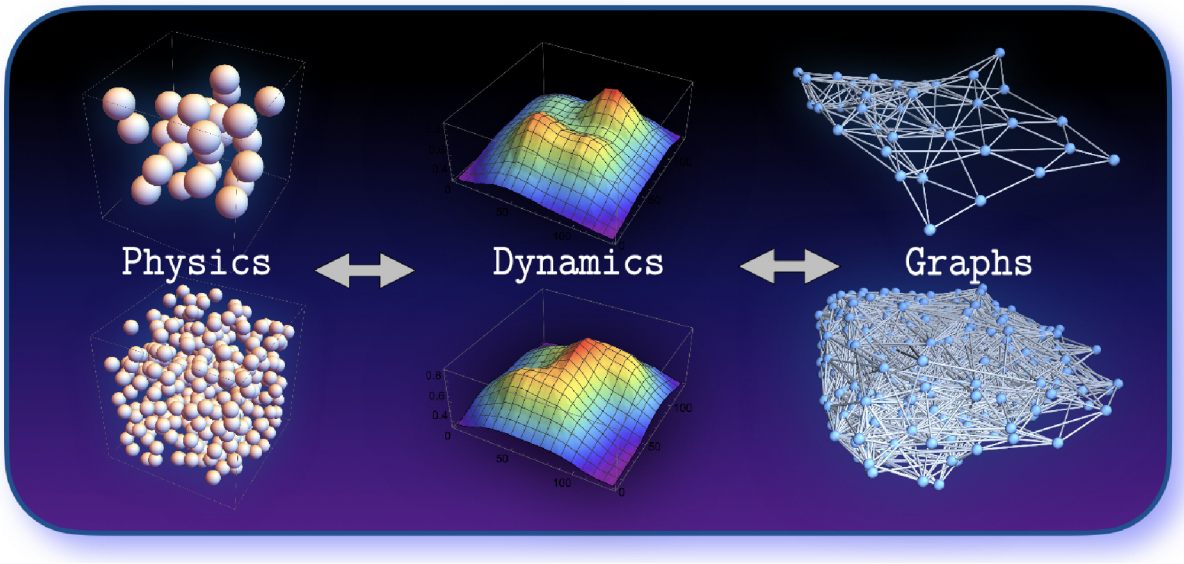


Characterizing Coarse Graining with Causal Graphs

Hanfeng Zhai

Cornell University

Abstract

Causal graphs are ubiquitous as a powerful tool for representing natural and engineered systems, including systems like molecular dynamics, particle and planetary collisions, chemical reactions, etc. Generally speaking, the higher the order, the higher the computational burden is required. In many engineering applications, scaling up the computation to capture the exact system is crucial. A crucial step is to reduce the order of the system with minimal loss of information, by performing a kind of coarse-graining procedure. In this project, we aim to understand and characterize coarse-graining as order reduction for hard sphere gas systems and develop a coarse-grained model for the causal graph derived from particle collision systems. We first examine the properties of spatial coarse-graining by testing various physical properties including density distribution, collision rate, kinetic energy, etc., and further examine their causal graphs by analyzing their adjacency matrices, motifs distributions, etc. We show that under ideal gas law, the proposed coarse-graining models are shown to be consistent in collision rate. We also propose a new method to do temporal coarse-graining on causal graphs retaining the same collision rate and test their motifs and related causal graph statistics. Our work sheds light on multiscale modeling and brings in a deeper understanding of coarse-graining by employing graph-based models.

Spatial Coarse-Graining for Macro-dynamics

Spatial Coarse-Grain: Survey and Formulation

For decades, a key problem in computational engineering is the bridge of multiple scales, i.e. multi-scale modeling. Physically, considering the length scale in the real world, this stands for bridging over Ångström, nanometers, micrometers, meters, etc. Mathematically and numerically, this could mean bridging different scales of order. Multiscale modeling includes bottom-up and top-down approaches. In bottom-up approaches, particle-based simulations are usually the fundamental approach. A central method employed for particle simulations is coarse-graining, i.e., reducing the order of the systems by assuming multiple smaller particles constitute a larger particle (Figure 1).

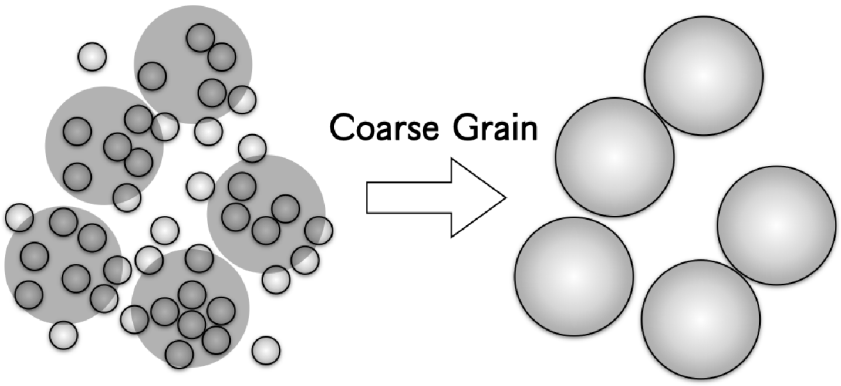


Figure 1 : Schematic of Spatial Coarse – Graining.

Such a simple and intuitive approach has been widely adopted in many scientific communities in molecular modeling, computational biology, granular flow, and many others. If one does not assume a new inter-particular forcefield, the hierarchical structure of the microscale particular structure is not preserved, making such a coarse-graining approach not feasible in problems like solid mechanics, soft materials, etc. However, if one is curious about the flow dynamics or thermodynamics of the gas state, just resizing the particle and rearranging the numbers with the same forcefield may also be considered as a way of coarse-graining. Such a method has been widely used in granular flows to study the morphology of multiphase systems that are widely used in 3D printing, manufacturing, etc. Our coarse-graining assumptions are mainly inspired by the work in discrete element simulations [Lu et al., 2016; Widartiningsih et al., 2020; Kanjilal & Schneiderbauer, 2021].

Here, we adopt such systems and examine the coarse-graining methods in the case of gas dynamics, curious about three core questions: (1) **Can this coarse-graining approach (order reduction) be applied to hard sphere gas systems?** (2) If the answer to (1) is Yes, **what are the feasibility and limitations of such an approach?** (3) **What are the hidden mechanism of such coarse-graining and can we bring in a deeper understanding with the Wolfram Physics Project?** To answer these ques-

tions, we begin by carrying out examine the global and local dynamics of different coarse-grained systems, and strive to understand the systems with causal graphs.

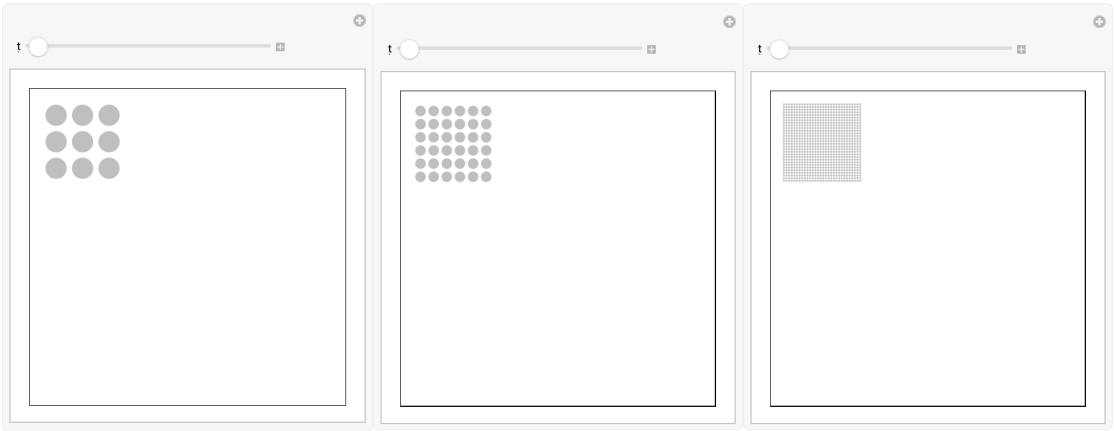
Macroscale Dynamics: An Illustrative Case

Overall Dynamics

We begin the study by setting up a classical simulation: a particle cluster was shoot from the left-upper corner to the right-lower corner and scattered into random gas dynamics [Wolfram, 2022]. The hypothesis is the original system contains 30 × 30 particles as a cluster, nominated as **system C**. This system can be coarse-grained to a cluster with 6 × 6 particles (25 → 1 particle), nominated as **system B**. System C can also be coarse-grained to a cluster with 3 × 3 particles (100 → 1 particle), nominated as **system A**. We can then simulate the dynamics of the three systems and observe their corresponding macroscale dynamics as the density profiles.

In[ ]:=
```
viz1 = ResourceFunction["HardSphereSimulation"][<|
    "Positions" → Catenate@Table[{-15 + 2.5 i, 15 - 2.5 j}, {i, 3}, {j, 3}],
    "Velocities" → Table[.5 {1, -1}, 9], "BoxSize" → 30,
    "StepSize" → 0.5, "ParticleRadius" → 1, "Steps" → 10,
    "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
viz2 = ResourceFunction["HardSphereSimulation"][<|"Positions" →
     Catenate@Table[{-14.375 + 1.25 i, 14.375 - 1.25 j}, {i, 6}, {j, 6}], "Velocities" →
     Table[.5 {1, -1}, 36], "BoxSize" → 30, "StepSize" → 0.5, "ParticleRadius" → 0.5,
    "Steps" → 10, "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
viz3 = ResourceFunction["HardSphereSimulation"][<|"Positions" →
     Catenate@Table[{-14 + .25 i, 14 - .25 j}, {i, 30}, {j, 30}], "Velocities" →
     Table[.5 {1, -1}, 900], "BoxSize" → 30, "StepSize" → 0.5, "ParticleRadius" → 0.1,
    "Steps" → 10, "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
Row[{viz1, viz2, viz3}]
```

Out[ ]=



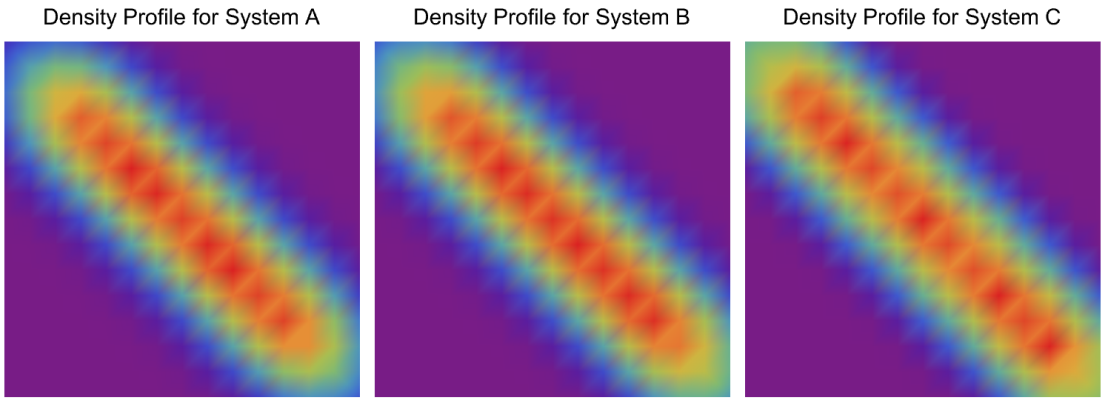
In[•]:=
```
colls9par = ResourceFunction["HardSphereSimulation"][<|
    "Positions" → Catenate@Table[{-15 + 2.5 i, 15 - 2.5 j}, {i, 3}, {j, 3}],
    "Velocities" → Table[.5 {1, -1}, 9], "BoxSize" → 30, "StepSize" → 0.5,
    "ParticleRadius" → 1, "Steps" → 10 000, "BoundaryCondition" → "Reflecting",
    "ParticleMass" → 1, "Output" → "PositionsByParticle"|>];
colls36par = ResourceFunction["HardSphereSimulation"][<|
    "Positions" → Catenate@Table[{-14.375 + 1.25 i, 14.375 - 1.25 j}, {i, 6}, {j, 6}],
    "Velocities" → Table[.5 {1, -1}, 36], "BoxSize" → 30, "StepSize" → 0.5,
    "ParticleRadius" → 0.5, "Steps" → 10 000, "BoundaryCondition" → "Reflecting",
    "ParticleMass" → 1 / 9, "Output" → "PositionsByParticle"|>];
colls900par = ResourceFunction["HardSphereSimulation"][<|
     "Positions" → Catenate@Table[{-14 + .25 i, 14 - .25 j}, {i, 30}, {j, 30}],
     "Velocities" → Table[.5 {1, -1}, 900], "BoxSize" → 30, "StepSize" → 0.5,
     "ParticleRadius" → 0.1, "Steps" → 10 000, "BoundaryCondition" → "Reflecting",
     "ParticleMass" → 1 / 100, "Output" → "PositionsByParticle"|>];
```

This whole process can be decomposed into three regimes: (1) the process from the left corner to the right corner, considered as the pseudo-flow dynamics; (2) the moment when the collision happens, considered as the collision dynamics; (3) random motion of the particles within the box, considered as random particle dynamics. The three macro density profiles of the three regimes are shown as flows.

Regime 1: Particle Cluster Pseudo-Flow Dynamics

In[ ]:=
```
overall9parregime1 = Flatten[colls9par[[All, 1 ;; 90, All]], 1];
overall36parregime1 = Flatten[colls36par[[All, 1 ;; 90, All]], 1];
overall900parregime1 = Flatten[colls900par[[All, 1 ;; 90, All]], 1];
Row[{over9parsmoothregime1 = SmoothDensityHistogram[overall9parregime1,
    PlotRange → {-15, 15}, Frame → None, ImagePadding → 0, ColorFunction → "Rainbow",
    PlotLabel → Style["Density Profile for System A", Black, 15], ImageSize → 250],
  over36parsmoothregime1 = SmoothDensityHistogram[overall36parregime1,
    PlotRange → {-15, 15}, Frame → None, ImagePadding → 0, ColorFunction → "Rainbow",
    PlotLabel → Style["Density Profile for System B", Black, 15], ImageSize → 250],
  over900parsmoothregime1 = SmoothDensityHistogram[overall900parregime1,
    PlotRange → {-15, 15}, Frame → None, ImagePadding → 0, ColorFunction → "Rainbow",
    PlotLabel → Style["Density Profile for System C", Black, 15], ImageSize → 250]}]
```

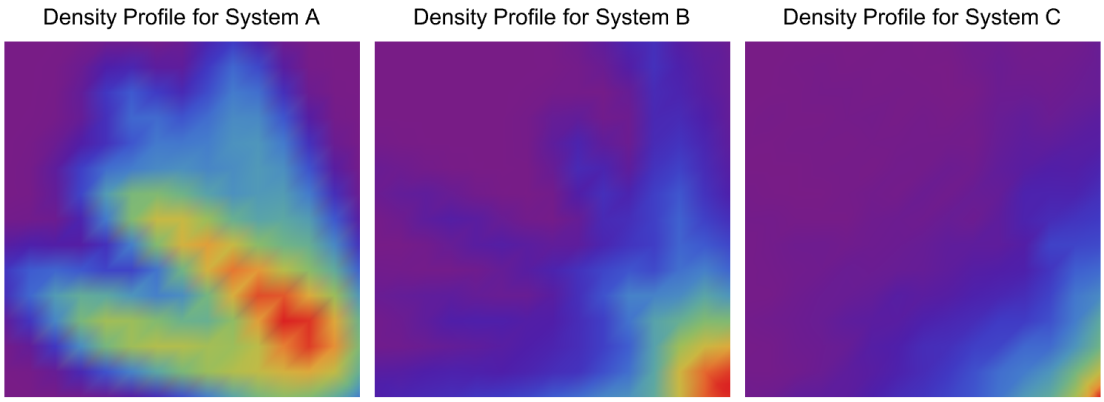
Out[ ]=



Regime 2 : Particle Cluster Collision Dynamics

In[ ]:=
```
overall9parregime2 = Flatten[colls9par[[All, 90 ;; 150, All]], 1];
overall36parregime2 = Flatten[colls36par[[All, 90 ;; 150, All]], 1];
overall900parregime2 = Flatten[colls900par[[All, 90 ;; 150, All]], 1];
Row[{over9parsmoothregime2 = SmoothDensityHistogram[overall9parregime2,
    PlotRange → {-15, 15}, Frame → None, ImagePadding → 0, ColorFunction → "Rainbow",
    PlotLabel → Style["Density Profile for System A", Black, 15], ImageSize → 250],
  over36parsmoothregime2 = SmoothDensityHistogram[overall36parregime2,
    PlotRange → {-15, 15}, Frame → None, ImagePadding → 0, ColorFunction → "Rainbow",
    PlotLabel → Style["Density Profile for System B", Black, 15], ImageSize → 250],
  over900parsmoothregime2 = SmoothDensityHistogram[overall900parregime2,
    PlotRange → {-15, 15}, Frame → None, ImagePadding → 0, ColorFunction → "Rainbow",
    PlotLabel → Style["Density Profile for System C", Black, 15], ImageSize → 250]}]
```

Out[ ]=

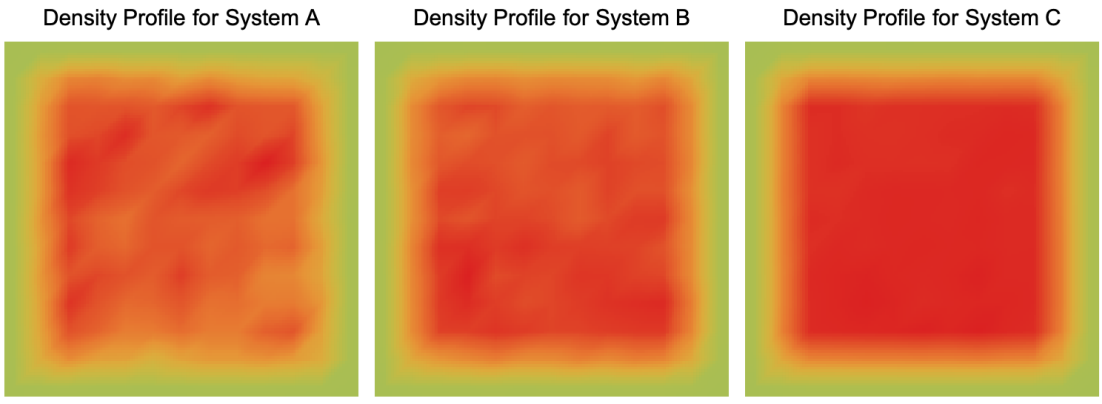


Regime 3 : Discretize Random Particle Dynamics

```
overall9parregime3 = Flatten[colls9par[[All, 150 ;; 9000, All]], 1];
overall36parregime3 = Flatten[colls36par[[All, 150 ;; 9000, All]], 1];
overall900parregime3 = Flatten[colls900par[[All, 150 ;; 9000, All]], 1];
figpar9rgm3 = SmoothDensityHistogram[overall9parregime3,
  PlotRange → All, Frame → None, ImagePadding → 0];
figpar9rgm3 = Image[figpar9rgm3, ImageSize → 150, ImageResolution → 72];
figpar9rgm3 = ImageData[figpar9rgm3];
figpar9rgm3 = Transpose[figpar9rgm3, {2, 3, 1}];
figpar9rgm3 = figpar9rgm3[[1 ;; All, 11 ;; 140, 11 ;; 140]];
figpar36rgm3 = SmoothDensityHistogram[overall36parregime3,
  PlotRange → All, Frame → None, ImagePadding → 0]; figpar36rgm3 =
 Image[figpar36rgm3, ImageSize → 150, ImageResolution → 72]; figpar36rgm3 =
 ImageData[figpar36rgm3];  figpar36rgm3 = Transpose[figpar36rgm3, {2, 3, 1}];
figpar36rgm3 = figpar36rgm3[[1 ;; All, 11 ;; 140, 11 ;; 140]];
figpar900rgm3 = SmoothDensityHistogram[overall900parregime3,
  PlotRange → All, Frame → None, ImagePadding → 0];
figpar900rgm3 = Image[figpar900rgm3, ImageSize → 150, ImageResolution → 72];
figpar900rgm3 = ImageData[figpar900rgm3];
figpar900rgm3 = Transpose[figpar900rgm3, {2, 3, 1}];
figpar900rgm3 = figpar900rgm3[[1 ;; All, 11 ;; 140, 11 ;; 140]];
```

In[ ]:=
```
Row[{MatrixPlot[Mean[figpar9rgm3], ColorFunction → "Rainbow", Frame → None,
    ImageSize → 250, PlotLabel → Style["Density Profile for System A", Black, 15]],
  MatrixPlot[Mean[figpar36rgm3], ColorFunction → "Rainbow", Frame → None,
    ImageSize → 250, PlotLabel → Style["Density Profile for System B", Black, 15]],
  MatrixPlot[Mean[figpar900rgm3], ColorFunction → "Rainbow", Frame → None,
    ImageSize → 250, PlotLabel → Style["Density Profile for System C", Black, 15]]}]
```

Out[ ]=



The visualizations above show that for both three regimes, similar density profiles for different coarse-grained systems. However, from the observations, one can only argue that based on rough observations it is difficult to extract the small differences between other systems, but one cannot quantitatively argue that the different coarse-grained systems have similar dynamics under specific criteria. To strengthen the argument, we first need to detect the exact differences between the details of other

systems. We decomposed the whole particle motion into snapshots to capture the local dynamics of the motion directly and visualize it as follows.

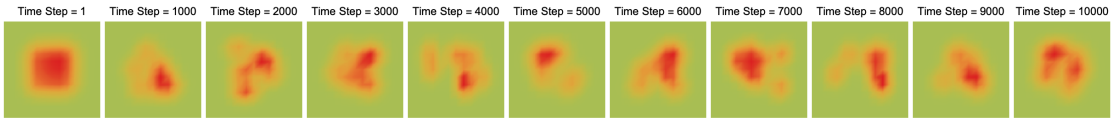
Local Dynamics

System A

System A contains 9 particles, and the density profile is extracted based on the spatial distribution of the particle at different time steps.

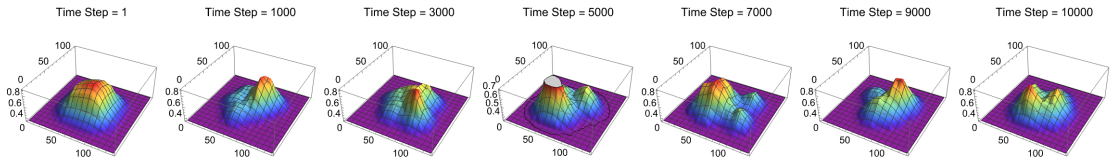
In[ ]:=
```
Row[{MatrixPlot[par9imgdat1, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 1", Black, 12]],
  MatrixPlot[par9imgdat100, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 1000", Black, 12]],
  MatrixPlot[par9imgdat200, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 2000", Black, 12]],
  MatrixPlot[par9imgdat300, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 3000", Black, 12]],
  MatrixPlot[par9imgdat400, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 4000", Black, 12]],
  MatrixPlot[par9imgdat500, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 5000", Black, 12]],
  MatrixPlot[par9imgdat600, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 6000", Black, 12]],
  MatrixPlot[par9imgdat700, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 7000", Black, 12]],
  MatrixPlot[par9imgdat800, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 8000", Black, 12]],
  MatrixPlot[par9imgdat900, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 9000", Black, 12]],
  MatrixPlot[par9imgdat1000, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 10000", Black, 12]]}]
```

Out[ ]=



In[ ]:=
```
Row[{ListPlot3D[par9imgdat1, ColorFunction → "Rainbow",
    ImageSize → 165, PlotLabel → Style["Time Step = 1", Black, 12]],
  ListPlot3D[par9imgdat100, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 1000", Black, 12]],
  ListPlot3D[par9imgdat300, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 3000", Black, 12]],
  ListPlot3D[par9imgdat500, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 5000", Black, 12]],
  ListPlot3D[par9imgdat700, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 7000", Black, 12]],
  ListPlot3D[par9imgdat900, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 9000", Black, 12]],
  ListPlot3D[par9imgdat1000, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 10000", Black, 12]]}]
```

Out[ ]=

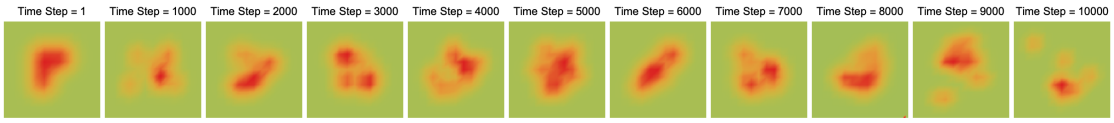


System B

System B contains 36 particles, and the density profile is extracted based on the spatial distribution of the particle at different time steps.

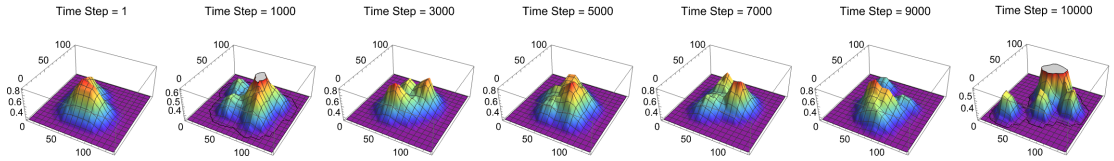
In[ ]:=
```
Row[{MatrixPlot[par36imgdat1, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 1", Black, 12]],
  MatrixPlot[par36imgdat100, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 1000", Black, 12]],
  MatrixPlot[par36imgdat200, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 2000", Black, 12]],
  MatrixPlot[par36imgdat300, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 3000", Black, 12]],
  MatrixPlot[par36imgdat400, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 4000", Black, 12]],
  MatrixPlot[par36imgdat500, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 5000", Black, 12]],
  MatrixPlot[par36imgdat600, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 6000", Black, 12]],
  MatrixPlot[par36imgdat700, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 7000", Black, 12]],
  MatrixPlot[par36imgdat800, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 8000", Black, 12]],
  MatrixPlot[par36imgdat900, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 9000", Black, 12]],
  MatrixPlot[par36imgdat1000, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 10000", Black, 12]]}]
```

Out[ ]=



In[ ]:=
```
Row[{ListPlot3D[par36imgdat1, ColorFunction → "Rainbow",
    ImageSize → 165, PlotLabel → Style["Time Step = 1", Black, 12]],
  ListPlot3D[par36imgdat100, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 1000", Black, 12]],
  ListPlot3D[par36imgdat300, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 3000", Black, 12]],
  ListPlot3D[par36imgdat500, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 5000", Black, 12]],
  ListPlot3D[par36imgdat700, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 7000", Black, 12]],
  ListPlot3D[par36imgdat900, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 9000", Black, 12]],
  ListPlot3D[par36imgdat1000, ColorFunction → "Rainbow", ImageSize → 165,
    PlotLabel → Style["Time Step = 10000", Black, 12]]}]
```

Out[ ]=

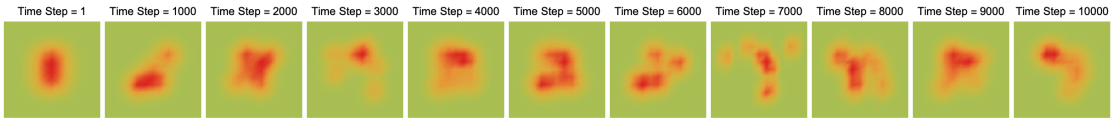


System C

System C contains 900 particles, and the density profile is extracted based on the spatial distribution of the particle at different time steps.

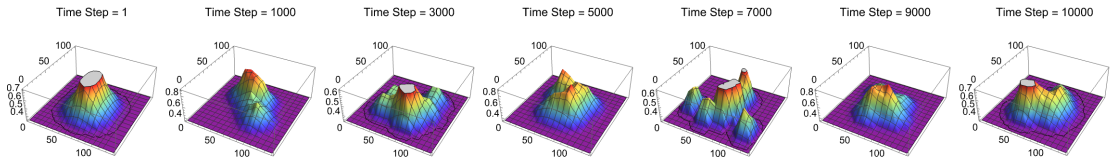
In[ ]:=
```
Row[{MatrixPlot[par900imgdat1, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 1", Black, 12]],
  MatrixPlot[par900imgdat100, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 1000", Black, 12]],
  MatrixPlot[par900imgdat200, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 2000", Black, 12]],
  MatrixPlot[par900imgdat300, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 3000", Black, 12]],
  MatrixPlot[par900imgdat400, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 4000", Black, 12]],
  MatrixPlot[par900imgdat500, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 5000", Black, 12]],
  MatrixPlot[par900imgdat600, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 6000", Black, 12]],
  MatrixPlot[par900imgdat700, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 7000", Black, 12]],
  MatrixPlot[par900imgdat800, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 8000", Black, 12]],
  MatrixPlot[par900imgdat900, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 9000", Black, 12]],
  MatrixPlot[par900imgdat1000, ColorFunction → "Rainbow", Frame → None,
   ImageSize → 110, PlotLabel → Style["Time Step = 10000", Black, 12]]}]
```

Out[ ]=



In[ ]:=
```
Row[{ListPlot3D[par900imgdat1, ColorFunction → "Rainbow",
    ImageSize → 165, PlotLabel → Style["Time Step = 1", Black, 12]],
  ListPlot3D[par900imgdat100, ColorFunction → "Rainbow", ImageSize → 165,
   PlotLabel → Style["Time Step = 1000", Black, 12]],
  ListPlot3D[par900imgdat300, ColorFunction → "Rainbow", ImageSize → 165,
   PlotLabel → Style["Time Step = 3000", Black, 12]],
  ListPlot3D[par900imgdat500, ColorFunction → "Rainbow", ImageSize → 165,
   PlotLabel → Style["Time Step = 5000", Black, 12]],
  ListPlot3D[par900imgdat700, ColorFunction → "Rainbow", ImageSize → 165,
   PlotLabel → Style["Time Step = 7000", Black, 12]],
  ListPlot3D[par900imgdat900, ColorFunction → "Rainbow", ImageSize → 165,
   PlotLabel → Style["Time Step = 9000", Black, 12]],
  ListPlot3D[par900imgdat1000, ColorFunction → "Rainbow", ImageSize → 165,
   PlotLabel → Style["Time Step = 10000", Black, 12]]}]
```

Out[ ]=



In this section, we directly visualize the physical intuition under particle-based coarse-graining: the macroscale observation, i.e., the density profile, is observed to be similar. However, from both mathematical and physical perspectives, this is not a strong argument as there is no exact quantitative or symbolic analysis directing this evidence. Following the idea, we first show the differences in the local dynamics and are to show how to characterize these different coarse-grained systems using graph models in the following section.

Characterization of Spatial Coarse-Grain

The Spatial Coarse-Grain Approach for Equilibrium Process

Following the previous sections, one can deduce that the three regimes of the illustrative example contain two nonequilibrium (flow and collision) and one equilibrium (random motion) process. As we did not modify the forcefield (hard sphere gas models), the larger particle does not preserve the hierarchical dynamics of the smaller particles, i.e., a larger particle cannot represent multiple smaller particles after a collision happens when the motion of smaller particles are randomized at a smaller time scale. This indicates that using this kind of coarse-graining to study the nonequilibrium process is not physically rigorous.

However, equilibrium processes, physically different coarse-grained systems confined in the box running for a long time until the overall thermodynamical properties of the systems reach the same, possess clear criteria to study and measure, e.g., Maxwell-Boltzmann function. One may also check the

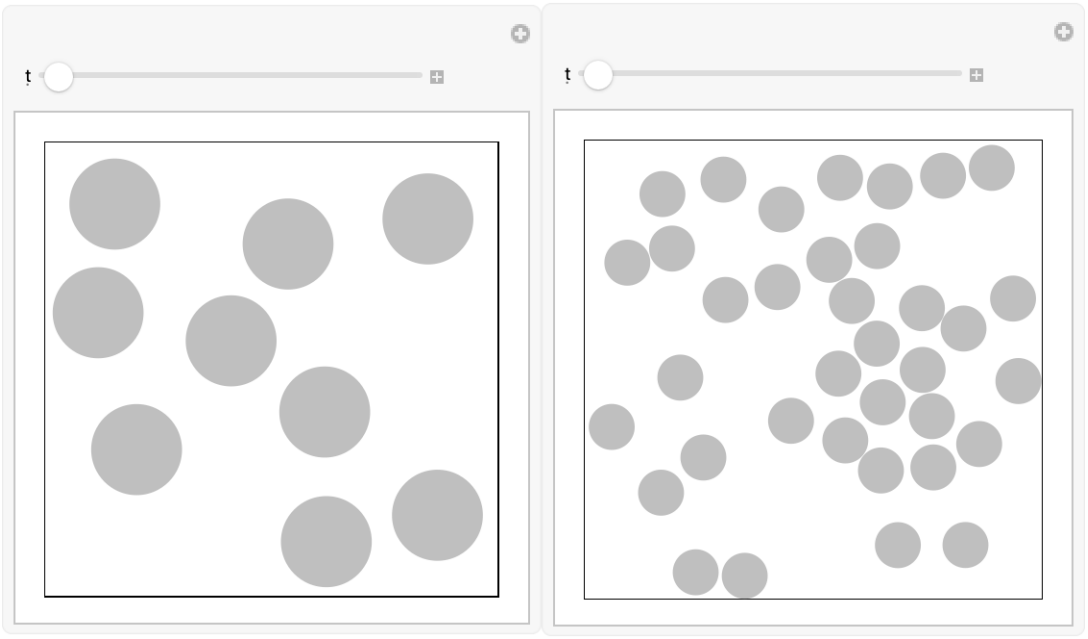
local dynamics of such systems, e.g., the collective behavior of particles in the short time regime.

Here, we investigate the equilibrium process. Assuming we have two systems containing 9 and 36 particles respectively. We want to look at both the hypergraphs and causal graphs of the two systems for the examination of the coarse-graining. We first run the two systems until thermodynamic equilibrium.

In[ ]:=
```
vizpar9 =
 ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
   "Velocities" → RandomPoint[Ball[{0, 0}], 9], "BoxSize" → 10,
   "StepSize" → 0.5, "ParticleRadius" → 1, "Steps" → 100,
   "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
vizpar36 = ResourceFunction["HardSphereSimulation"][<|"Positions" →
    "RandomNonOverlapping", "Velocities" → RandomPoint[Ball[{0, 0}], 36],
   "BoxSize" → 10, "StepSize" → 0.5, "ParticleRadius" → 0.5, "Steps" → 100,
   "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
```

In[ ]:=
```
Row[{vizpar9, vizpar36}]
```

Out[ ]=



The left visualization corresponds to the case of the 9-particle system and the right corresponds to the case of the 36-particle system.

Thermodynamics

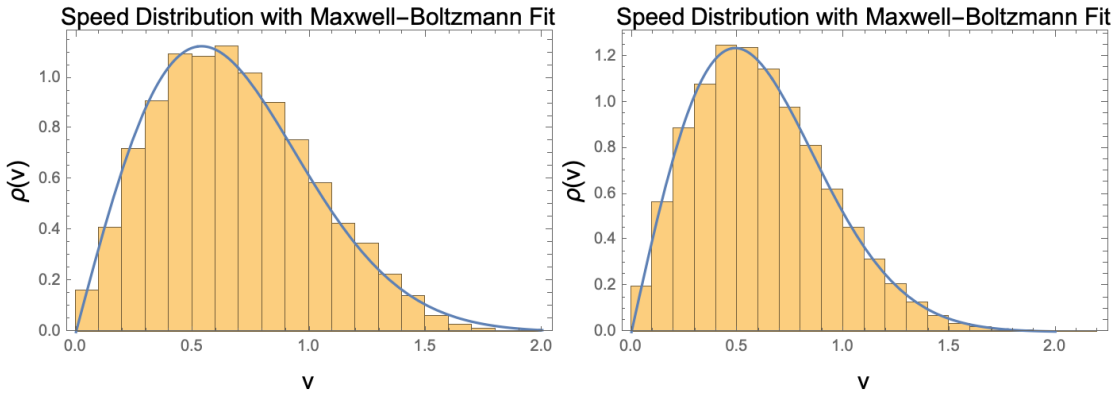
To check the thermodynamical equilibrium of the two systems, one can check the distribution of speeds w.r.t. temperature and fit with the Maxwell-Boltzmann function, shown as follows.

```
In[ ]:= Module[{speeds, Temp},
  {speeds, Temp} = {Flatten[#[[1]]], First[#[[2]]]} &@ResourceFunction[
      "HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
      "Velocities" → RandomPoint[Ball[{0, 0}], 9], "BoxSize" → 10, "StepSize" → 0.5,
      "ParticleRadius" → 1, "Steps" → 5000, "BoundaryCondition" → "Reflecting",
      "ParticleMass" → 1, "Output" → {"SpeedsByTime", "Temperature"}|>];
  thermopar9 = Show[(*Empirical histogram of speeds*)Histogram[speeds, {0.1`},
     "PDF", Frame → True, FrameLabel → (Style[#1, Black, 15] &) /@ {"v", "ρ(v)"},
     PlotLabel → Style["Speed Distribution with Maxwell-Boltzmann Fit", Black, 15],
     ImageSize → 350], (*Plot of 2D Maxwell-Boltzmann distribution with best-
     fit temperature.*)Plot[1 / Temp v E^(-(v^2 / (2 Temp))), {v, 0, 2}]]];
```

```
In[ ]:= Module[{speeds, Temp},
  {speeds, Temp} = {Flatten[#[[1]]], First[#[[2]]]} &@ResourceFunction[
      "HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
      "Velocities" → RandomPoint[Ball[{0, 0}], 36], "BoxSize" → 10, "StepSize" → 0.5,
      "ParticleRadius" → 0.5, "Steps" → 5000, "BoundaryCondition" → "Reflecting",
      "Output" → {"SpeedsByTime", "Temperature"}|>];
  thermopar36 = Show[Histogram[speeds, {0.1`}, "PDF", Frame → True,
     FrameLabel → (Style[#1, Black, 15] &) /@ {"v", "ρ(v)"},
     PlotLabel → Style["Speed Distribution with Maxwell-Boltzmann Fit", Black, 15],
     ImageSize → 350], Plot[1 / Temp v E^(-(v^2 / (2 Temp))), {v, 0, 2}]]];
```

```
In[ ]:= Row[{thermopar9, thermopar36}]
```

```
Out[ ]=
```



For a detailed analysis of these equilibrium systems using hypergraphs and causal graphs, please check Supplementary Section I and Supplementary Section II in the Appendix.

Examine the Properties of the Graphs

Under this equilibrium process, we can then explore the graphs representing the physical systems that strive to dig hidden mechanisms under this process. We explore both the hypergraph, i.e., the graph representing the interactions of the particles; and the causal graphs, i.e., the graph representing the collision events happening in chronological order; to provide more insights into different coarse-

grained systems.

The Hypergraphs

```
In[•]:= causalGpar9 =
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Ball[{0, 0}], 9], "BoxSize" → 10, "StepSize" → 0.5,
    "ParticleRadius" → 1, "Steps" → 5000, "BoundaryCondition" → "Reflecting",
    "Output" → {"Collisions", "Temperature", "CausalGraph"}|>];
```

```
In[•]:= causalGpar36 =
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Ball[{0, 0}], 36], "BoxSize" → 10, "StepSize" → 0.5,
    "ParticleRadius" → 0.5, "Steps" → 5000, "BoundaryCondition" → "Reflecting",
    "Output" → {"Collisions", "Temperature", "CausalGraph"}|>];
```

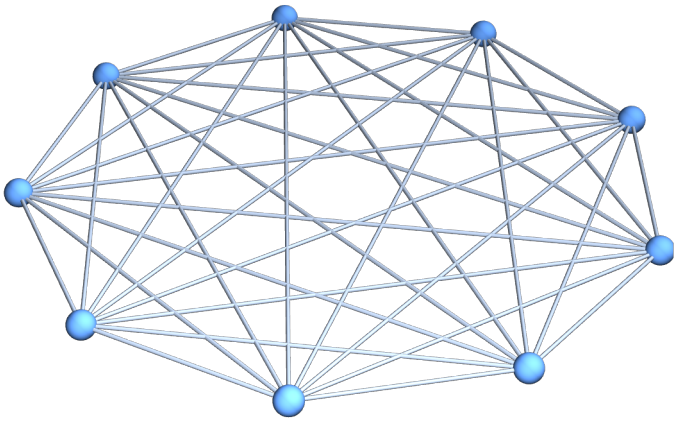
We can first visualize the hypergraphs:

```
In[•]:= hyperGpar9 = SimpleGraph[UndirectedEdge @@@ causalGpar9[[1]]];
hyperGpar36 = SimpleGraph[UndirectedEdge @@@ causalGpar36[[1]]];
```

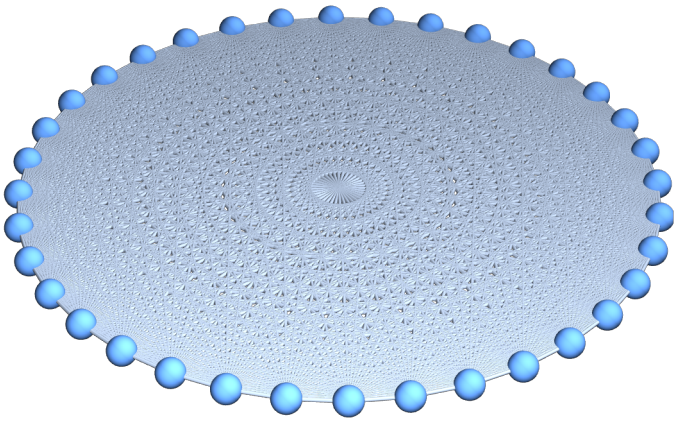
In[ ]:=
```
sysvizpar9 = GraphPlot3D[hyperGpar9, PlotLabel →
   Style["Hypergraph of the 9 particle systems", Black, 15], ImageSize → 350];
sysvizpar36 = GraphPlot3D[hyperGpar36, PlotLabel →
   Style["Hypergraph of the 36 particle systems", Black, 15], ImageSize → 350];
Row[{sysvizpar9, sysvizpar36}]
```

Out[ ]=

Hypergraph of the 9 particle systems



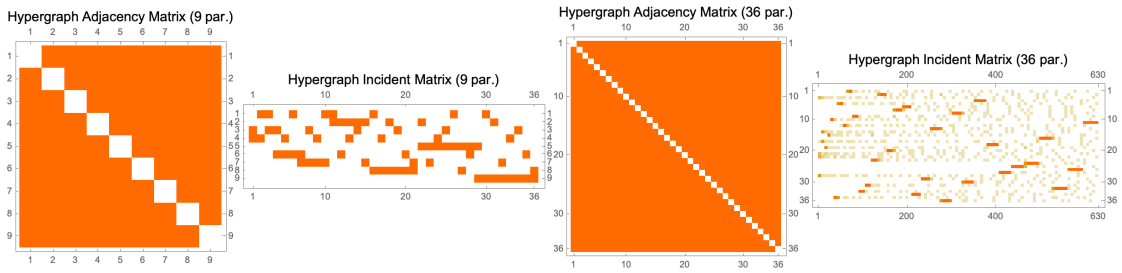
Hypergraph of the 36 particle systems



It can be deduced from the hypergraph that all the particles within the system interacted thoroughly as the graph nodes are fully connected. However, one may also be cautious that thermodynamically equilibrium is not equivalent to fully connected hypergraphs.

In[ ]:=
```
adjmathyperGpar9 = AdjacencyMatrix[hyperGpar9];
incimathyperGpar9 = IncidenceMatrix[hyperGpar9];
Plotadjmhyperpar9 = MatrixPlot[adjmathyperGpar9, PlotLabel →
    Style["Hypergraph Adjacency Matrix (9 par.)", Black, 15], ImageSize → 250];
Plotincimhyperpar9 = MatrixPlot[incimathyperGpar9, PlotLabel →
    Style["Hypergraph Incident Matrix (9 par.)", Black, 15], ImageSize → 350];
adjmathyperGpar36 = AdjacencyMatrix[hyperGpar36];
incimathyperGpar36 = IncidenceMatrix[hyperGpar36];
Plotadjmhyperpar36 = MatrixPlot[adjmathyperGpar36, PlotLabel →
    Style["Hypergraph Adjacency Matrix (36 par.)", Black, 15], ImageSize → 270];
Plotincimhyperpar36 = MatrixPlot[incimathyperGpar36, PlotLabel →
    Style["Hypergraph Incident Matrix (36 par.)", Black, 15], ImageSize → 350];
Row[{Plotadjmhyperpar9, Plotincimhyperpar9,
  Plotadjmhyperpar36, Plotincimhyperpar36}]
```

Out[ ]=



Another way to visualize the properties of the graph is through the adjacency and incident matrices. When the graph size is large it is difficult to deduce the connectivity of the hypergraphs, in which the adjacency matrix plays a more important role when scaling up the systems. From the above adjacency matrices, one can reaffirm that the particles fully interacted with each other, as all the elements besides the diagonal possess the value of 1 (marked in orange). The hypergraph matrices stand for the interaction orders between the particles, in which one can hardly deduce any hierarchal orders or similar patterns. Hence, this indicates that interactions between particles are more random, where our spatial coarse-graining approach cannot capture the dynamics. Another important property of the systems is the collision events, which can be preserved in the causal graphs shown below.

The Causal Graphs

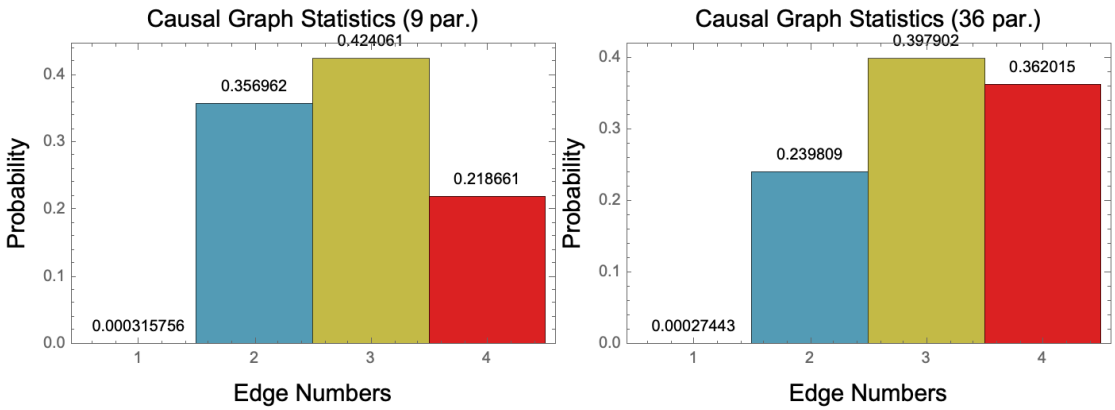
Statistics of the Graphs

Due to the large size of the causal graphs, it is extremely difficult to directly visualize the whole graph. We here only show the statistics hidden within the causal graph by counting the edge number probability distribution. It can be seen that both causal graphs have the most 3-edge nodes, with slightly different 2-edge and 4-edge nodes. This may indicate that the connections between collision events are slightly different for the two coarse-grained systems.

In[ ]:=
```
causalgraphpar9 = causalGpar9[[3]]; causalgraphpar36 = causalGpar36[[3]];
```

In[ ]:=
```
histcausalGpar9 =
 Histogram[DegreeCentrality[TransitiveReductionGraph[causalgraphpar9]],
  Automatic, "Probability",
  FrameLabel → (Style[#1, Black, 15] &) /@ {"Edge Numbers", "Probability"},
  PlotLabel → Style["Causal Graph Statistics (9 par.)", Black, 15], Frame → True,
  ImageSize → 350, ChartStyle → "Rainbow", LabelingFunction → (Placed[#1, Above] &)];
histcausalGpar36 = Histogram[DegreeCentrality[
   TransitiveReductionGraph[causalgraphpar36]], Automatic, "Probability",
  FrameLabel → (Style[#1, Black, 15] &) /@ {"Edge Numbers", "Probability"},
  PlotLabel → Style["Causal Graph Statistics (36 par.)", Black, 15], Frame → True,
  ImageSize → 350, ChartStyle → "Rainbow", LabelingFunction → (Placed[#1, Above] &)];
Row[{histcausalGpar9, histcausalGpar36}]
```

Out[ ]=

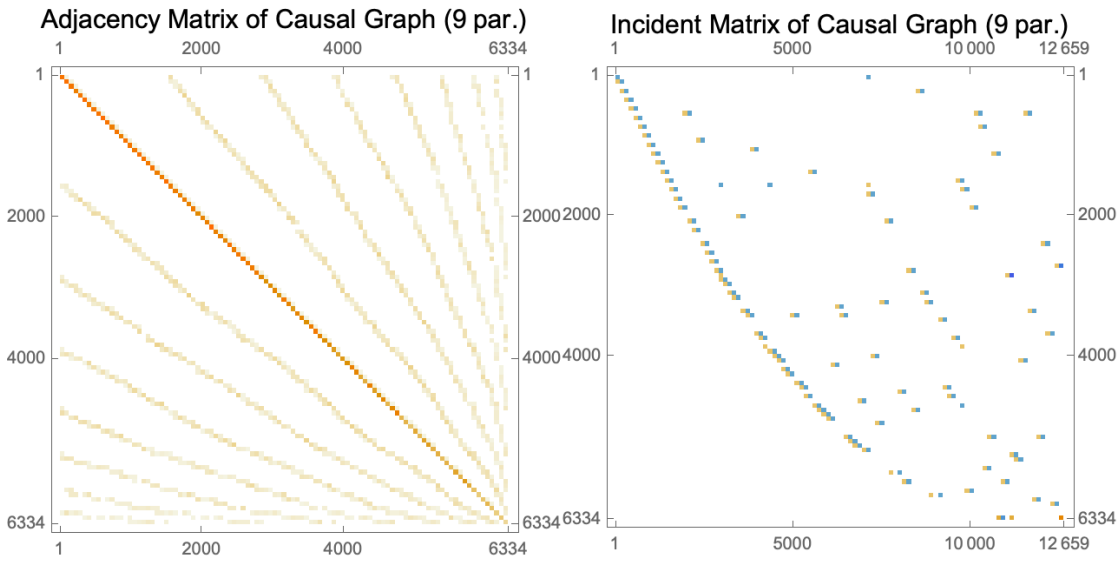


Characterization Matrices

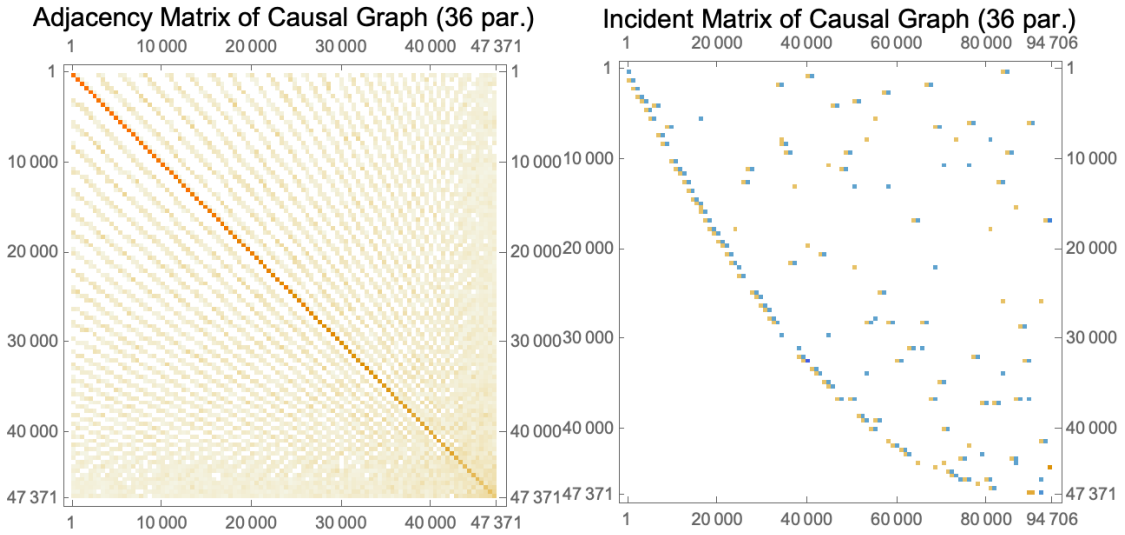
We can also plot the adjacency and incident matrices for the causal graphs. The adjacency matrices for the causal graph stand for the direct connections between the collision events. The incident matrices shall stand for all the different types of connections. Just from purely visual observations, one may find that the coarse-grained system possesses a sparser adjacency matrix with a similar trend and both the incident matrices are observed to be similar. However, this does not indicate a direct connection between the two. The goal here is to provide new insights from the perspective of considering the connections between the events for depicting the local dynamics. (One can also see more in the Supplementary Section for other adjacency and incident matrices of other systems)

In[ ]:=
```
par9causaladjmat = MatrixPlot[AdjacencyMatrix[causalgraphpar9],
    PlotLabel → Style["Adjacency Matrix of Causal Graph (9 par.)", Black, 15],
    ImageSize → 320];
par9causalincimat = MatrixPlot[IncidenceMatrix[causalgraphpar9], PlotLabel →
     Style["Incident Matrix of Causal Graph (9 par.)", Black, 15], ImageSize → 320];
par9causalredincimat =
 MatrixPlot[IncidenceMatrix[TransitiveReductionGraph[causalgraphpar9]],
  PlotLabel → Style["Reduced Incident Matrix (9 par.)", Black, 15], ImageSize → 320];
par36causaladjmat = MatrixPlot[AdjacencyMatrix[causalgraphpar36], PlotLabel → Style[
     "Adjacency Matrix of Causal Graph (36 par.)", Black, 15], ImageSize → 320];
par36causalincimat = MatrixPlot[IncidenceMatrix[causalgraphpar36],
    PlotLabel → Style["Incident Matrix of Causal Graph (36 par.)", Black, 15],
    ImageSize → 320];
par36causalredincimat = MatrixPlot[
    IncidenceMatrix[TransitiveReductionGraph[causalgraphpar36]], PlotLabel →
     Style["Reduced Incident Matrix (36 par.)", Black, 15], ImageSize → 320];
Row[{par9causaladjmat, par9causalincimat}]
Row[{par36causaladjmat, par36causalincimat}]
```

Out[ ]=



Out[ ]=



Motifs Representing the Local Graph Dynamics

Since we care about the local structure of the causal graph representing the connections of collision events. A better way to directly visualize the statistics of those connections is to observe the probability of motifs in the causal graph. Here, we use to {I, II, E, Y, K, A, V} to denote 7 different kinds of motifs:

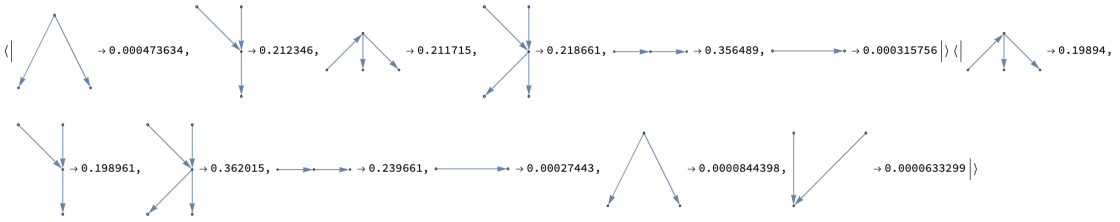
The statistics of the causal graphs are shown below.

In[ ]:=
```
motifcgpar9 = TransitiveReductionGraph[causalgraphpar9];
motifgrouppar9 = GroupBy[NeighborhoodGraph[motifcgpar9, #, 1] & /@
    VertexList[motifcgpar9], CanonicalGraph, Length];
motifcgpar36 = TransitiveReductionGraph[causalgraphpar36];
motifgrouppar36 = GroupBy[NeighborhoodGraph[motifcgpar36, #, 1] & /@
    VertexList[motifcgpar36], CanonicalGraph, Length];
```

In[ ]:=
```
probdistpar9 = Values[motifgrouppar9] / Total[Values[motifgrouppar9]] // N;
probdistpar36 = Values[motifgrouppar36] / Total[Values[motifgrouppar36]] // N;
probplotpar9 = AssociationThread[Keys[motifgrouppar9] → probdistpar9];
probplotpar36 = AssociationThread[Keys[motifgrouppar36] → probdistpar36];
```

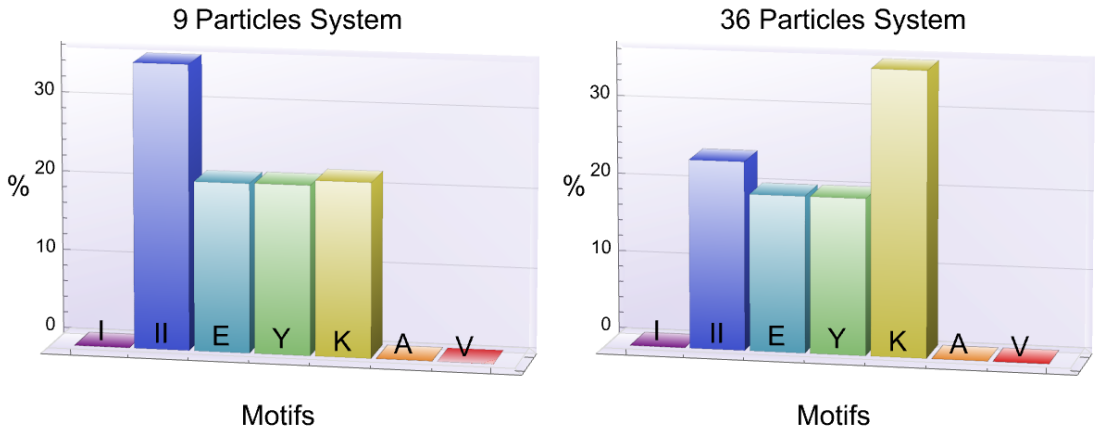
In[ ]:=
```
Row[{probplotpar9, probplotpar36}]
```

Out[ ]=



In[ ]:=
```
distrimotifpar9 = BarChart3D[
   {100 {0.00031575623618566466`, 0.3564887906536154`, 0.21171455636248815`,
      0.21234606883485949`, 0.21866119355857278`, 0.000473634354278497`, 0}},
   ChartLabels → (Style[#1, Black, 15] &) /@ {"I", "II", "E", "Y", "K", "A", "V"},
   AxesLabel → (Style[#1, Black, 15] &) /@ {"Motifs", "", "%"}, ChartStyle → "Rainbow",
   PlotLabel → Style["9 Particles System", Black, 15], ImageSize → 300];
distrimotifpar36 = BarChart3D[{100 {0.00027442950328259905`,
      0.23966139621287286`, 0.19894027991809335`, 0.19896138987988432`,
      0.36201473475333007`, 0.00008443984716387663`, 0.00006332988537290748`}},
   ChartLabels → (Style[#1, Black, 15] &) /@ {"I", "II", "E", "Y", "K", "A", "V"},
   AxesLabel → (Style[#1, Black, 15] &) /@ {"Motifs", "", "%"}, ChartStyle → "Rainbow",
   PlotLabel → Style["36 Particles System", Black, 15], ImageSize → 300];
Row[{distrimotifpar9, distrimotifpar36}]
```

Out[ ]=



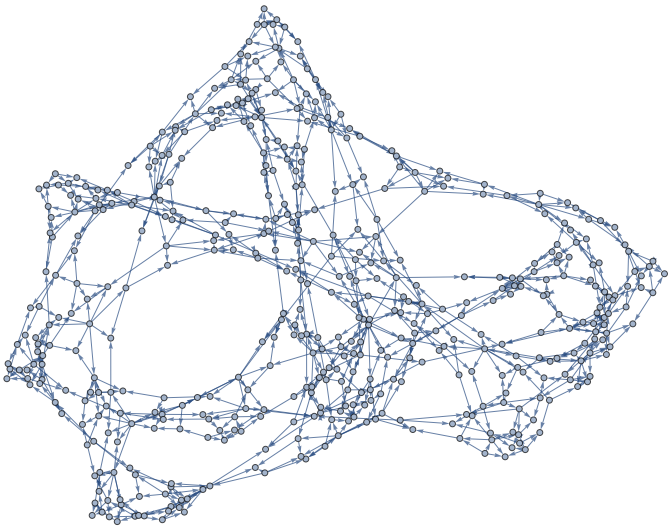
Just as the edge number counts, the probability distribution of the motifs is reported to be similar for the two different coarse-grained systems, yet the distribution is slightly different with different ratios of II- and K-typed motifs in the causal graphs.

Benchmark Cases for Graph Characterizations

However, based on our previous analysis and examinations, one may ask an intuitive question: does all the causal graph look similar? To answer this, we also adopt some models from the Wolfram Physics project to illustrate some characteristic matrices from the rewriting systems. Taking the simple rule $(x, y), (x, z) \to (x, z), (x, w), (y, w), (z, w)$, we can first visualize the corresponding hypergraph:

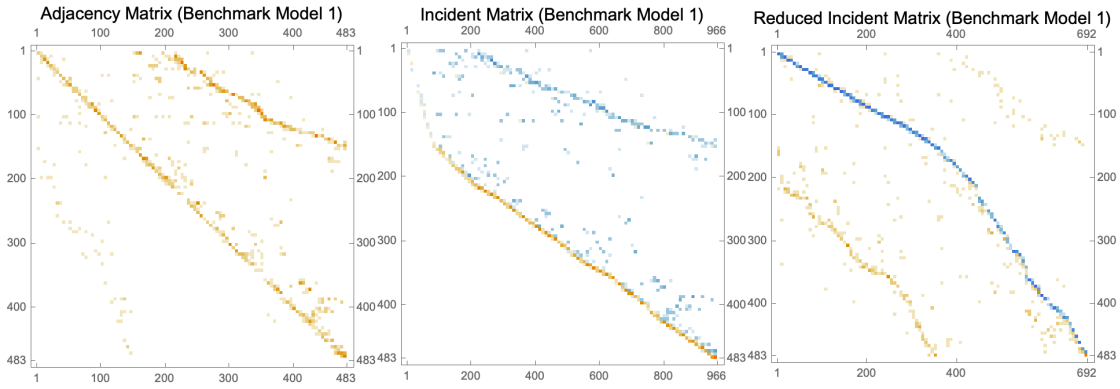
In[ ]:=
```
Wolframmodelsample1 = Graph[HypergraphToGraph[
```
Version (latest): 2.0.0
Documentation »
```
WolframModel[{{x, y}, {x, z}} → {{x, z}, {x, w}, {y, w}, {z, w}},
```
Version (latest): 7.0.0
Documentation »
```
{{0, 0}, {0, 0}}, 10, "FinalState"]]]
```

Out[ ]=



In[ ]:=
```
adjmatsample1 = MatrixPlot[AdjacencyMatrix[Wolframmodelsample1], PlotLabel →
    Style["Adjacency Matrix (Benchmark Model 1)", Black, 15], ImageSize → 320];
incimatsample1 = MatrixPlot[IncidenceMatrix[Wolframmodelsample1], PlotLabel →
    Style["Incident Matrix (Benchmark Model 1)", Black, 15], ImageSize → 320];
eigenincimatsample1 = MatrixPlot[IncidenceMatrix[
    TransitiveReductionGraph[Wolframmodelsample1]], PlotLabel → Style[
     "Reduced Incident Matrix (Benchmark Model 1)", Black, 15], ImageSize → 320];
Row[{adjmatsample1, incimatsample1, eigenincimatsample1}]
```

Out[ ]=



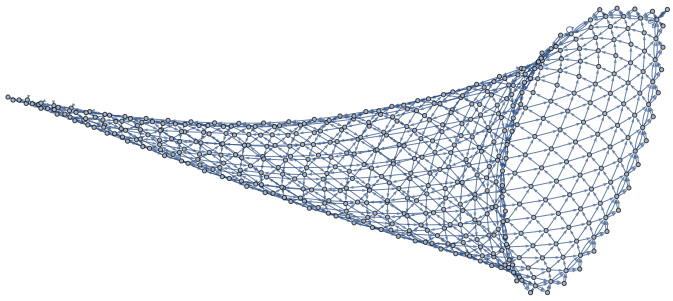
Taking another case, assuming we have the transformation rule:
$(x, x, y), (x, z, u) \to (u, u, z), (y, v, z), (y, v, z)$

In[ ]:=
```
Wolframmodelsample2 =
Graph[HypergraphToGraph[WolframModel[{{x, x, y}, {x, z, u}} →
    {{u, u, z}, {y, v, z}, {y, v, z}}, {{0, 0, 0}, {0, 0, 0}}, 500, "FinalState"]]]
```

HypergraphToGraph — Version (latest): 2.0.0 — Documentation »

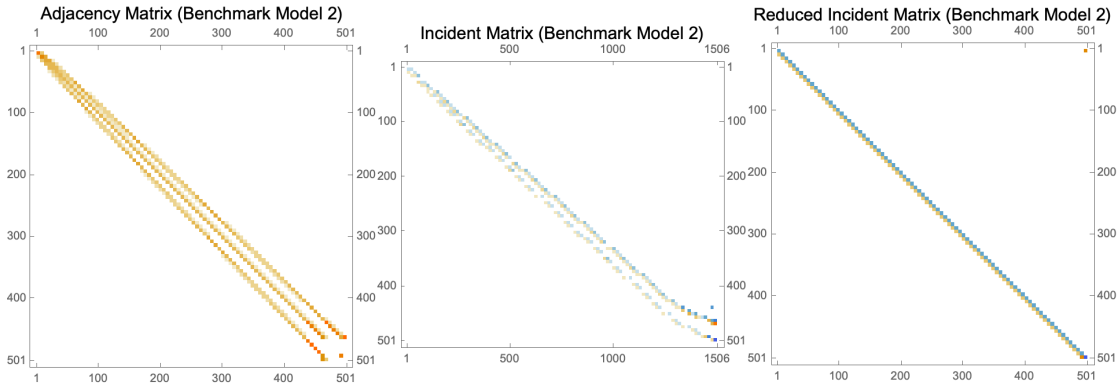
WolframModel — Version (latest): 7.0.0 — Documentation »

Out[ ]=



In[ ]:=
```
adjmatsample2 = MatrixPlot[AdjacencyMatrix[Wolframmodelsample2], PlotLabel →
    Style["Adjacency Matrix (Benchmark Model 2)", Black, 15], ImageSize → 320];
incimatsample2 = MatrixPlot[IncidenceMatrix[Wolframmodelsample2], PlotLabel →
    Style["Incident Matrix (Benchmark Model 2)", Black, 15], ImageSize → 320];
eigenincimatsample2 = MatrixPlot[IncidenceMatrix[
    TransitiveReductionGraph[Wolframmodelsample2]], PlotLabel → Style[
     "Reduced Incident Matrix (Benchmark Model 2)", Black, 15], ImageSize → 320];
Row[{adjmatsample2, incimatsample2, eigenincimatsample2}]
```

Out[ ]=



Spatial and Temporal Coarse-Graining

There are two ways to do coarse-grain to reduce the order of the system for reduced computational costs, spatial and temporal. For spatial, we take the common approach used in the discrete element

communities, which assumes by enlarging the particle size and decreasing the particle numbers in ratio, the macroscopic behavior of the system should roughly be the same. We compare different systems of the same box running with the same time steps but with different particle numbers and radii. We examine this assumption by testing the linear relation between the collision rate and the square root of the temperature. For the temporal coarse-grain approach, we directly operate on the causal graph by uniformly subtracting the "extra" node (representing the collision events) from the overall causal graph. By resampling the causal and assign with new labels to represent the density of the new nodes, the temporal coarse-grained causal graph is able to preserve the collision rates.

The two approaches are introduced and examined in different cases with different particle sizes and numbers. Their corresponding assumptions, verifications, and limitations are then briefly elaborated on and discussed. Most importantly, we indicate under which scenario these coarse-grain approaches then tend to fail, i.e., preserve the physics of the original system.

Examine the Physics from the Causal Graph for Spatial Coarse-Graining

Beginning with the previous thoughts on resizing the particles and changing the particle numbers, one hopes to preserve the volume fraction between the particles and "vacuum" space. Under this assumption, we begin exploring physics by testing the relationship between the collision rate and the square root of the temperature.

To show how these two physical quantities, we first recall the second law of thermodynamics, according to ideal gas law [Thoennessen, 1995]: $P\,V = n\,R\,T$, where $R = 8.314\,J/(m\,o\,l\cdot K)$ is the universal gas constant and n corresponds to the quantity of gas in moles. Also, based on the ideal gas approximation, we know: $P\,V = 2/3\,N\,\overline{K\,E}$, where $\overline{K\,E}$ is the mean kinetic energy of an individual molecule and N is the total number of molecules in the gas. We also know that the kinetic energy relates to the velocity by its square: $\overline{K\,E} \sim v^2$, and the velocity is directly related to the collision rate $c\,r$. Hence, one should expect a linear relationship between the collision rate with the square root of the temperature.

To examine this relation, we begin by defining a function that takes a series of different initial velocities (of the particles), varying box sizes, and different particle numbers to obtain the corresponding temperature and collision rate from the hard sphere simulations:

In[•]:=
```
TempVsCRcoarsegrain[v_, bs_, num_] := Module[{dataset, temp, finals, finaltime, cr},
  (*Compute dataset*)
  dataset =
   ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
     "Velocities" → RandomPoint[Sphere[{0, 0}, v], num], "BoxSize" → bs,
     "StepSize" → 0.1, "ParticleRadius" → 1, "Steps" → 5000,
     "ParticleMass" → (1 / bs) * ConstantArray[1, num], "BoundaryCondition" →
      "Reflecting", "Output" → {"Collisions", "Temperature"}|>];
  (*Compute temperature*)
temp = First[dataset[[2]]];
  (*Extract collision rate (next three lines)*)
finals = MaximalBy[dataset[[1]], #[[3]] &];
finaltime = First[finals][[3]];
cr = Length[dataset[[1]]] / finaltime;
  (*Return both temperature and collision rate*)
  {temp, cr}
 ]
```

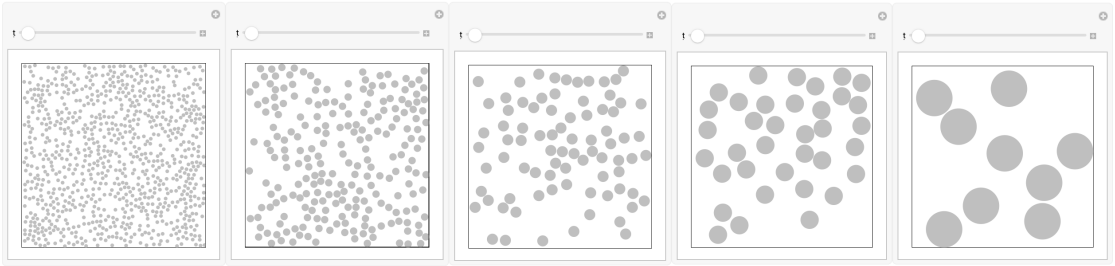
Given the relationship between the physical quantities, we can define the two simulation systems differently by varying the box sizes and particle radii for spatial coarse-graining, which is equivalent to changing the particles' radii and total numbers. The key goal is to keep the volume fraction of the systems roughly the same. Simultaneously, based on the density of the sphere should hold the same before and after we do coarse-graining, the mass of the sphere should also change as we change the box size (the equivalent to changing the sphere radii, holding the volume fraction constant).

Now, we can assume 5 different degrees of coarse-graining. Assuming a fine systems with $30 \times 30 = 900$ particles. Denote the original system as system A, this system can also be coarse-grained into (B) $10 \to 5 : 15 \times 15 = 225$ particles; (C) $10 \to 3 : 9 \times 9 = 81$ particles; (D) $10 \to 2 : 6 \times 6 = 36$ particles; (E) $10 \to 1 : 3 \times 3 = 9$ particles. Assuming system E has a box size of 10, the corresponding box sizes for the four others are (A) $10 \times \frac{900}{9} = 1000$; (B) $10 \times \frac{225}{9} = 250$; (C) $10 \times \frac{81}{9} = 90$; (D) $10 \times \frac{36}{9} = 40$. We can then test the physics of the four systems and check whether they agrees with the second law of thermodynamics with the same fitting coefficients.

To get a good sense of how the system may look like, we generate the visualizations as follows:

```
vizsysA =
 ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
   "Velocities" → RandomPoint[Ball[{0, 0}, 1], 900], "BoxSize" → 10,
   "StepSize" → 0.5, "ParticleRadius" → 0.1, "Steps" → 1,
   "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
vizsysB = ResourceFunction["HardSphereSimulation"][<|"Positions" →
    "RandomNonOverlapping", "Velocities" → RandomPoint[Ball[{0, 0}, 1], 225],
   "BoxSize" → 10, "StepSize" → 0.5, "ParticleRadius" → 0.2, "Steps" → 1,
   "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
vizsysC = ResourceFunction["HardSphereSimulation"][<|"Positions" →
    "RandomNonOverlapping", "Velocities" → RandomPoint[Ball[{0, 0}, 1], 81],
   "BoxSize" → 10, "StepSize" → 0.5, "ParticleRadius" → 0.3, "Steps" → 1,
   "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
vizsysD = ResourceFunction["HardSphereSimulation"][<|"Positions" →
    "RandomNonOverlapping", "Velocities" → RandomPoint[Ball[{0, 0}, 1], 36],
   "BoxSize" → 10, "StepSize" → 0.5, "ParticleRadius" → 0.5, "Steps" → 1,
   "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
vizsysE = ResourceFunction["HardSphereSimulation"][<|"Positions" →
    "RandomNonOverlapping", "Velocities" → RandomPoint[Ball[{0, 0}, 1], 9],
   "BoxSize" → 10, "StepSize" → 0.5, "ParticleRadius" → 1, "Steps" → 1,
   "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];

Row[{vizsysA, vizsysB, vizsysC, vizsysD, vizsysE}]
```

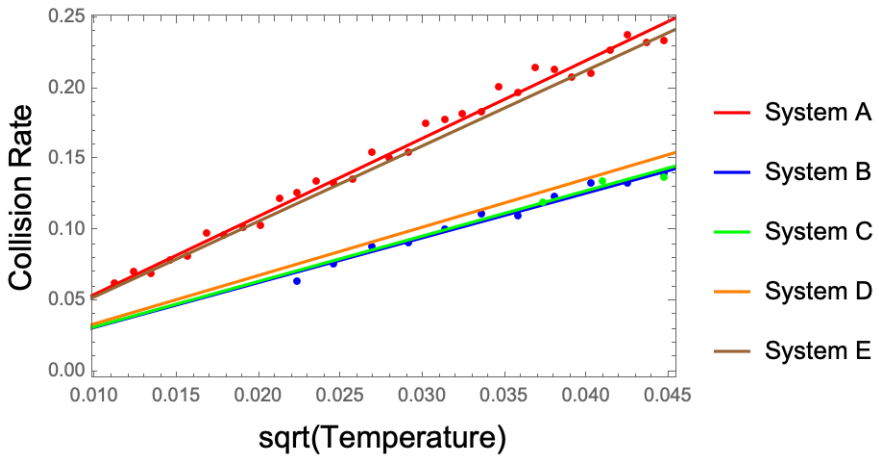
Out[ ]=



```
paramsweepA = ParallelMap[TempVsCRcoarsegrain[#, 1000, 900] &, Range[0.5, 2, 0.05]];
paramsweepB = ParallelMap[TempVsCRcoarsegrain[#, 250, 225] &, Range[0.5, 2, 0.05]];
paramsweepC = ParallelMap[TempVsCRcoarsegrain[#, 90, 81] &, Range[0.5, 2, 0.05]];
paramsweepD = ParallelMap[TempVsCRcoarsegrain[#, 40, 36] &, Range[0.5, 2, 0.05]];
paramsweepE = ParallelMap[TempVsCRcoarsegrain[#, 10, 9] &, Range[0.5, 2, 0.05]];
```

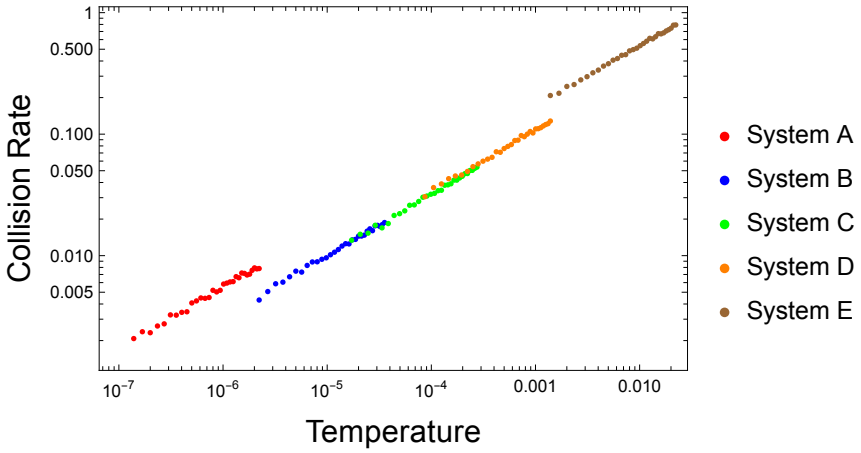
In[ ]:=
```
Show[
 ListPlot[{Sqrt[#[[1]]], (1 / 30) #[[2]]} & /@ paramsweepA,
  PlotStyle → Red, FrameLabel → {Style["sqrt(Temperature)", Black, 16],
    Style["Collision Rate", Black, 16]}, Frame → True],
 Plot[Evaluate[A x + b /. FindFit[{Sqrt[#[[1]]], #[[2]]} & /@ paramsweepA, 30 A x, {A, b}, x]],
  {x, 0, 2}, PlotStyle → Red, PlotLegends → LineLegend[{"System A"}]],
 ListPlot[{Sqrt[#[[1]]], (1 / 15) #[[2]]} & /@ paramsweepB, PlotStyle → Blue],
 Plot[Evaluate[A x + b /. FindFit[{Sqrt[#[[1]]], #[[2]]} & /@ paramsweepB, 15 A x, {A, b}, x]],
  {x, 0, 2}, PlotStyle → Blue, PlotLegends → LineLegend[{"System B"}]],
 ListPlot[{Sqrt[#[[1]]], (1 / 9) #[[2]]} & /@ paramsweepC, PlotStyle → Green],
 Plot[Evaluate[A x + b /. FindFit[{Sqrt[#[[1]]], #[[2]]} & /@ paramsweepC, 9 A x, {A, b}, x]],
  {x, 0, 2}, PlotStyle → Green, PlotLegends → LineLegend[{"System C"}]],
 ListPlot[{Sqrt[#[[1]]], (1 / 6) #[[2]]} & /@ paramsweepD, PlotStyle → Orange],
 Plot[Evaluate[A x + b /. FindFit[{Sqrt[#[[1]]], #[[2]]} & /@ paramsweepD, 6 A x, {A, b}, x]],
  {x, 0, 2}, PlotStyle → Orange, PlotLegends → LineLegend[{"System D"}]],
 ListPlot[{Sqrt[#[[1]]], (1 / 3) #[[2]]} & /@ paramsweepE, PlotStyle → Brown],
 Plot[Evaluate[A x + b /. FindFit[{Sqrt[#[[1]]], #[[2]]} & /@ paramsweepE, 3 A x, {A, b}, x]],
  {x, 0, 2}, PlotStyle → Brown, PlotLegends → LineLegend[{"System E"}]]
]
```

Out[ ]=



```
In[•]:= ListLogLogPlot[{(1 / 900) paramsweepA, (1 / 225) paramsweepB,
    (1 / 81) paramsweepC, (1 / 36) paramsweepD, (1 / 9) paramsweepE},
  PlotLegends → {"System A", "System B", "System C", "System D", "System E"},
  FrameLabel → {Style["Temperature", Black, 16], Style["Collision Rate", Black, 16]},
  PlotStyle → {Red, Blue, Green, Orange, Brown}, Frame → True]
```

Out[•]=

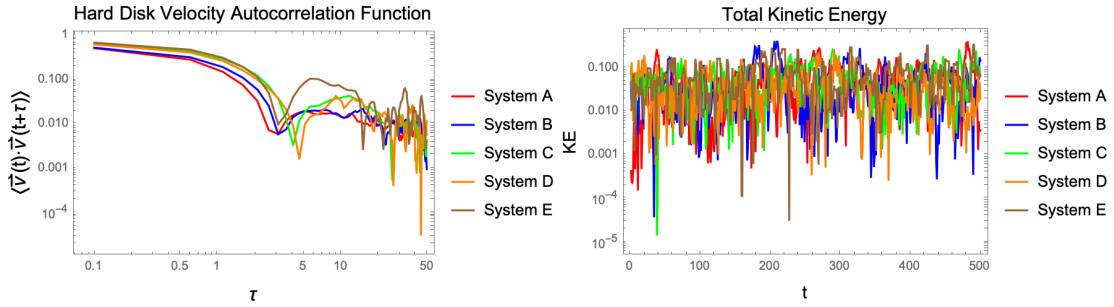


A very interesting phenomenon is observed: those different “coarse-grained” systems (we can think of the original system as a system with a coarse-grain degree of zero) all obey the ideal gas law with very similar fitting parameters (slope)! This indicates that the different coarse-grained systems can be interpreted as a surrogate for each other pertaining to the thermodynamics properties considering collision rate and temperature according to the ideal gas law. However, one may also argue that the short-time collective behavior of those different coarse-grained systems is still different even though their overall relationships are similar. Hence, we generate the following figures to show the velocity correlation (decay of particle velocities correlation w.r.t. time) and kinetic energy evolution w.r.t. time.

```
In[•]:= velosysA =
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Ball[{0, 0}, 1], 900], "BoxSize" → 10,
    "StepSize" → 0.5, "ParticleRadius" → 0.1, "ParticleMass" → 0.01, "Steps" → 500,
    "BoundaryCondition" → "Reflecting", "Output" → "VelocitiesByParticle"|>];

In[•]:= velosysB =
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Ball[{0, 0}, 1], 225], "BoxSize" → 10,
    "StepSize" → 0.5, "ParticleRadius" → 0.2, "ParticleMass" → 0.04, "Steps" → 500,
    "BoundaryCondition" → "Reflecting", "Output" → "VelocitiesByParticle"|>];

In[•]:= velosysC =
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Ball[{0, 0}, 1], 81], "BoxSize" → 10,
    "StepSize" → 0.5, "ParticleRadius" → 0.3, "ParticleMass" → 0.09, "Steps" → 500,
    "BoundaryCondition" → "Reflecting", "Output" → "VelocitiesByParticle"|>];
```

```
In[ ]:= velosysD =
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Ball[{0, 0}, 1], 36], "BoxSize" → 10,
    "StepSize" → 0.5, "ParticleRadius" → 0.5, "ParticleMass" → 0.25, "Steps" → 500,
    "BoundaryCondition" → "Reflecting", "Output" → "VelocitiesByParticle"|>];

In[ ]:= velosysE =
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Ball[{0, 0}, 1], 9], "BoxSize" → 10,
    "StepSize" → 0.5, "ParticleRadius" → 1, "ParticleMass" → 1, "Steps" → 500,
    "BoundaryCondition" → "Reflecting", "Output" → "VelocitiesByParticle"|>];

In[ ]:= Totalkineenerplot = ListLogPlot[{100 Total[Transpose[Mean[velosysA]^2]],
     (1 / 0.04) Total[Transpose[Mean[velosysB]^2]],
     (1 / 0.09) Total[Transpose[Mean[velosysC]^2]], (1 / 0.25)
      Total[Transpose[Mean[velosysD]^2]], Total[Transpose[Mean[velosysE]^2]]},
    PlotLegends → {"System A", "System B", "System C", "System D", "System E"},
    PlotStyle → {Red, Blue, Green, Orange, Brown},
    FrameLabel → Map[Style[#, Black, 14] &, {"t", "KE"}],
    PlotLabel → Style["Total Kinetic Energy", Black, 15],
    Joined → True, Frame → True, ImageSize → 350];

In[ ]:= Figplotvelocorr = ListLogLogPlot[{Abs[autocorlistsysA], Abs[autocorlistsysB],
     Abs[autocorlistsysC], Abs[autocorlistsysD], Abs[autocorlistsysE]},
    Frame → True, PlotStyle → {Red, Blue, Green, Orange, Brown},
    FrameLabel → Map[Style[#, Black, 16] &, {"τ", "⟨v⃗(t)·v⃗(t+τ)⟩"}],
    PlotLegends → {"System A", "System B", "System C", "System D", "System E"},
    PlotLabel → Style["Hard Disk Velocity Autocorrelation Function", Black, 15],
    Joined → True, ImageSize → 350];
  Linecomparevelocor = ListLinePlot[{Abs[autocorlistsysA], Abs[autocorlistsysB],
     Abs[autocorlistsysC], Abs[autocorlistsysD], Abs[autocorlistsysE]},
    PlotStyle → {Red, Blue, Green, Orange, Brown}, ImageSize → 350];

In[ ]:= Row[{Figplotvelocorr, Totalkineenerplot}]
```

Out[ ]=



Admittedly, it is intuitive that one may expect a large particle may have a slower velocity correlation than the smaller particles (If not in a coarse-grained system this may be true, e.g., with similar particle

numbers). However, the two left figures above show that all five different coarse-grained systems possess similar velocity correlation properties, indicating the collective particle behavior in the short-time regime is similar. Also, the total kinetic energy fluctuation w.r.t. time shows that the collective kinetic energies are preserved well for different coarse-grained systems tested, with no obvious differences detected.

Along with the collision rates, this shall be a shred of direct quantitative evidence showing that the spatial coarse-graining preserves physical properties for thermodynamically equilibrium systems.

Examine the Physics from the Causal Graph for Temporal Coarse-Graining

Now, we directly operate on the causal by cutting out edges for coarse-graining the causal graph. The way we propose to coarse-grain the causal graph is to uniformly sample the node that represent the collision events and cut off by assigned quantities. For example, if one has a list representing the node: {1,2,3,4,5,6,7,8,9,10}, assume this list stand for a causal graph representing a series of events with the ordered label, we can get the coarse-grained causal graph by simply extract by arithmetic order. Say one wants to get the coarse-grained graph preserving half the nodes, i.e., by the order of 2, the new graph writes: {1,3,5,7,9}. Similarly, if one wants to obtain a coarse-grained graph by the order of 3, one obtain a new graph {1,4,7,10}.

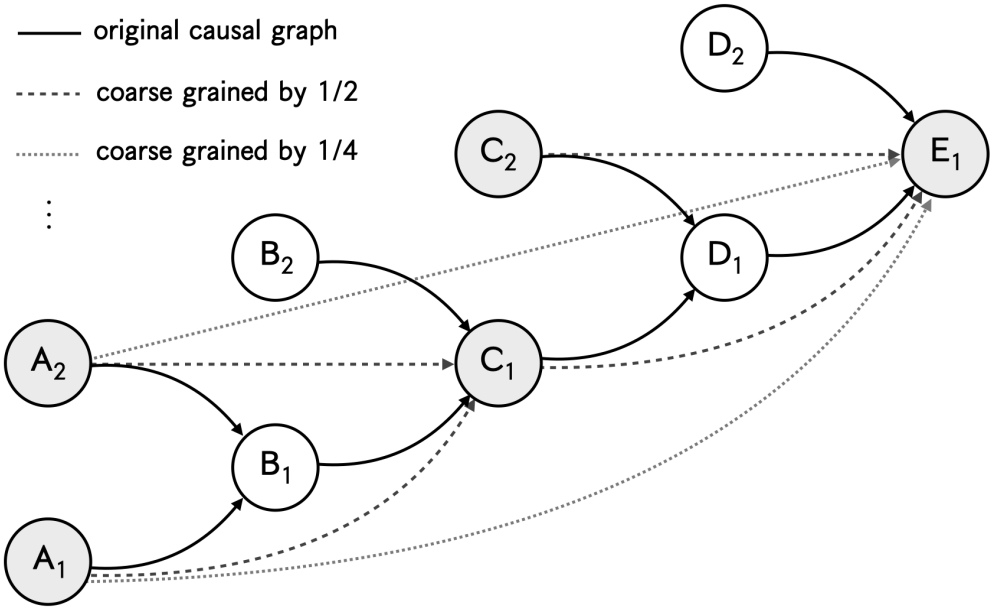


Figure 2 : Schematic of Temporal Coarse – Graining.

Now, the problem here would be if the node numbers in the graph are small, then by doing this kind of "coarse-graining" one cannot guarantee the new graph possesses exacting the information memory one hopes to keep. From our previous example by sampling differently the "order-3" coarse-grained graph contains either 3 or 4 nodes. Putting this in the general case of simulating hard sphere models one concludes that the collision events one stores in the new graph may contain unavoidable errors.

But with the increase of causal graph size this error with become relatively smaller until it eventually yields to zero.

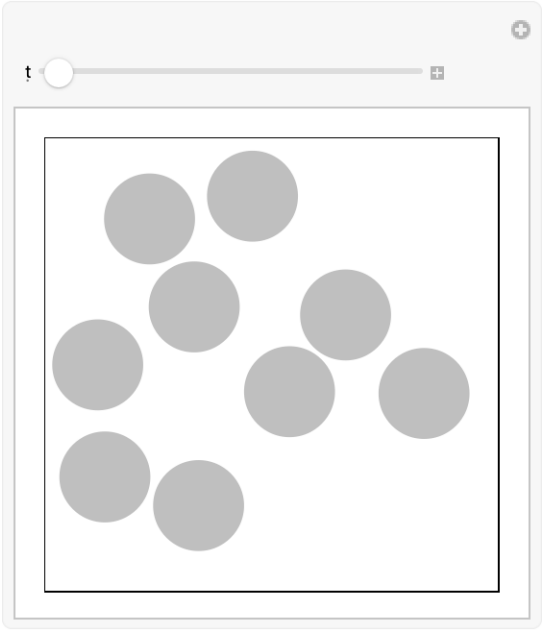
Here, we begin with a toy example to illustrate what this kind of coarse-graining looks like, and then we implement this coarse-grain technique to particle systems or larger scales. We will briefly elaborate on the pros and cons of such an approach and how this approach may further inspire a graph-based approach to better understand physics.

A Toy Example

We begin with the System A we just used and run for only 50 steps as it only contains a few collision events so we could better illustrate how the causal graph is coarse-grained.

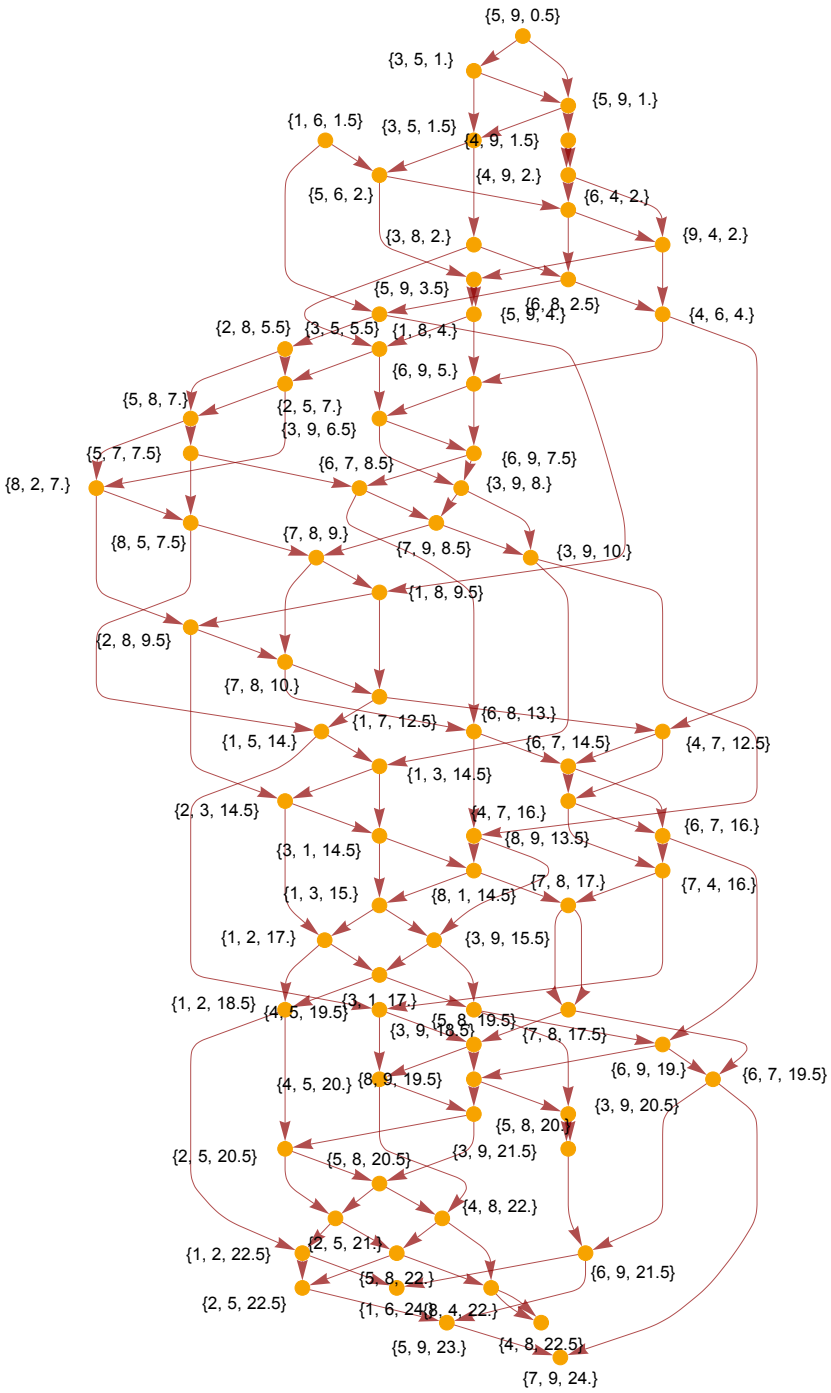
In[ ]:=
```
ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
  "Velocities" → RandomPoint[Sphere[{0, 0}], 9], "BoxSize" → 10,
  "StepSize" → 0.5, "ParticleRadius" → 1, "Steps" → 50,
  "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>]
```

Out[ ]=



In[ ]:=
```
toyexplcgcg = Graph[
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Sphere[{0, 0}], 9], "BoxSize" → 10,
    "StepSize" → 0.5, "ParticleRadius" → 1, "Steps" → 50,
    "BoundaryCondition" → "Reflecting", "Output" → "CausalGraph"|>], AspectRatio → 2]
```

Out[ ]=



The causal graph shows the collisions hidden beneath the toy example. The label on each node stands for the labels of the two particles involved in the collision and the third element is the corresponding

collision time. One can observe that on this causal graph, each node contains the label of different particles and the corresponding time at each collision. Just from the causal graph, one can directly compute the collision rate by counting the total number of nodes first and dividing by the simulation time. In order to coarse-grain this causal graph, we first assign a value of "1" at the end of each node element:

```
In[ ]:= toycgapd1 = Join[#, {1}] & /@ VertexList[toyexplcgcg];

In[ ]:= toycgapd1
Out[ ]=
{{1, 5, 3.5, 1}, {1, 6, 6.5, 1}, {1, 2, 7., 1}, {1, 6, 10.5, 1}, {1, 6, 11., 1},
 {1, 4, 11.5, 1}, {1, 4, 15.5, 1}, {4, 1, 15.5, 1}, {1, 6, 19., 1}, {1, 3, 21.5, 1},
 {1, 3, 22., 1}, {1, 3, 22.5, 1}, {2, 9, 0.5, 1}, {2, 5, 2.5, 1}, {2, 4, 3., 1},
 {2, 4, 3.5, 1}, {2, 3, 4., 1}, {2, 4, 4.5, 1}, {2, 4, 5., 1}, {2, 5, 6.5, 1},
 {2, 5, 7.5, 1}, {2, 5, 9.5, 1}, {2, 3, 13., 1}, {2, 5, 14., 1}, {2, 5, 15.5, 1},
 {2, 7, 16., 1}, {2, 3, 18., 1}, {2, 5, 18.5, 1}, {2, 5, 19., 1}, {2, 5, 20., 1},
 {3, 5, 2., 1}, {3, 5, 4., 1}, {3, 4, 6., 1}, {3, 4, 10., 1}, {3, 5, 11., 1},
 {3, 4, 12., 1}, {3, 4, 13.5, 1}, {3, 7, 15., 1}, {3, 4, 15.5, 1}, {3, 5, 21.5, 1},
 {3, 4, 22., 1}, {3, 6, 23., 1}, {3, 5, 24., 1}, {4, 5, 0.5, 1}, {4, 7, 2.5, 1},
 {4, 8, 3., 1}, {4, 9, 4.5, 1}, {4, 7, 5., 1}, {4, 7, 6.5, 1}, {4, 6, 12., 1},
 {4, 6, 14.5, 1}, {4, 8, 18., 1}, {4, 6, 18.5, 1}, {4, 7, 22.5, 1}, {5, 7, 12.5, 1},
 {5, 7, 13.5, 1}, {5, 7, 14., 1}, {5, 7, 17., 1}, {5, 7, 18.5, 1}, {6, 7, 3., 1},
 {6, 7, 9., 1}, {6, 8, 10., 1}, {6, 8, 18., 1}, {6, 8, 24.5, 1}, {7, 8, 1.5, 1},
 {7, 8, 2., 1}, {7, 8, 7., 1}, {9, 7, 7., 1}, {7, 9, 12.5, 1}, {7, 9, 13., 1},
 {7, 8, 14.5, 1}, {9, 7, 14.5, 1}, {7, 9, 16.5, 1}, {7, 9, 21.5, 1}, {8, 9, 1., 1},
 {8, 9, 2.5, 1}, {8, 9, 3.5, 1}, {8, 9, 19., 1}, {8, 9, 22.5, 1}, {8, 9, 23.5, 1}}
```

We can then count the total number of collisions by summing over the final element of each node:

```
In[ ]:= collitoyexpl = Total[toycgapd1[[All, 4]]]
Out[ ]=
77
```

We then do the coarse-grained sampling procedure on this causal graph, by the order of 2. We first extract the nodes in arithmetic order using the method we just mentioned and reassign the tale value as 2:

```
In[ ]:= toycgapd1coarsegrain = toycgapd1[[1 ;; -1 ;; 2]]; toycgapd1coarsegrain[[All, 4]] = 2;
```

We can then check the total number of collisions by summarizing the tale values:

```
In[ ]:= Total[toycgapd1coarsegrain[[All, 4]]]
Out[ ]=
78
```

The collision numbers deviate from the ground truth by 1, although sometimes they could be the same (as explained previously). We can then compute the collision rates of the two different causal graphs (that are expected to be similar since the total time is the same) to show how the thermodynamics quantities are preserved on the coarse-grained causal graph(s).

We start the computation by obtaining the final time of the whole collision process:

```
In[ ]:= finaltimetoyoriginal = Max[toycgapd1[[All, 3]]];
finaltimetoycoarsegrain = Max[toycgapd1coarsegrain[[All, 3]]];
Row[finaltimetoyoriginal, finaltimetoycoarsegrain]
```

```
Out[ ]=
Row[24., 24.]
```

The collision rate for both the two systems are then:

```
In[ ]:= colliratetoyoriginal = Total[toycgapd1[[All, 4]]] / finaltimetoyoriginal;
colliratetoycoarsegrain =
 Total[toycgapd1coarsegrain[[All, 4]]] / finaltimetoycoarsegrain;
Row[colliratetoyoriginal, colliratetoycoarsegrain]
```

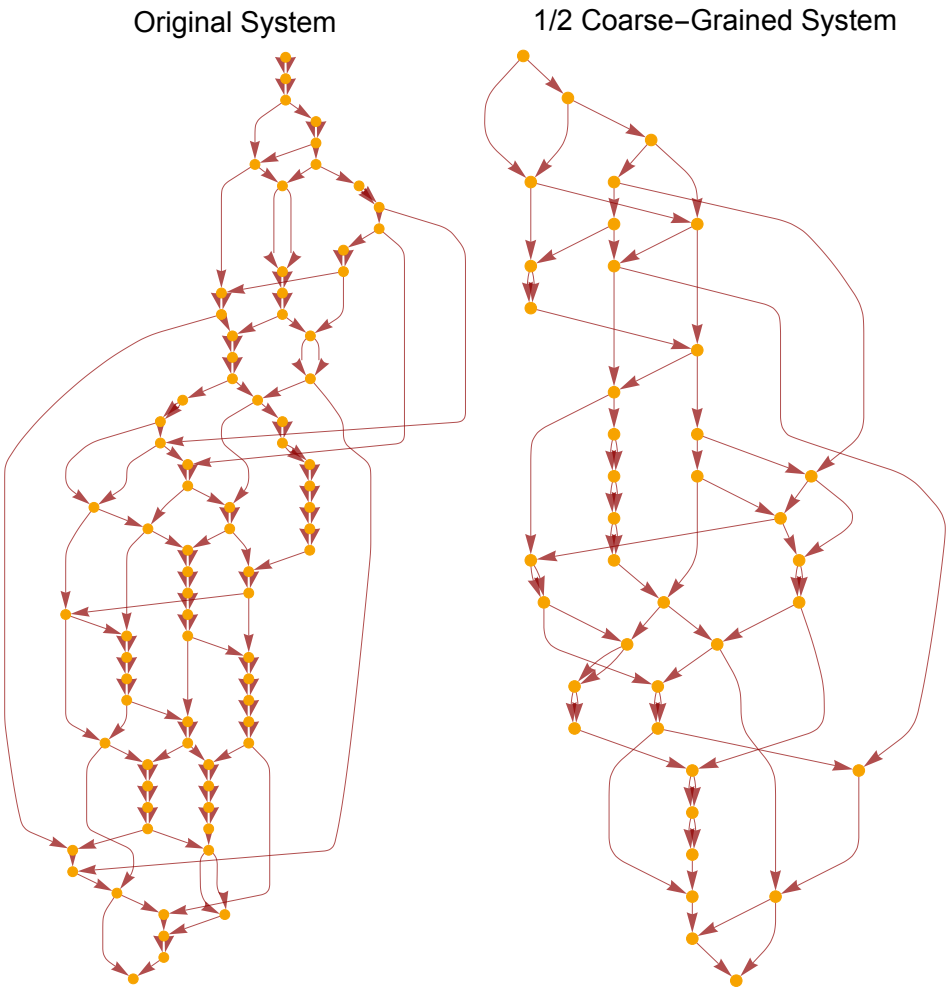
```
Out[ ]=
Row[3.20833, 3.25]
```

The collision rate results deviate a little because the two final times of the two systems are different caused by our sampling. However, these collision rates are expected to be the same under larger systems since the deviation of the time would not matter to the total collisions as much compared to the toy case here with total collisions of only 80. To better understand what we did on the toy system, we visualize the graph from the new collisions compared with the old one.

```
In[ ]:= CollisionsToCausalGraph[colls_] := Module[{particles},
  particles = Union @@ colls[[All, ;; 2]];
  Graph[
   Catenate[
    (* Extract all collisions involving particle p. Sort them by time,
    then connect them sequentially with directed edges.*)
    Function[p,
      Module[{sortedcolls},
       sortedcolls = SortBy[Select[colls, #[[1]] == p || #[[2]] == p &], Last];
       Table[sortedcolls[[n]] → sortedcolls[[n + 1]], {n, Length[sortedcolls] - 1}]]
     ] /@ particles](*Apply to all particles*)
   , VertexLabels → "Name", GraphLayout → "LayeredDigraphEmbedding",
   AspectRatio → 1/GoldenRatio, ImageSize → Large,
   EdgeStyle → {Hue[0, 1, 0.56]}, VertexStyle →
    {Directive[Hue[0.11, 1, 0.97], EdgeForm[{Hue[0.11, 1, 0.97], Opacity[1]}]]}]
 ]
```

In[ ]:=
```
Row[{Graph[CollisionsToCausalGraph[toycgapd1], AspectRatio → 2, VertexLabels → None,
    ImageSize → 250, PlotLabel → Style["Original System", Black, 15]],
  Graph[CollisionsToCausalGraph[toycgapd1coarsegrain],
    AspectRatio → 2, VertexLabels → None, ImageSize → 250,
    PlotLabel → Style["1/2 Coarse-Grained System", Black, 15]]}]
```

Out[ ]=



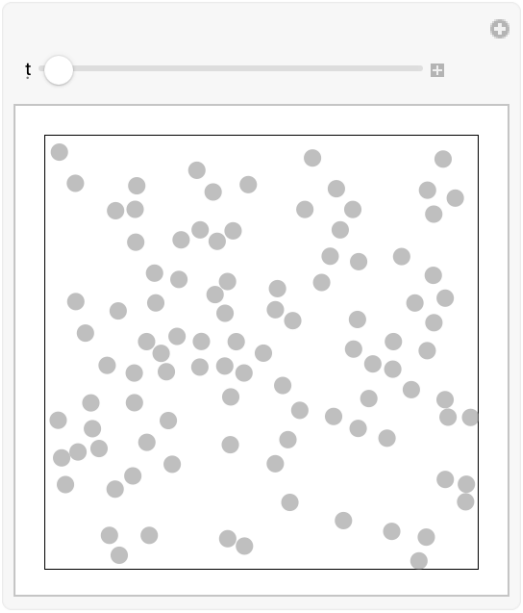
Practice on Coarse-Graining on Causal Graph for Large Scale Systems

Now we will use a thermodynamical equilibrium system (large system with long time steps) to conduct the causal graph coarse-grain based on our general formulation shown above. We begin with a system of 100 particles running for 5000 time steps and do coarse-graining on its corresponding causal graph.

In[ ]:=
```
practicesyscoarsegrain =
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Ball[{0, 0}, 1], 100], "BoxSize" → 10,
    "StepSize" → 0.5, "ParticleRadius" → 0.2, "Steps" → 1000,
    "BoundaryCondition" → "Reflecting", "Output" → "Visualize"|>];
```

In[ ]:= **practicesyscoarsegrain**

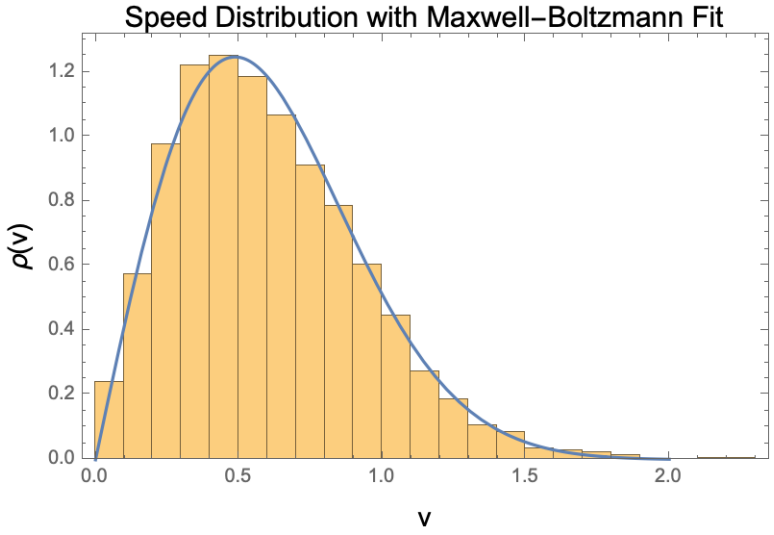
Out[ ]=



We check the systems reach equilibrium by testing its corresponding Maxwell-Boltzmann fit:

In[ ]:=
```
Module[{speeds, Temp}, {speeds, Temp} = {Flatten[#[[1]]], First[#[[2]]]} &@
    ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
      "Velocities" → RandomPoint[Ball[{0, 0}, 1], 100], "BoxSize" → 10,
      "StepSize" → 0.5, "ParticleRadius" → 0.2, "Steps" → 500, "BoundaryCondition" →
       "Reflecting", "Output" → {"SpeedsByTime", "Temperature"}|>];
   Show[(*Empirical histogram of speeds*)Histogram[speeds, {0.1`}, "PDF",
    Frame → True, FrameLabel → (Style[#1, Black, 14] &) /@ {"v", "ρ(v)"},
    PlotLabel → Style["Speed Distribution with Maxwell-Boltzmann Fit", Black, 15],
    ImageSize → 400], (*Plot of 2D Maxwell-Boltzmann distribution with best-
    fit temperature.*)Plot[1 / Temp v E^(-(v^2 / (2 Temp))), {v, 0, 2}]]]
```

Out[ ]=



It can be deduced that this is an equilibrium system. We can hence operate the coarse-graining procedure we did on the toy example to this system. The key is due to the large size of this system, we may not be able to directly visualize the causal graph from the simulation. But we can examine the statistics of the causal graph, which is both mathematical and physically more interesting to understand this “sample and filter” approach.

In[ ]:=
```
practicesyscoarsegrain =
  ResourceFunction["HardSphereSimulation"][<|"Positions" → "RandomNonOverlapping",
    "Velocities" → RandomPoint[Ball[{0, 0}, 1], 100], "BoxSize" → 10,
    "StepSize" → 0.5, "ParticleRadius" → 0.2, "Steps" → 500,
    "BoundaryCondition" → "Reflecting", "Output" → "CausalGraph"|>];
```

We then do the coarse grain on our causal graph with five different degrees, just as what we did for the spatial coarse-grain:

In[ ]:=
```
practicesyscoarsegrain = Join[#, {1}] & /@ VertexList[practicesyscoarsegrain];
practicecoarsegrainA = practicesyscoarsegrain;
practicecoarsegrainB = practicesyscoarsegrain[[1 ;; -1 ;; 2]];
practicecoarsegrainB[[All, 4]] = 2;
practicecoarsegrainC = practicecoarsegrainB[[1 ;; -1 ;; 2]];
practicecoarsegrainC[[All, 4]] = 4;
practicecoarsegrainD = practicecoarsegrainC[[1 ;; -1 ;; 2]];
practicecoarsegrainD[[All, 4]] = 8;
practicecoarsegrainE = practicecoarsegrainD[[1 ;; -1 ;; 2]];
practicecoarsegrainE[[All, 4]] = 16;
```

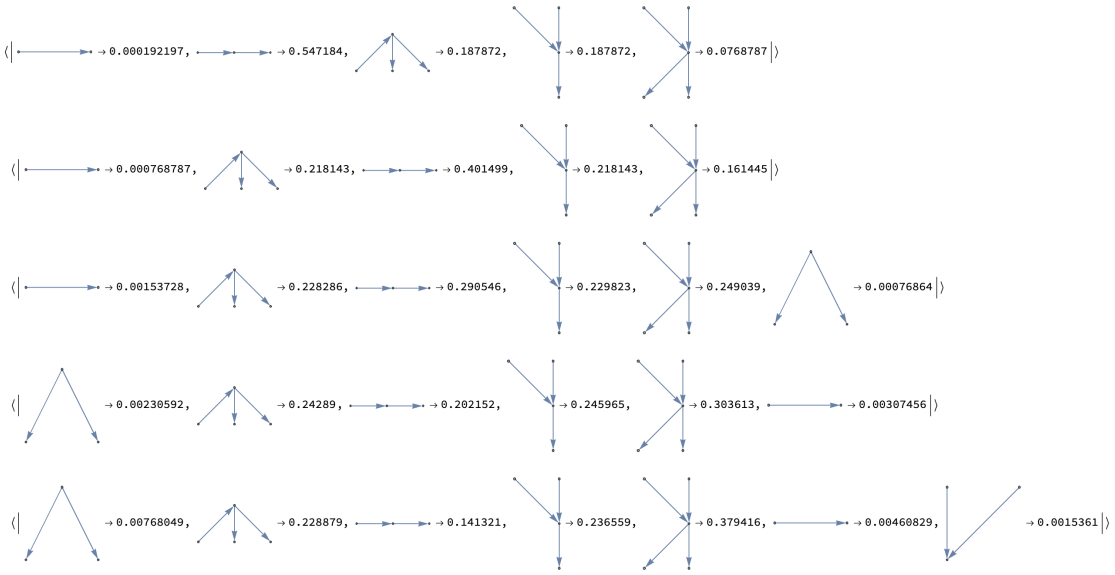
In[ ]:=
```
Dimensions[practicecoarsegrainE]
```

Out[ ]=
```
{651, 4}
```

In[ ]:=
```
probplotsysA = AssociationThread[Keys[RGUsysA] → probdistsysA];
probplotsysB = AssociationThread[Keys[RGUsysB] → probdistsysB];
probplotsysC = AssociationThread[Keys[RGUsysC] → probdistsysC];
probplotsysD = AssociationThread[Keys[RGUsysD] → probdistsysD];
probplotsysE = AssociationThread[Keys[RGUsysE] → probdistsysE];
```

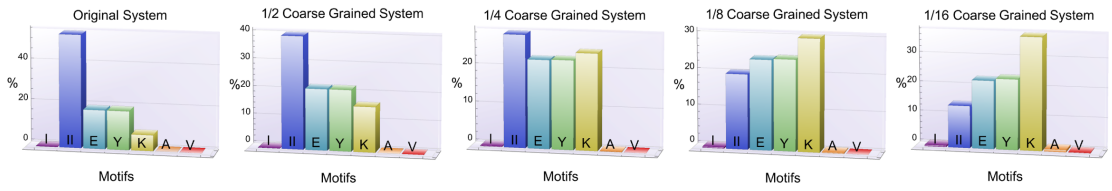
In[ ]:=
```
Row[{probplotsysA, probplotsysB, probplotsysC, probplotsysD, probplotsysE}]
(*Row[{RGUsysA,RGUsysB,RGUsysC,RGUsysD,RGUsysE}]*)
```

Out[ ]=



In[ ]:=
```
distrimotifsysA = BarChart3D[
  {100 {0.00019219680953296174`, 0.5471843167403421`, 0.18787238131847012`,
     0.18787238131847012`, 0.0768787238131847`, 0, 0}},
  ChartLabels → (Style[#1, Black, 15] &) /@ {"I", "II", "E", "Y", "K", "A", "V"},
  AxesLabel → (Style[#1, Black, 15] &) /@ {"Motifs", "", "%"}, ChartStyle → "Rainbow",
  PlotLabel → Style["Original System", Black, 15], ImageSize → 250];
distrimotifsysB = BarChart3D[{100 {0.000768787238131847`, 0.4014991351143571`,
     0.21814337881991158`, 0.21814337881991158`, 0.16144532000768788`, 0, 0}},
  ChartLabels → (Style[#1, Black, 15] &) /@ {"I", "II", "E", "Y", "K", "A", "V"},
  AxesLabel → (Style[#1, Black, 15] &) /@ {"Motifs", "", "%"}, ChartStyle → "Rainbow",
  PlotLabel → Style["1/2 Coarse Grained System", Black, 15], ImageSize → 250];
distrimotifsysC = BarChart3D[
  {100 {0.0015372790161414297`, 0.2905457340507302`, 0.2282859338970023`,
     0.22982321291314373`, 0.2490392006149116`, 0.0007686395080707148`, 0}},
  ChartLabels → (Style[#1, Black, 15] &) /@ {"I", "II", "E", "Y", "K", "A", "V"},
  AxesLabel → (Style[#1, Black, 15] &) /@ {"Motifs", "", "%"}, ChartStyle → "Rainbow",
  PlotLabel → Style["1/4 Coarse Grained System", Black, 15], ImageSize → 250];
distrimotifsysD = BarChart3D[
  {100 {0.0030745580322828594`, 0.202152190622598`, 0.24289008455034589`,
     0.24596464258262873`, 0.30361260568793236`, 0.0023059185242121443`, 0}},
  ChartLabels → (Style[#1, Black, 15] &) /@ {"I", "II", "E", "Y", "K", "A", "V"},
  AxesLabel → (Style[#1, Black, 15] &) /@ {"Motifs", "", "%"}, ChartStyle → "Rainbow",
  PlotLabel → Style["1/8 Coarse Grained System", Black, 15], ImageSize → 250];
distrimotifsysE = BarChart3D[{100 {0.004608294930875576`,
     0.141321044546851`, 0.22887864823348694`, 0.23655913978494625`,
     0.3794162826420891`, 0.007680491551459293`, 0.0015360983102918587`}},
  ChartLabels → (Style[#1, Black, 15] &) /@ {"I", "II", "E", "Y", "K", "A", "V"},
  AxesLabel → (Style[#1, Black, 15] &) /@ {"Motifs", "", "%"}, ChartStyle → "Rainbow",
  PlotLabel → Style["1/16 Coarse Grained System", Black, 15], ImageSize → 250];
Row[{distrimotifsysA, distrimotifsysB,
  distrimotifsysC, distrimotifsysD, distrimotifsysE}]
```

Out[ ]=



Concluding remarks

In this project, we strive the understand a commonly used approach in multiscale mode as a bottom-up strategy, coarse-graining. Under our context of particle-based simulations, we propose and define coarse-grain as enlarging the particle radii and simultaneously reducing the particle numbers hoping

to pertain constant volume fraction to reduce computational costs. We examine the hypothesis, assumptions, and results from numerical experiments for coarse-graining from designed benchmark cases. The goal is to answer key questions from (1) Whether the order reduction coarse-graining approach can be applied to the hard-sphere gas model without rearranging the forcefield. (2) If we can adopt such an approach, what are the limitations? and (3) How to understand such an approach in the context of the Wolfram Physics Project with graph models? In our effort to answer those questions, we obtain the following main conclusions and bring a deeper understanding of such a subject to the physics communities:

1. By employing the order reduction coarse-graining method to hard sphere gas models, one finds the global density distribution is observed to be similar, yet the local density profile is totally different as an observation from the designed particle collision experiments.
2. The assumed coarse-grained systems possess a similar hypergraph structure (with a similar adjacency matrix) under an equilibrium state. The causal graphs of the two systems also possess similar adjacency matrix structures, with similar but slightly different motif distributions within the causal graphs. Note that it is not sufficient for one to determine the coarse-grainability just from the adjacency matrices of the hyper-and-causal graphs since the observations were carried out under prior assumptions.
3. We observe that for different coarse-grained systems, the relationships between the collision rate w.r.t. temperature approximately remains the same, leading to the conclusion of different coarse-grained systems can be viewed as surrogates of each other under the ideal gas law.
4. The velocity correlation function and total kinetic energy fluctuation w.r.t. time are observed to not have evident differences with similar trends for different coarse-grained systems, indicating the collective behavior of particles can be viewed as equivalent in the consistency of time.
5. We propose a novel technique to directly conduct coarse-graining on the causal graphs, by uniformly sampling over the nodes and reassigning the collision properties to retain the same collision rate. Under this effort, we observe that a thermodynamic equilibrium system possesses similar motif probability distribution, with an increasing ratio of branchial motifs (e.g., K-, Y-, and E-typed), indicating that the local structure slightly changes during our causal graph coarse-graining approach.

Based on the work we have conducted by far, we not only answer our initial questions proposed in the introduction section but also provide answers for three more new questions: (4) What are the criteria for measuring the coarse-grainability of hard sphere gas systems that can connect with graph models? This question can be the follow-up to both questions (1) and (2), with a measurable collision rate and temperature that bridges real physics to causal graphs. (5) How can we do coarse-grain without operating on the physical systems? The answer would be the temporal coarse-graining on the causal graph we proposed; and (6) Does this causal graph coarse-graining change the local structures of the graph? The answer would be yes but the general distribution of the motifs is still observed to be similar.

Based on the work we have already done, several future works can be drawn to make this research more concrete and interesting:

1. Comparing the coarse-grain approach with the representative volume element (RVE) approach by switching the new boundary conditions of the larger particle simulations to periodic and seeing the difference between coarse-graining and unit-based modeling.
2. Conduct graph foliations on for a more rigorous and explainable causal graph coarse-graining.
3. Elicit new properties preserved in the causal graph that can retain the probability distribution of the motifs in the causal graph when conducting temporal coarse-graining.

Keywords

- Hard Sphere Simulations
- Statistical Mechanics
- Coarse Graining
- Multiscale Modeling
- Causal Graph
- Thermodynamics
- Computational Physics
- Computational Irreducibility
- Model Reduction

Acknowledgement

I would like to thank my mentors Sotiris Michos, Nik Murzin, and James Boyd for constantly providing me with technical guidance, feedback, support, and advice on this project. I would also like to thank Matt Kafker, the developer of hard sphere simulation models in Wolfram language, for his selfless contribution to this project, especially for his key ideas and insight and for taking the time to provide me with tutorials and guidance. I would also like to Stephen Wolfram, for assigning me this project, providing his unique insights on multiscale modeling, and for the amazing discussions during the Winter School. I would also like to thank Zach Shelton, for providing computational and technical support in Wolfram technologies through the Wolfram Student Ambassador Program. The amazing research talks and discussions held during the Winter School lead me to the exciting projects and works conducted at Wolfram Research.

References

- [Thoennessen, 1995] Michael Thoennessen. Gas Laws and Kinetic Theory. URL: https://people.nscl.msu.edu/~thoennes/personal/phy231/p231_notes6.pdf.

- [Wolfram, 2022] Stephen Wolfram. On the Concept of Motion. URL: https://writings.stephenwolfram.com/2022/03/on-the-concept-of-motion/
- [Kafker, 2020] Matt Kafker. [WSS20] Exploring Networks in Hard Sphere Models. URL: https://community.wolfram.com/groups/-/m/t/2028871
- [Widartiningsih et al., 2020] Putri Mustika Widartiningsih et al. Coarse graining DEM simulations of a powder die-filling system. Powder Technology. URL: https://doi.org/10.1016/j.powtec.2020.05.063
- [Kanjilal & Schneiderbauer, 2021] Suranita Kanjilal and Simon Schneiderbauer. A revised coarse-graining approach for simulation of highly poly-disperse granular flows. Powder Technology. URL: https://doi.org/10.1016/j.powtec.2021.02.015
- [Lu et al., 2016] Liqiang Lu et al. Coarse-Grained-Particle Method for Simulation of Liquid–Solids Reacting Flows. Ind. Eng. Chem. Res. URL: https://doi.org/10.1021/acs.iecr.6b02688
- [Wolfram, 2002] Stephen Wolfram. A New Kind of Science. URL: https://www.wolframscience.com/nks/

Appendix

Supplementary Codes

Macro-dynamics Analysis

Examine the Local Dynamics From Equilibrium Process

Exploring the Motifs of Temporal Causal Graph Coarse-Graining

In[12]:=
```
CloudPublish[EvaluationNotebook[],
 Information[EvaluationNotebook[], "WindowTitle"]];
```