi_1 = 0$ or $\phi_1 = \pi$, we set $\phi_2 = 0$ and $\phi_3 = 0$. Similarly, whenever $\phi_2 = 0$ or $\phi_2 = \pi$, we set $\phi_3 = 0$.

A.1. Possible Geometries

To make analysis easier, a notion of a "canonical configuration" is introduced for the three colliding particles in the 2-dimensional space. The canonical view is to align the x-axis with the line joining the centers of two particles in contact. The choice of the pair chosen for this line is designed to be recoverable in reverse execution, essentially

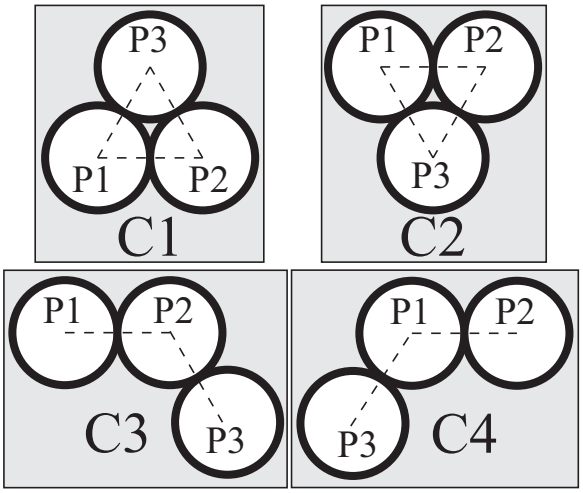

Fig. 1: The canonical forms that we define for the four possible configurations in which three particles may undergo collision in 2 dimensions

making the choice of the pair only dependent on the geometry of collision, independent of the velocities of the particles undergoing the 3-particle collision.

When the x-axis is aligned along such a pair of particles in contact, they can appear in one of four configurations shown in Figure 1. In all configurations, the horizontal line is chosen to be the line joining the two particles whose identifiers are smaller than that of the third one. The particle with the smallest identifier is always chosen as the particle on the left of the horizontal axis line.

In configurations **C1** and **C2**, all three particles are in contact with each other, giving three pairs of particles in contact. In each of the rest, **C3** and **C4**, only two pairs of particles are in contact. In all configurations, **P1** and **P2** are the left and right particles forming the horizontal axis. In **C1**, the third particles **P3** is above the two horizontally placed particles, while, in **C2**, the third particle is below them. The configurations **C3** and **C4** cover the rest of the possibilities in which only two pairs of the particles are in contact with each other at the same time. In **C3**, the third particle **P3** is only in contact with particle **P2**, while, in **C4**, **P3** is only in contact with particle **P1**.

Once the canonical configuration is chosen, all the velocities are rotated to reorient to the new axes. Total momenta and energy undergo a resultant change that can be reversed to recover original momenta and energy values simply by rotating the axes back to original axes. Since the original velocities are in a one-to-one relation with the transformed velocities, it is the transformed velocities that will be considered as velocities $a..f$ defined earlier for the 3-particle, 2-dimension collision model. After computing individual post-collision velocities, they are rotated back to the original frame of reference, thus restoring the original total momenta and energy.

A.2. Sampling the Ψ Phase Space

To generate a random set of offsets, $\mathbf{\Psi}$, a random sample point on the hyper-ellipsoid of Equation (3) is generated by invoking the numerical method given in Procedure 1 with $s = 4$, and $\{{}_4\lambda_i \mid 1 \leq i \leq 4\}$ obtained by converting Equation (3) into the canonical form expected by the algorithm. The sampling approach is a generalized version of the approach employed for 3-particle collisions in 1 dimension (Section 4.4) to sample the phase space of $\mathbf{\Psi}$. As mentioned earlier, although rejection-based methods are avail-

able for generating the samples, rejection is incompatible with reversibility, making them inapplicable here.

Procedure 1 ($\boldsymbol{G \longrightarrow \Psi}$): *Generate the Parameters $\boldsymbol{\Psi}$ of a Random Point on the Surface of an s-Dimensional Hyper-Ellipsoid, $\mathcal{H}_s$, using Random Numbers $\boldsymbol{G} = \{G_1, \ldots, G_{s-1}\}$*

1: **Input**: $s, \{{}_s\lambda_i \mid 1 \leq i \leq s\}$, where integer $s > 1$, and $\sum_{i=1}^{s} \left(\frac{x_i}{{}_s\lambda_i}\right)^2 = 1$ is the hyper-ellipsoid
2: **Output**: $\{\psi_i \mid 1 \leq i < s\}$, where ψ_i are the parameters of a random point $({}_rx_1, \ldots, {}_rx_s)$ on the hyper-ellipsoid, such that ${}_rx_i = {}_s\lambda_i \cos\psi_i \prod_{j=1}^{i-1} \sin\psi_j$ for all $1 \leq i < s$, and ${}_rx_s = {}_s\lambda_s \prod_{j=1}^{s} \sin\psi_j$
3: **for** $k = s$ **down-to** 2 **do**
4: Let $\overline{k} = s - k + 1$
5: **if** $k = 2$ **then**
6: Invoke the algorithm in Section 4.4.3 using $G_{\overline{k}}$ to generate the parameter $0 \leq \psi^* < 2\pi$ corresponding to a random point on the perimeter of the (2-dimensional) ellipse $\mathcal{H}_2$
7: $\psi_{\overline{k}} \leftarrow \psi^*$
8: **else**
9: Compute the surface area A_k of $\mathcal{H}_k$
10: Compute the random fraction ${}_rA_k$ of A_k as: ${}_rA_k \leftarrow G_{\overline{k}} \cdot A_k$
11: Using the bisection method analogous to that in Section 4.4.3, determine a $0 \leq \psi^* \leq \pi$ such that surface area of the calotte of $\mathcal{H}_k$ defined by the latitudinal angle ψ^* from the pole equals ${}_rA_k$
12: $\psi_{\overline{k}} \leftarrow \psi^*$
13: Determine the parameters $\{{}_{s-1}\lambda_i \mid 1 \leq i \leq s-1\}$ of the $(s-1)$-dimensional ellipsoid $\mathcal{H}_{s-1}$ formed by the opening of the calotte, obtained by substituting ${}_sx_{\overline{k}} = {}_s\lambda_{\overline{k}} \cos\psi_{\overline{k}}$.
14: **end if**
15: **end for**

Note that, by using one extra random number (i.e., using $d_n + 1$ random numbers instead of d_n), Procedure 1 can be elegantly generalized, avoiding reliance on the separate algorithm of Section 4.4.3 for the special case of a 2-dimensional ellipsoid (ellipse). This can be achieved by iterating down to $s = 1$, introducing an extra parameter ψ_s, and using the additional random number to set ψ_s to either $\frac{\pi}{2}$ or $\frac{3\pi}{2}$ (*i.e.*, ${}_1x_1 = \pm_1\lambda_1$). Using such a generalized algorithm for sampling the surface of an s-dimensional ellipsoid, $s \geq 1$, the algorithm in Section 4.4.3 can be replaced by a call to the generalized Procedure 1 with $s = 2$. However, to restrict the number of random numbers to d_n, we customize the last iteration to use the optimized version given in Section 4.4.3.

In the next sections, we derive the permissible ranges of the parameters, restricted from their full, nominal ranges due to conservation laws as well as geometrical constraints, and develop the forward and reverse collision procedures based on the derived parameter ranges.

A.3. Configuration C1

From the geometry of **C1** (ref. Figure 2), it can be seen that $r_{21x} = D$, $r_{21y} = 0$, $r_{32x} = \frac{-D}{2}$, $r_{32y} = \frac{\sqrt{3}}{2}D$, $r_{13x} = \frac{-D}{2}$, and $r_{13y} = \frac{-\sqrt{3}}{2}D$. Using these in Equation (2) and Equation (5) to account for the geometrical constraints, we obtain the ranges of ϕ_1 and ϕ_2 that are more constrained than in Equation (4).

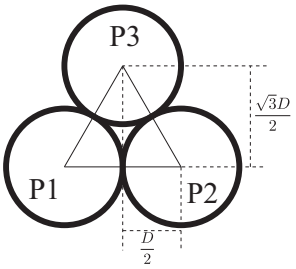

Fig. 2: Measures of interest in configuration **C1**

The inequality **K1** directly gives the following:

$$\begin{aligned} D \cdot (\lambda \cos\phi_1) + 0 \cdot (\cdots) &> 0 \\ \implies \cos\phi_1 &\geq 0 \\ \implies -\tfrac{\pi}{2} \leq \phi_1 &\leq \tfrac{\pi}{2} \\ \implies 0 \leq \phi_1 < \tfrac{\pi}{2} &\text{ from Equation (4).} \end{aligned} \tag{8}$$

Inequality **K2** gives:

$$\begin{aligned} \tfrac{-D}{2} \cdot \Big(\tfrac{\lambda}{2}(\sqrt{3}\sin\phi_1 \sin\phi_2 \cos\phi_3 - \cos\phi_1)\Big) + \\ \tfrac{\sqrt{3}D}{2} \cdot \Big(\tfrac{\lambda}{2}(\sqrt{3}\sin\phi_1 \sin\phi_2 \sin\phi_3 - \sin\phi_1 \cos\phi_2)\Big) > 0. \end{aligned} \tag{9}$$

or, $-\sqrt{3}\sin\phi_1 \sin\phi_2 \cos\phi_3 + \cos\phi_1 + 3\sin\phi_1 \sin\phi_2 \sin\phi_3 - \sqrt{3}\sin\phi_1 \cos\phi_2 > 0.$

Inequality **K3** gives:

$$\begin{aligned} \tfrac{-D}{2} \cdot \Big(\tfrac{-\lambda}{2}(\sqrt{3}\sin\phi_1 \sin\phi_2 \cos\phi_3 + \cos\phi_1)\Big) + \\ \tfrac{-\sqrt{3}D}{2} \cdot \Big(\tfrac{-\lambda}{2}(\sqrt{3}\sin\phi_1 \sin\phi_2 \sin\phi_3 + \sin\phi_1 \cos\phi_2)\Big) > 0. \end{aligned} \tag{10}$$

or, $\sqrt{3}\sin\phi_1 \sin\phi_2 \cos\phi_3 + \cos\phi_1 + 3\sin\phi_1 \sin\phi_2 \sin\phi_3 + \sqrt{3}\sin\phi_1 \cos\phi_2 > 0.$

From **K2**, we get:

$$\textbf{L1}: \quad \cos(\phi_3 + \tfrac{\pi}{3}) < \tfrac{1}{2\sqrt{3}\tan\phi_1 \sin\phi_2} - \tfrac{1}{2\tan\phi_2}. \tag{11}$$

Similarly, from **K3**, we get:

$$\textbf{L2}: \quad -\cos(\phi_3 - \tfrac{\pi}{3}) < \tfrac{1}{2\sqrt{3}\tan\phi_1 \sin\phi_2} + \tfrac{1}{2\tan\phi_2}. \tag{12}$$

To ensure a valid range for the left hand side in **L1**, the right hand side (RHS) of the same must not be less than -1. Setting the RHS to -1 defines the boundary between the possible and impossible regions, in terms of the relation between ϕ_1 and ϕ_2. Similar restrictions arise from **L2**. These considerations give the limits on ϕ_1 and ϕ_2 as follows:

$$\begin{aligned} \phi_1 = &\quad \cot^{-1}(\sqrt{3}\cos\phi_2 - 2\sqrt{3}\sin\phi_2) \text{ (from } \textbf{L1}\text{, RHS=}-1) \\ \phi_1 = &\quad \cot^{-1}(-\sqrt{3}\cos\phi_2 - 2\sqrt{3}\sin\phi_2) \text{ (from } \textbf{L2}\text{, RHS=}-1). \end{aligned} \tag{13}$$

When $0 \leq \phi_1 \leq \phi_1^* = \frac{\pi}{6}$, ϕ_2 is unrestricted in its range of $[0, \pi]$. When $\phi_1 > \phi_1^* = \frac{\pi}{6}$, the lower- and upper bounds of ϕ_2 are restricted, as determined next. Let $r = \frac{\cot\phi_1}{\sqrt{3}}$ (giving $r < 1$ when $\frac{\pi}{6} < \phi_1 < \frac{\pi}{2}$). Then, the lower bound $\phi_{2_l} \leq \phi_2$ is obtained by

solving $r = \cos\phi_2 - 2\sin\phi_2$, giving $\phi_{2_l} = 2\tan^{-1}(\frac{-2+\sqrt{5-r^2}}{1+r})$. Similarly, the upper bound $\phi_2 \leq \phi_{2_u}$ is obtained by solving $r = -\cos\phi_2 - 2\sin\phi_2$, giving $\phi_{2_u} = 2\tan^{-1}(\frac{+2+\sqrt{5-r^2}}{1-r})$.

Also, the values of ϕ_1 and ϕ_2 could restrict the range of ϕ_3, whose limits are obtained by setting the RHS to unity. The restrictions on ϕ_3 are obtained from:

$$\begin{aligned} \phi_1 = &\quad \cot^{-1}(\sqrt{3}\cos\phi_2 + 2\sqrt{3}\sin\phi_2) \text{ (from } \mathbf{L1}\text{, RHS=1)} \\ \phi_1 = &\quad \cot^{-1}(-\sqrt{3}\cos\phi_2 + 2\sqrt{3}\sin\phi_2) \text{ (from } \mathbf{L2}\text{, RHS=1).} \end{aligned} \tag{14}$$

All the limiting curves on the angles are illustrated in Figure 3, which shows the space spanned by the nominal ranges of $\phi_1 \in [0, \frac{\pi}{2}]$ and $\phi_2 \in [0, \pi]$. The space is divided into six different regions that impose different constraints on the ranges of ϕ_1, ϕ_2 and ϕ_3. The first two regions, labeled $\boldsymbol{R}_0^-$ and $\boldsymbol{R}_\pi^-$, demarcate the regions excluded to make RHS> -1 (Equation (13)), one on each side of $\phi_2 = 0$ and $\phi_2 = \pi$. In the region marked $\boldsymbol{R}_0^+$, the range of ϕ_3 is restricted from below to be greater than 0, and in the region marked $\boldsymbol{R}_\pi^+$, the range of ϕ_3 is restricted from the above to be less than 2π. In the region marked $\boldsymbol{R}_-^-$, ϕ_3 is restricted from both below and above. The appropriate lower bound ϕ_{3_l}, and upper bound ϕ_{3_u} on ϕ_3 can be computed accordingly. In the region marked $\boldsymbol{R}_+^+$, ϕ_3's original range of $[0, 2\pi)$ is unrestricted.

Thus, for **C1**, the ranges for post-collision parameters are:

$$\begin{aligned} &\phi_1 \in [0, \tfrac{\pi}{2}) \\ &\phi_2 \in \begin{cases} [0,0] \text{ if } \phi_1 = 0 \\ [0,\pi] \text{ if } 0 < \phi_1 \leq \frac{\pi}{6} \\ [\phi_{2_l}, \phi_{2_u}] \text{ otherwise } (\textit{i.e.}, \frac{\pi}{6} < \phi_1 < \frac{\pi}{2}) \end{cases} \\ &\phi_3 \in \begin{cases} [0,0] \text{ if } \phi_1 = 0 \text{ or } \phi_2 = 0 \text{ or } \phi_2 = \pi \\ [\phi_{3_l}, \phi_{3_u}] \text{ if } (\phi_1,\phi_2) \text{ falls in } \boldsymbol{R}_0^+, \boldsymbol{R}_\pi^+ \text{ or } \boldsymbol{R}_-^- \\ [0, 2\pi) \text{ otherwise } (\textit{i.e.}, (\phi_1,\phi_2) \text{ falls in } \boldsymbol{R}_+^+). \end{cases} \end{aligned} \tag{15}$$

Using a similar analysis, the pre-collision ranges can be obtained, as illustrated in Figure 3, which shows the range of $\phi_1^{'} \in (\frac{\pi}{2}, \pi]$, and corresponding constraints of $\phi_2^{'}$ and $\phi_2^{'}$ in terms of the *forward* regions $\boldsymbol{F}_0^-$, $\boldsymbol{F}_\pi^-$, and so on.

The forward and reverse procedures for this configuration are given in Procedure 2 and Procedure 3 respectively.

A.4. Configuration C2

Using algebra similar to that for **C1**, the ranges for configuration **C2** are obtained, with swapped signs for r_{32y} and r_{13y}.

A.5. Configuration C3

Configuration **C3** can be parameterized by an angle θ that **P3** makes relative to **P2**, as shown in Figure 4.

In this configuration, $r_{21x} = D$, $r_{21y} = 0$ (as was the case for **C1** and **C2**), but $r_{32x} = -D\cos\theta$, $r_{32y} = -D\sin\theta$, and we are not concerned about r_{13x} and r_{13y}. Using these in Equation (2), we get the geometrically constrained ranges of ϕ_1, ϕ_2, and ϕ_3. Since the inquality **K1** for this configuration is the same as for configurations **C1** and **C2**, the range of ϕ_1 remains $(0, \frac{\pi}{2}]$. However, only the inequality **K2** applies to this

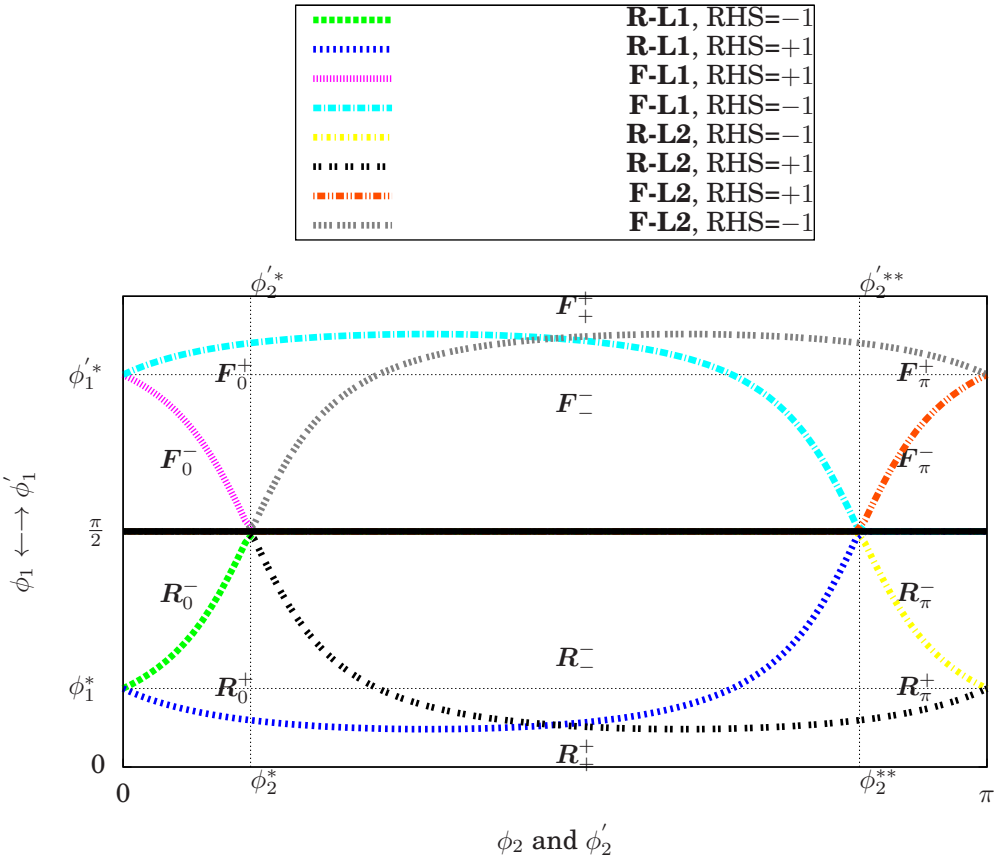

Fig. 3: Regions in the nominal phase spaces of ϕ_1 and ϕ_2 demarcating different bounds of ϕ_1, ϕ_2, and ϕ_3 in Configuration 1 of 3-particle collisions in 2 dimensions

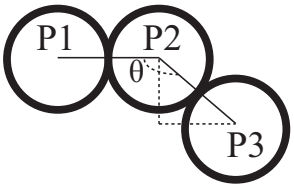

Fig. 4: Parametrization of configuration **C3** by angle θ, $\frac{\pi}{3} < \theta < \frac{5\pi}{3}$

configuration, and **K3** need not apply. From **K2**, we get:

$$-D\cos\theta\tfrac{\lambda}{2}(\sqrt{3}\sin\phi_1\sin\phi_2\cos\phi_3 - \cos\phi_1)$$
$$-D\sin\theta\tfrac{\lambda}{2}(\sqrt{3}\sin\phi_1\sin\phi_2\sin\phi_3 - \sin\phi_1\cos\phi_2) > 0\ .$$

Procedure 2 ($(\phi_1^{'}, \phi_2^{'}, \phi_3^{'}) \longrightarrow (\phi_1, \phi_2, \phi_3)$): *Forward Procedure for Configuration 1 of a 3-Particle Collision in 2 dimensions*

1: Invoke Procedure 1 to generate the parameters (ψ_1, ψ_2, ψ_3) of a random point on surface of the hyper-ellipsoid represented by Equation (3)
2: $\phi_1 \leftarrow (\phi_1^{'} - \frac{\pi}{2} + \psi_1) \mod \frac{\pi}{2}$
3: $\phi_2 \leftarrow (\phi_2^{'} + \psi_2) \mod \pi$
4: Compute ϕ_{2_l} and ϕ_{2_u} {based on ϕ_1}
5: **if** $\phi_2 < \phi_{2_l}$ **or** $\phi_{2_u} < \phi_2$ **then**
6: $\quad \phi_2 \leftarrow (\phi_2 + (\phi_{2_l} + \pi - \phi_{2_u})) \mod \pi$
7: **end if**
8: Compute ϕ_{3_l} and ϕ_{3_u} {based on ϕ_1 and ϕ_2}
9: $\phi_3 \leftarrow (\phi_3^{'} + \psi_3) \mod 2\pi$
10: **if** $\phi_3 < \phi_{3_l}$ **then**
11: $\quad \phi_3 \leftarrow (\phi_3 + \phi_{3_l}) \mod 2\pi$
12: **else if** $\phi_3 > \phi_{3_u}$ **then**
13: $\quad \phi_3 \leftarrow (\phi_3 + (2\pi - \phi_{3_u})) \mod 2\pi$
14: **end if**

Procedure 3 ($(\phi_1, \phi_2, \phi_3) \longrightarrow (\phi_1^{'}, \phi_2^{'}, \phi_3^{'})$): *Reverse Procedure for Configuration 1 of a 3-Particle Collision in 2 dimensions*

1: Invoke Procedure 1 to *re*-generate the parameters (ψ_1, ψ_2, ψ_3) of the random point on surface of the hyper-ellipsoid represented by Equation (3), previously generated by Procedure 2
2: Recompute ϕ_{2_l} and ϕ_{2_u} {based on ϕ_1}
3: Recompute ϕ_{3_l} and ϕ_{3_u} {based on ϕ_1 and ϕ_2}
4: $\phi_1^{'} \leftarrow \frac{\pi}{2} + ((\phi_1 - \psi_1) \mod \frac{\pi}{2})$
5: $\phi_2^{'} \leftarrow (\phi_2 - \psi_2) \mod \pi$
6: **if** $\phi_2^{'} < \phi_{2_l}$ **or** $\phi_{2_u} < \phi_2^{'}$ **then**
7: $\quad \phi_2^{'} \leftarrow (\phi_2^{'} - (\phi_{2_l} + \pi - \phi_{2_u})) \mod \pi$
8: **end if**
9: $\phi_3^{'} \leftarrow (\phi_3 - \psi_3) \mod 2\pi$
10: **if** $\phi_3^{'} < \phi_{3_l}$ **then**
11: $\quad \phi_3^{'} \leftarrow (\phi_3^{'} - \phi_{3_l}) \mod 2\pi$
12: **else if** $\phi_3^{'} > \phi_{3_u}$ **then**
13: $\quad \phi_3^{'} \leftarrow (\phi_3^{'} - (2\pi - \phi_{3_u})) \mod 2\pi$
14: **end if**

Since $\sin\phi_1 \neq 0$ and $\sin\phi_2 \neq 0$,

$$
\begin{aligned}
& \cos(\phi_3 - \theta)\sqrt{3}\sin\phi_1\sin\phi_2 < \cos\theta\cos\phi_1 + \sin\theta\sin\phi_1\cos\phi_2 \\
\Longrightarrow\ & \cos(\phi_3 - \theta) < \gamma, \text{ where} \\
& \gamma = \frac{\cos\theta\cos\phi_1 + \sin\theta\sin\phi_1\cos\phi_2}{\sqrt{3}\sin\phi_1\sin\phi_2} \\
\Longrightarrow\ & \phi_3 \in [\phi_{3_l} + \theta, \phi_{3_u} + \theta], \text{ where} \\
& 0 \leq (\phi_{3_l} + \theta) \bmod 2\pi \leq (\phi_{3_u} + \theta) \bmod 2\pi \leq 2\pi, \text{ and} \\
& \phi_{3_l} \text{ and } \phi_{3_u} \text{ are solutions to } \cos^{-1}\gamma.
\end{aligned}
\tag{16}
$$

Also, the range of ϕ_2 may be constrained due to the requirement that $-1 \leq \gamma \leq 1$. Let $\mu = \frac{\cos\theta}{\sqrt{3}\tan\phi_1}$ and $\nu = \frac{\sin\theta}{\sqrt{3}}$. Then, $\gamma = \frac{\mu+\nu\cos\phi_2}{\sin\phi_2}$. If $\mu - \nu \geq 0$ or $\mu + \nu \leq 0$, then the range of ϕ_2 is not constrained, giving the lowerbound $\phi_{2_l} \leq \phi_2$ equal to 0 and upperbound $\phi_2 \leq \phi_{2_u}$ equal to π. Otherwise, the lower bound (greater than 0) and upper bound (less than π) of ϕ_2 must be determined as follows.

If $\mu - \nu < 0$, the lowerbound of ϕ_2 is obtained by solving for ϕ_2 in $-\sin\phi_2 = \mu + \nu\cos\phi_2$. Since $\sin\phi_2$ and $\mu + \nu\cos\phi_2$ intersect in at most two points within the range $(0, \pi)$ (see Figure 5), a unique lowerbound ϕ_{2_l}, $0 < \phi_{2_l} \leq \phi_2 < \pi$ is obtainable. Similarly, if $\mu + \nu > 0$, then a unique upperbound ϕ_{2_u}, $0 < \phi_2 \leq \phi_{2_u} < \pi$ is obtained by solving for ϕ_2 in $\sin\phi_2 = \mu + \nu\cos\phi_2$.

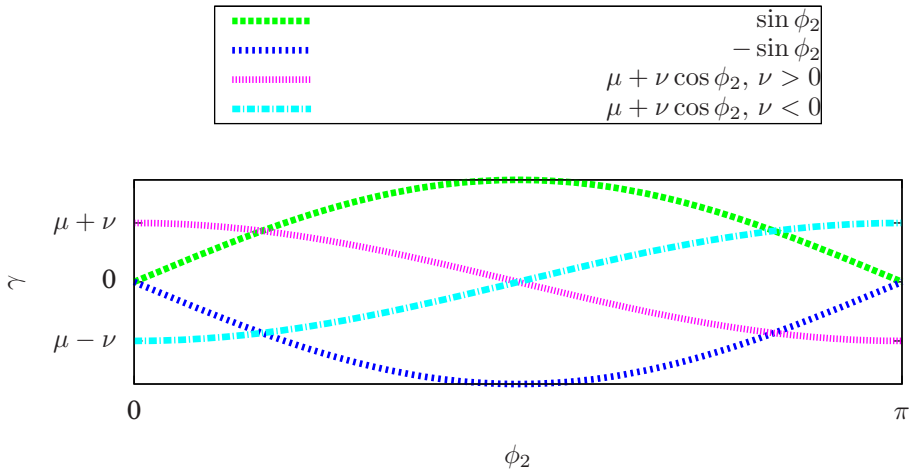

Fig. 5: Illustration of lower and upper bounds of ϕ_2 in configuration 3 of 3-particle collisions in 2 dimensions

Thus, for **C3**, the ranges are:

$$\begin{aligned}
\phi_1 &\in [0, \tfrac{\pi}{2}) \\
\phi_2 &\in \begin{cases} [0,0] \text{ if } \phi_1 = 0 \\ [0,\pi] \text{ if } \mu - \nu \geq 0 \text{ or } \mu + \nu \leq 0 \\ [\phi_{2_l}, \phi_{2_u}] \text{ otherwise} \end{cases} \\
\phi_3 &\in \begin{cases} [0,0] \text{ if } \phi_1 = 0 \text{ or } \phi_2 = 0 \text{ or } \phi_2 = \pi \\ [\phi_{3_l} + \theta, \phi_{3_u} + \theta] \text{ otherwise.} \end{cases}
\end{aligned} \tag{17}$$

The ranges may be derived similarly for pre-collision. The range of $\phi_1^{'}$ remains to be the same as in configuration 1 at $(\frac{\pi}{2}, \pi]$, but, Equation (16) changes to $\cos(\phi_3^{'} - \theta) > \gamma$.

A.6. Configuration C4

The treatment of configuration **C4** proceeds similar to that for configuration **C3**, using **K3** instead of **K2**.