Introduction to Dynamical Particle Simulations

DISCRETE ELEMENT METHOD

Basic Theory and Algorithms

Daniel Queteschiner*

Christoph Kloss**

***daniel.queteschiner@gmx.at**

****christoph.kloss@cfdem.com**

Discrete Element Method
Principles

- DEM manages information about **each individual particle** (mass, velocity, ...) and the forces acting on it.
- Each particle is tracked in **Lagrangian Frame**, the force balance

  $$m_p \ddot{\vec{x}}_p = \sum_i \vec{F}_i$$

  is integrated using an appropriate integration scheme.
- DEM can take into account the particle‘s shape

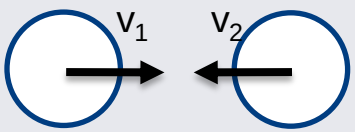
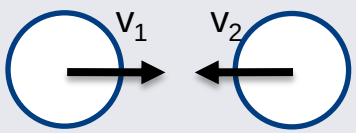
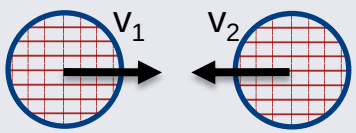
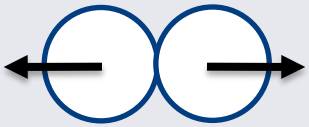
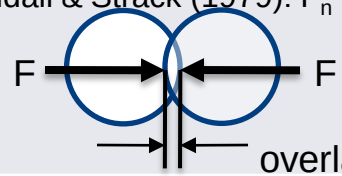
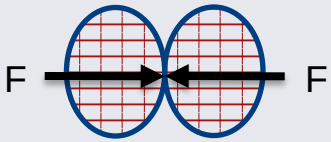
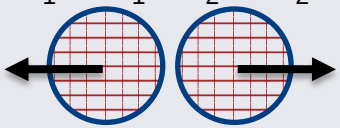
Examples of forces $\vec{F}_i$ that can be included:

- Contact forces (particle-particle, particle-wall)
- Gravity
- Fluid drag force

Discrete Element Method

Types

Rigid particle | **Deformable particle**

Example: Normal force	Hard Particle approach	Soft Particle (Classical DEM)	Finite Discrete Element Method
Before impact			
At/During impact	$v_1^* = -e\, v_1$ $v_2^* = -e\, v_2$  e...coefficient of restitution	Hertz: $F_n \sim \delta^{3/2}k$ Cundall & Strack (1979): $F_n \sim \delta k$ 	 F from FEM
After impact	$v_1^* < v_1$ $v_2^* < v_2$ 	$v_1^* < v_1$ $v_2^* < v_2$ 	$v_1^* < v_1$ $v_2^* < v_2$ 

Discrete Element Method
Time Integration (1)

Several different integration schemes are available

- **Euler** integration (1st order)
- **Leapfrog** integration (2nd order)
- **Verlet** integration (2nd order)

Higher-order / multi-timestep integrators

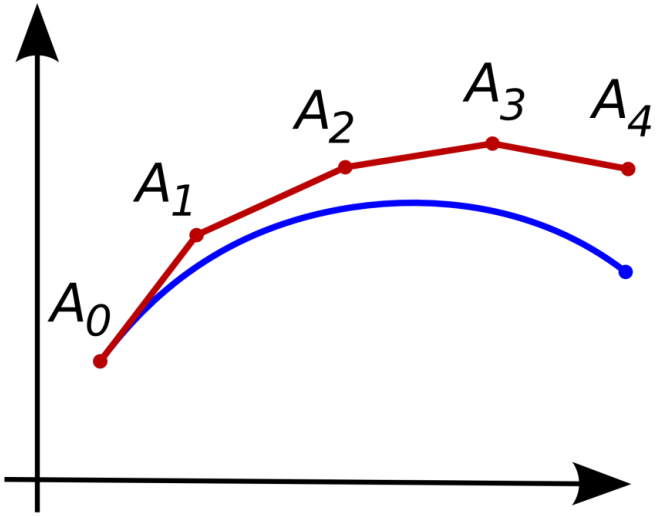
- **Respa**
- **Gear** integration (can be of 2nd, 3rd, 4th, 5th order)

Discrete Element Method
Time Integration (2)

Euler integration (1st order)

$$\vec{x}(t + \Delta t) = \vec{x}(t) + \vec{v}(t)\Delta t$$

$$\vec{v}(t + \Delta t) = \vec{v}(t) + \vec{a}(t)\Delta t$$

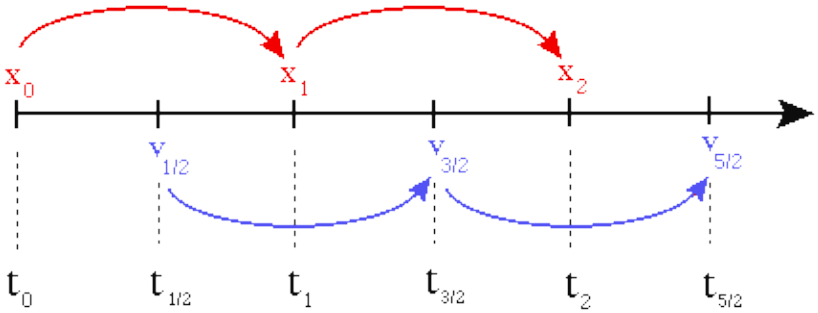


Leapfrog integration (2nd order)

$$\vec{x}(t + \Delta t) = \vec{x}(t) + \vec{v}(t - \tfrac{1}{2}\Delta t)\Delta t$$

$$\vec{a}(t) = \vec{F}(\vec{x}(t))$$

$$\vec{v}(t + \tfrac{1}{2}\Delta t) = \vec{v}(t - \tfrac{1}{2}\Delta t) + \vec{a}(t)\Delta t$$



Discrete Element Method
Time Integration (3)

Verlet integration

With a Taylor series, one finds:

$$\vec{x}(t + \Delta t) = \vec{x}(t) + \vec{v}(t)\Delta t + \tfrac{1}{2}\,\vec{a}(t)\Delta t^2 + \tfrac{1}{6}\,\vec{b}(t)\Delta t^3 + O(\Delta t^4)$$

$$\vec{x}(t - \Delta t) = \vec{x}(t) - \vec{v}(t)\Delta t + \tfrac{1}{2}\,\vec{a}(t)\Delta t^2 - \tfrac{1}{6}\,\vec{b}(t)\Delta t^3 + O(\Delta t^4)$$

which yields:

$$\vec{x}(t + \Delta t) = 2\vec{x}(t) - \vec{x}(t - \Delta t) + \vec{a}(t)\Delta t^2 + O(\Delta t^4)$$

For the first time-step, since $\vec{x}(-\Delta t)$ is unknown, one uses:

$$\vec{x}(\Delta t) \approx \vec{x}(0) + \vec{v}(0)\Delta t + \tfrac{1}{2}\,\vec{a}(0)\Delta t^2 + O(\Delta t^3)$$

Discrete Element Method
Time Integration (4)

Velocity Verlet integration

Tracking of particle **position and velocity**, again using Taylor expansion:

$$\vec{x}(t + \Delta t) = \vec{x}(t) + \vec{v}(t)\Delta t + \tfrac{1}{2}\,\vec{a}(t)\Delta t^2$$

$$\vec{v}(t + \Delta t) = \vec{v}(t) + \tfrac{1}{2}\{\vec{a}(t) + \vec{a}(t + \Delta t)\}\Delta t$$

Discrete Element Method
Time Integration (5)

Velocity Verlet integration

The standard implementation scheme of this algorithm is:

1. Calculate: $\vec{v}(t + ½\Delta t) = \vec{v}(t) + ½\, \vec{a}(t)\Delta t$

2. Calculate: $\vec{x}(t + \Delta t) = \vec{x}(t) + \vec{v}(t + ½\Delta t)\Delta t$

3. Derive $\vec{a}(t + \Delta t)$ from the interaction (e.g. particle collision force).

4. Calculate: $\vec{v}(t + \Delta t) = \vec{v}(t + ½\Delta t) + ½\, \vec{a}(t + \Delta t)\Delta t$

Commonly used in LIGGGHTS

Discrete Element Method
Time Integration (6)

Velocity Verlet has the following attractive properties:

- Only terms differing one order in Δt at maximum are combined. (machine precision)
- It is time reversible (if no dissipation is present).
- It is symplectic, i.e. it does not violate Liouville's theorem. (This means it conserves phase-space density)
- It is easy to implement and memory efficient

Integrators such as standard **Runge-Kutta** are **not symplectic**, and are thus less frequently used for Molecular Dynamics / DEM.

Discrete Element Method

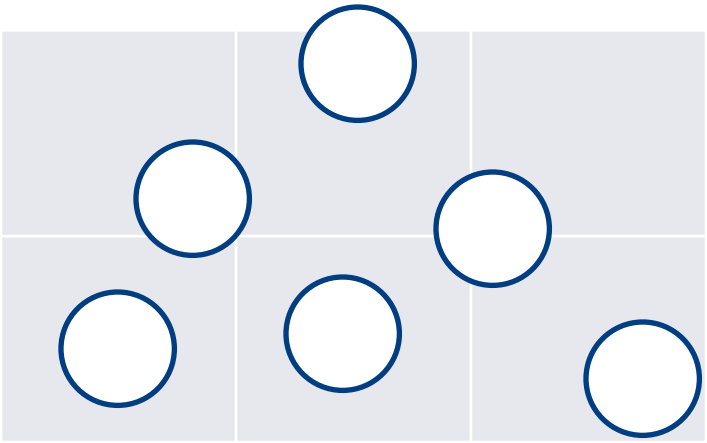
Contact Detection (1)

- Impractical to check each pair of particles for possible contacts ($O(n^2)$ runtime behaviour)

- Use a grid based structure to exclude potential partners
  ➔ $O(n)$ runtime behaviour

- Still, detection of contact partners remains bottle-neck of DEM

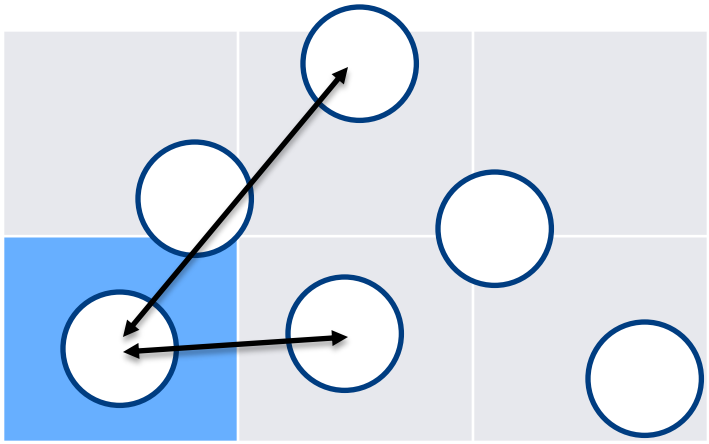
- Simulation results are not grid-dependent, but runtime is



Discrete Element Method
Contact Detection (2)

Example for contact detection algorithm

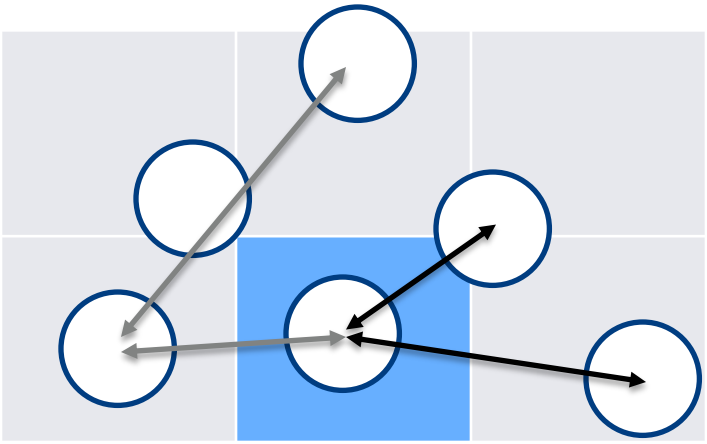
Check all right and bottom neighbour cells of the cell a particle has its centre in.



Discrete Element Method
Contact Detection (2)

Example for contact detection algorithm

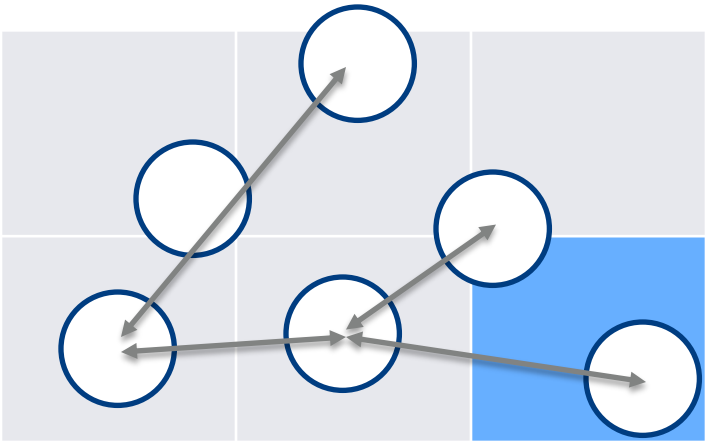
Check all right and bottom neighbour cells of the cell a particle has its centre in.



Discrete Element Method
Contact Detection (2)

Example for contact detection algorithm

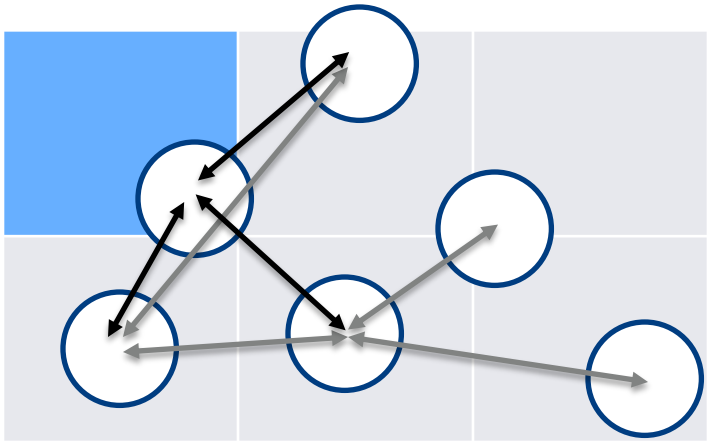
Check all right and bottom neighbour cells of the cell a particle has its centre in.



Discrete Element Method
Contact Detection (2)

Example for contact detection algorithm

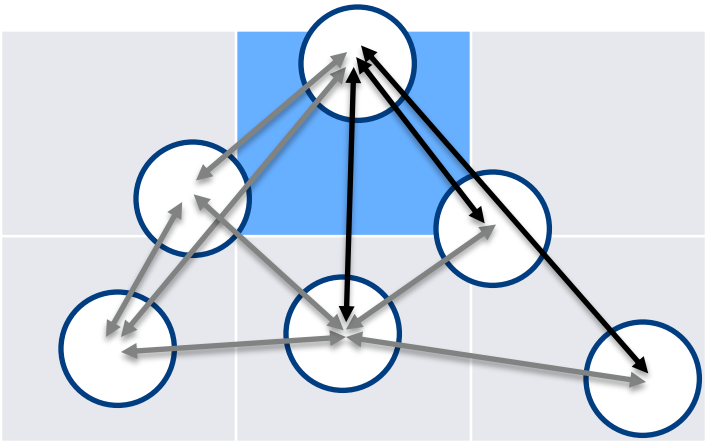
Check all right and bottom neighbour cells of the cell a particle has its centre in.



Discrete Element Method
Contact Detection (2)

Example for contact detection algorithm

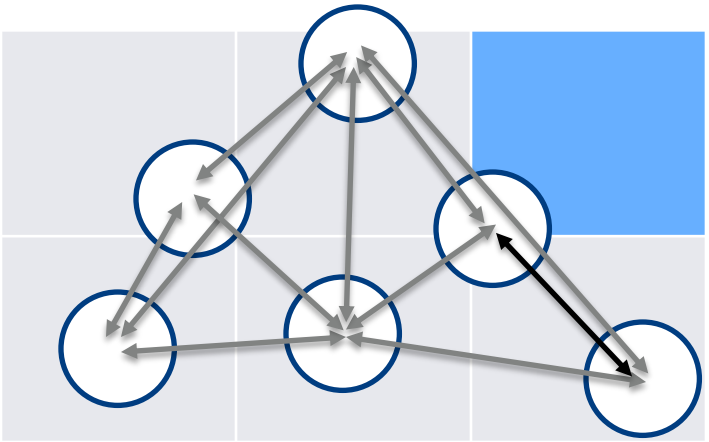
Check all right and bottom neighbour cells of the cell a particle has its centre in.



Discrete Element Method
Contact Detection (2)

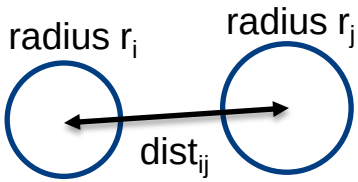
Example for contact detection algorithm

Check all right and bottom neighbour cells of the cell a particle has its centre in.



Discrete Element Method
Contact Detection (3)

Neighbour-Lists or **Verlet-Lists** (Verlet, 1967)



- Pairs of particles p_i, p_j are added to a "neighbour-list" if $dist_{ij} < r_i + r_j + c$ c ... skin parameter

- Collision detection is based on this list for the next N_{Verlet} time-steps (Walther, 2009)

  $N_{Verlet} = c / (2\, v_{max}\, \Delta t)$

Usually, a **combined approach of grid decomposition and Verlet-Lists** is used. The grid spacing and the Verlet parameter c are optimized to get a fast algorithm.

Introduction to Dynamical Particle Simulations

LIGGGHTS

LAMMPS Improved for General Granular and Granular Heat Transfer Simulations

Daniel Queteschiner*

Christoph Kloss**

***daniel.queteschiner@gmx.at**

****christoph.kloss@cfdem.com**

The MD Code LAMMPS

LIGGGHTS

LAMMPS - Open Source MD + DEM

- **LAMMPS** =
  Large Atomic and Molecular Massively Parallel Simulator

- **OpenSource under GPL**, provided by **Sandia National Laboratories** since the mid 90‘s (http://lammps.sandia.gov/)

- **Widely used** (over **500 journal publications** 2000-2009 using LAMMPS) see http://lammps.sandia.gov/papers.html

- LAMMPS has potentials for soft materials (biomolecules, polymers), solid-state materials (metals, semiconductors) and **coarse-grained systems**. It can be used to model atoms or, more generically, as a **parallel particle simulator** at the **atomic, meso, or continuum scale**.

- LAMMPS is a **C++** code, it runs on single processors or in parallel using **message-passing** techniques and a **spatial-decomposition of the simulation domain**. The code is designed to be **easy to modify or extend** with new functionality.

- It is **very fast** and also used on huge clusters (e.g. on Sandia Red Storm with 16k Quadcore nodes, simulations with 2 billion particles performed)

LIGGGHTS
Why take LAMMPS?

The strengths of LAMMPS

- LAMMPS is **fast** and has a scope for **massively parallel computing**
- The LAMMPS **documentation is good**
- LAMMPS has a **large user community**
- LAMMPS is **easy to use** (good scripting language)
- LAMMPS **source code is easy to read, understand and modify**
- **GPU acceleration** efforts are underway right now
- LAMMPS offers a great **coupling interface**

LIGGGHTS
The CFDEM Project

What does Open Source Software stand for?

- Anyone is free to use, modify, or extend, including commercial purpose
- If you distribute a modified version, it must remain open-source

LAMMPS
established MD code

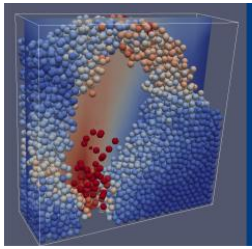
OpenFOAM®
established CFD code

LIGGGHTS

LAMMPS IMPROVED FOR GENERAL GRANULAR AND GRANULAR HEAT TRANSFER SIMULATIONS

CFDEM coupling

COMPUTATIONAL FLUID DYNAMICS COUPLED TO DISCRETE ELEMENT METHOD



CFDEM project

www.cfdem.com

Core developers

Christoph Kloss
LIGGGHTS

Christoph Goniva
CFDEM Coupling

Getting And Installing LIGGGHTS

LIGGGHTS
Installed In 5 Minutes

LIGGGHTS installation from Git repository

- Open a terminal type (using https protocol)

  ```
  git clone https://github.com/CFDEMproject/LIGGGHTS-PUBLIC.git
  ```

  or (using git protocol)

  ```
  git clone git@github.com:CFDEMproject/LIGGGHTS-PUBLIC.git
  ```

- Change to the LIGGGHTS-PUBLIC/src folder and type:

  ```
  make fedora
  ```

  to compile (OpenMPI installation required).

- This will create an executable called

  ```
  lmp_fedora
  ```

- To start a simulation, you need the executable and an input script

  ```
  lmp_fedora < in.example
  ```

LIGGGHTS
Installed In 5 Minutes

For Post-Processing: LPP/pizza.py installation

- Type

```
git clone git://cfdem.git.sourceforge.net/gitroot/cfdem/lpp
```

LIGGGHTS
Simulation Running

What does the output mean?

```
LIGGGHTS 1.2.7 based on lammps-10Mar10
Created orthogonal box = (-0.05 -0.05 0) to (0.05 0.05 0.15)
  1 by 1 by 1 processor grid
0 atoms in group nve_group
Particle insertion: 2100 every 3197 steps, 1800 by step 1
Setting up run ...
Memory usage per processor = 9.57344 Mbytes
Step Atoms KinEng 1 Volume
       0        0           -0            0       0.0015
WARNING: Less insertions than requested
       1      963    0.0391011            0       0.0015
Loop time of 1.76714 on 1 procs for 1 steps with 963 atoms

Pair  time (%) = 4.1008e-05 (0.00232058)
Neigh time (%) = 0.00112796 (0.0638295)
Comm  time (%) = 8.10623e-06 (0.000458719)
Outpt time (%) = 1.90735e-05 (0.00107934)
Other time (%) = 1.76595 (99.9323)

Nlocal:    963 ave 963 max 963 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Nghost:    0 ave 0 max 0 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Neighs:    3134 ave 3134 max 3134 min
Histogram: 1 0 0 0 0 0 0 0 0 0

Total # of neighbors = 3134
Ave neighs/atom = 3.25441
Neighbor list builds = 1
Dangerous builds = 0
Setting up run ...
```

LIGGGHTS
Simulation Running

What does the output mean?

Simulation Box and processor grid (for parallel computation)

```
Created orthogonal box = (-0.05 -0.05 0) to (0.05 0.05 0.15)
  1 by 1 by 1 processor grid
```

Info about insertion of particles

```
Particle insertion: 2100 every 3197 steps, 1800 by step 1
Setting up run ...
Memory usage per processor = 9.14886 Mbytes
```

"Thermo" Info about time-step, number of particles in domain, translatory and rotational energy, and total simulation box volume

```
Step Atoms KinEng 1 Volume
       1      963    0.0391011            0       0.0015
    1000      963   0.05064266            0       0.0015
    2000      963  0.063711413            0       0.0015
    3000      963  0.078296562            0       0.0015
    4000     1800   0.13582887            0       0.0015
    5000     1800   0.16448873            0       0.0015
    5001     1800   0.16451881            0       0.0015
Loop time of 1.05419 on 1 procs for 5000 steps with 1800 atoms
```

LIGGGHTS
Simulation Running

What does the output mean?

Statistics – how much time was needed for which parts of the algorithms?

```
Pair  time (%) = 0.218345 (20.712)
Neigh time (%) = 0.0682487 (6.47402)
Comm  time (%) = 0.00209141 (0.198389)
Outpt time (%) = 0.0375419 (3.56119)
Other time (%) = 0.727967 (69.0543)
```

Statistics – how was the particle distribution and neighbor distribution among the processors

```
Nlocal:    1800 ave 1800 max 1800 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Nghost:    0 ave 0 max 0 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Neighs:    3623 ave 3623 max 3623 min
Histogram: 1 0 0 0 0 0 0 0 0 0
```

LIGGGHTS
Input Script

Input scripts do:

- Simulation setup
- Setup of simulation output
- Optimization loops
- Definition of variables
- Execution of shell commands

LIGGGHTS
Input Script Features/Rules

- LIGGGHTS executes by reading commands from an input script (text file) one line at a time. (each command takes effect when it is read)
- In many cases, the ordering of commands is not important, however, some commands are only valid when they follow other commands.

- Each non-blank line in the input script is treated as a command
- LIGGGHTS commands are **case sensitive**.
- An **&** at the end of a line means that the command continues on the next line
- All characters following a **#** character are treated as comments
- A **$** character indicates the beginning of a variable name in a line
- A line is broken into “words” separated by whitespaces (tabs, spaces)
- The **first word** is the **command name**, the rest are arguments

LIGGGHTS
Input Script Structure

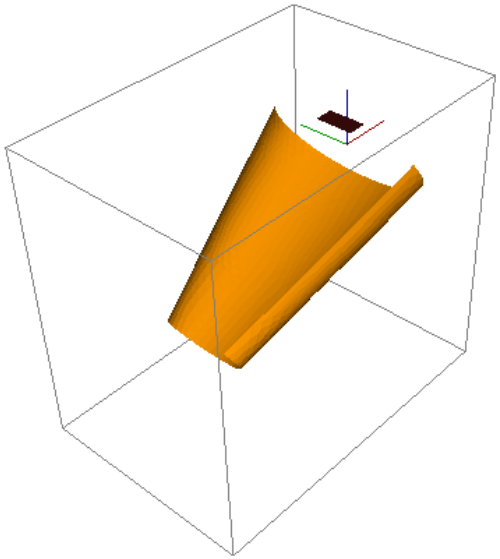
A LIGGGHTS input script typically has 4 parts

- Initialization
- Atom definition
- Settings
- Run a simulation

The last 2 parts can be repeated as many times as desired, i.e. run a simulation, change some settings, run some more, etc.

LIGGGHTS
Input Script Example (1)

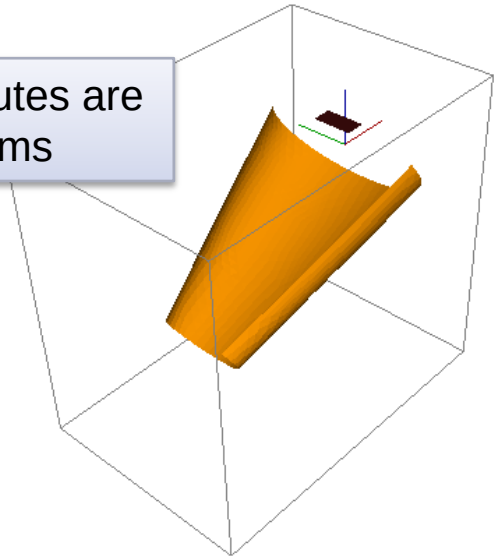
```
#Simple chute wear test

atom_style        granular
atom_modify       map array
boundary          f f f
newton            off

communicate       single vel yes

units             si

region            domain block -0.5 0.1 -0.2 0.2 -0.4 0.15 units box
create_box        1 domain

neighbor          0.002 bin
neigh_modify      delay 0
```



LIGGGHTS
Input Script Example (1)

```
#Simple chute wear test

atom_style          granular
atom_modify         map array
boundary            f f f
newton              off

communicate         single vel yes

units               si

region              domain block -0.5 0.1 -0.2 0.2 -0.4 0.15 units box
create_box          1 domain

neighbor            0.002 bin
neigh_modify        delay 0
```

Determines what attributes are associated with the atoms

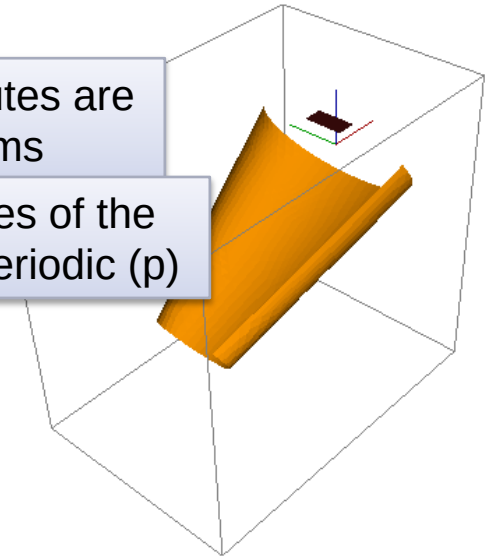


LIGGGHTS
Input Script Example (1)

```
#Simple chute wear test

atom_style          granular
atom_modify         map array
boundary            f f f
newton              off

communicate         single vel yes

units               si

region              domain block -0.5 0.1 -0.2 0.2 -0.4 0.15 units box
create_box          1 domain

neighbor            0.002 bin
neigh_modify        delay 0
```

Determines what attributes are associated with the atoms

Describes the boundaries of the domain as fixed (f) or periodic (p)



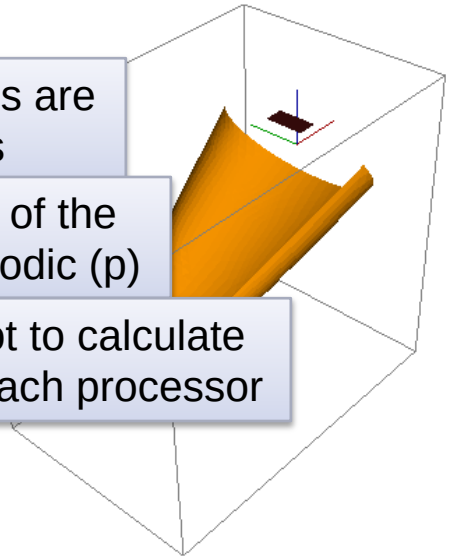
LIGGGHTS
Input Script Example (1)

```
#Simple chute wear test

atom_style          granular
atom_modify         map array
boundary            f f f
newton              off

communicate         single vel yes

units               si

region              domain block -0.5 0.1 -0.2 0.2 -0.4 0.15 units box
create_box          1 domain

neighbor            0.002 bin
neigh_modify        delay 0
```

Determines what attributes are associated with the atoms

Describes the boundaries of the domain as fixed (f) or periodic (p)

Determines whether or not to calculate pairwise interactions on each processor



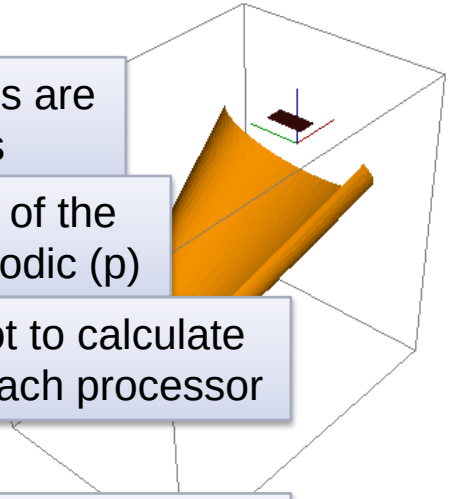
LIGGGHTS
Input Script Example (1)

```
#Simple chute wear test

atom_style          granular
atom_modify         map array
boundary            f f f
newton              off

communicate         single vel yes

units               si

region              domain block -0.5 0.1 -0.2 0.2 -0.4 0.15 units box
create_box          1 domain

neighbor            0.002 bin
neigh_modify        delay 0
```

Determines what attributes are associated with the atoms

Describes the boundaries of the domain as fixed (f) or periodic (p)

Determines whether or not to calculate pairwise interactions on each processor



Specifies a region called ‘domain’ that describes the bounds of the domain

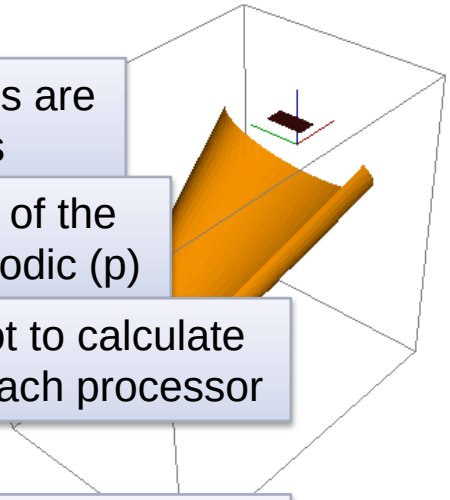
LIGGGHTS
Input Script Example (1)

```
#Simple chute wear test

atom_style          granular
atom_modify         map array
boundary            f f f
newton              off

communicate         single vel yes

units               si

region              domain block -0.5 0.1 -0.2 0.2 -0.4 0.15 units box
create_box          1 domain

neighbor            0.002 bin
neigh_modify        delay 0
```

Determines what attributes are associated with the atoms

Describes the boundaries of the domain as fixed (f) or periodic (p)

Determines whether or not to calculate pairwise interactions on each processor

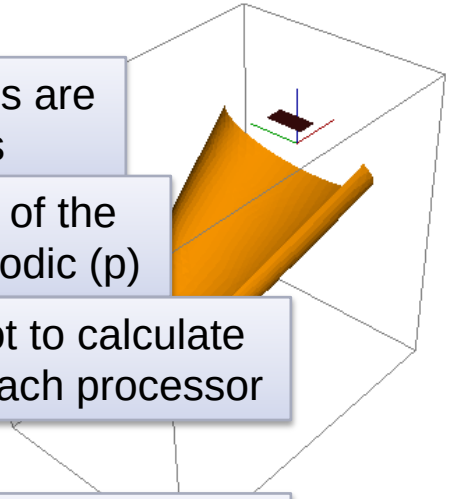


Specifies a region called ‘domain’ that describes the bounds of the domain

This simulation uses one material type

LIGGGHTS
Input Script Example (1)

```
#Simple chute wear test

atom_style          granular
atom_modify         map array
boundary            f f f
newton              off

communicate         single vel yes

units               si

region              domain block -0.5 0.1 -0.2 0.2 -0.4 0.15 units box
create_box          1 domain

neighbor            0.002 bin
neigh_modify        delay 0
```



Determines what attributes are associated with the atoms

Describes the boundaries of the domain as fixed (f) or periodic (p)

Determines whether or not to calculate pairwise interactions on each processor

Specifies a region called ‘domain’ that describes the bounds of the domain

This simulation uses one material type

Neighbor statements describe how large neighbor lists will be and how often to recalculate

LIGGGHTS
Input Script Example (2)

```
#Material properties required for pair style

fix      m1 all property/global youngsModulus peratomtype 5.e6
fix      m2 all property/global poissonsRatio peratomtype 0.45
fix      m3 all property/global coefficientRestitution peratomtypepair 1 0.3
fix      m4 all property/global coefficientFriction peratomtypepair 1 0.5
fix      m5 all property/global k_finnie peratomtypepair 1 1.0

#pair style
pair_style       gran/hertz/history
pair_coeff       * *

timestep         0.00001

fix              gravi all gravity 9.81 vector 0.0 0.0 -1.0
```

LIGGGHTS
Input Script Example (2)

Statements to specify material and interaction properties

```
#Material properties required for pair style

fix      m1 all property/global youngsModulus peratomtype 5.e6
fix      m2 all property/global poissonsRatio peratomtype 0.45
fix      m3 all property/global coefficientRestitution peratomtypepair 1 0.3
fix      m4 all property/global coefficientFriction peratomtypepair 1 0.5
fix      m5 all property/global k_finnie peratomtypepair 1 1.0

#pair style
pair_style        gran/hertz/history
pair_coeff        * *

timestep          0.00001

fix               gravi all gravity 9.81 vector 0.0 0.0 -1.0
```

LIGGGHTS
Input Script Example (2)

```
#Material properties required for pair style

fix      m1 all property/global youngsModulus peratomtype 5.e6
fix      m2 all property/global poissonsRatio peratomtype 0.45
fix      m3 all property/global coefficientRestitution peratomtypepair 1 0.3
fix      m4 all property/global coefficientFriction peratomtypepair 1 0.5
fix      m5 all property/global k_finnie peratomtypepair 1 1.0
```

Statements to specify material and interaction properties

```
#pair style
pair_style        gran/hertz/history
pair_coeff        * *

timestep          0.00001

fix               gravi all gravity 9.81 vector 0.0 0.0 -1.0
```

Define the model to be used for pairwise interaction; pair_coeff statements can be used to describe different interactions between different particle types

LIGGGHTS
Input Script Example (2)

Statements to specify material and interaction properties

```
#Material properties required for pair style

fix      m1 all property/global youngsModulus peratomtype 5.e6
fix      m2 all property/global poissonsRatio peratomtype 0.45
fix      m3 all property/global coefficientRestitution peratomtypepair 1 0.3
fix      m4 all property/global coefficientFriction peratomtypepair 1 0.5
fix      m5 all property/global k_finnie peratomtypepair 1 1.0
```

Define the model to be used for pairwise interaction; pair_coeff statements can be used to describe different interactions between different particle types

```
#pair style
pair_style        gran/hertz/history
pair_coeff        * *

timestep          0.00001

fix               gravi all gravity 9.81 vector 0.0 0.0 -1.0
```

command name

LIGGGHTS
Input Script Example (2)

```
#Material properties required for pair style

fix      m1 all property/global youngsModulus peratomtype 5.e6
fix      m2 all property/global poissonsRatio peratomtype 0.45
fix      m3 all property/global coefficientRestitution peratomtypepair 1 0.3
fix      m4 all property/global coefficientFriction peratomtypepair 1 0.5
fix      m5 all property/global k_finnie peratomtypepair 1 1.0
```

Statements to specify material and interaction properties

```
#pair style
pair_style        gran/hertz/history
pair_coeff        * *

timestep          0.00001

fix               gravi all gravity 9.81 vector 0.0 0.0 -1.0
```

Define the model to be used for pairwise interaction; pair_coeff statements can be used to describe different interactions between different particle types

command ID

LIGGGHTS
Input Script Example (2)

Statements to specify material and interaction properties

```
#Material properties required for pair style

fix      m1 all property/global youngsModulus peratomtype 5.e6
fix      m2 all property/global poissonsRatio peratomtype 0.45
fix      m3 all property/global coefficientRestitution peratomtypepair 1 0.3
fix      m4 all property/global coefficientFriction peratomtypepair 1 0.5
fix      m5 all property/global k_finnie peratomtypepair 1 1.0
```

Define the model to be used for pairwise interaction; pair_coeff statements can be used to describe different interactions between different particle types

```
#pair style
pair_style        gran/hertz/history
pair_coeff        * *

timestep          0.00001

fix               gravi all gravity 9.81 vector 0.0 0.0 -1.0
```

group ID

LIGGGHTS
Input Script Example (2)

```
#Material properties required for pair style

fix      m1 all property/global youngsModulus peratomtype 5.e6
fix      m2 all property/global poissonsRatio peratomtype 0.45
fix      m3 all property/global coefficientRestitution peratomtypepair 1 0.3
fix      m4 all property/global coefficientFriction peratomtypepair 1 0.5
fix      m5 all property/global k_finnie peratomtypepair 1 1.0
```

Statements to specify material and interaction properties

```
#pair style
pair_style        gran/hertz/history
pair_coeff        * *

timestep          0.00001

fix               gravi all gravity 9.81 vector 0.0 0.0 -1.0
```

Define the model to be used for pairwise interaction; pair_coeff statements can be used to describe different interactions between different particle types

fix style

LIGGGHTS
Input Script Example (3)

```
#the chute
fix      cad all mesh/surface/stress file simple_chute.stl type 1 wear finnie
fix      inface all mesh/surface file insertion_face.stl type 1
fix      granwalls all wall/gran/hertz/history mesh n_meshes 1 meshes cad

#particle distributions for insertion
fix      pts1 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0015
fix      pts2 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0025
fix      pdd1 all particledistribution/discrete 1 2 pts1 0.3 pts2 0.7

#particle insertion
fix      ins all insert/stream seed 5330 distributiontemplate pdd1 &
         nparticles 6000 massrate 0.1 insert_every 1000 overlapcheck yes &
         all_in no vel constant 0.0 0.0 -1.0 insertion_face inface
```

LIGGGHTS
Input Script Example (3)

Geometry files to be read in to the simulation

```
#the chute
fix      cad all mesh/surface/stress file simple_chute.stl type 1 wear finnie
fix      inface all mesh/surface file insertion_face.stl type 1
fix      granwalls all wall/gran/hertz/history mesh n_meshes 1 meshes cad

#particle distributions for insertion
fix      pts1 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0015
fix      pts2 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0025
fix      pdd1 all particledistribution/discrete 1 2 pts1 0.3 pts2 0.7

#particle insertion
fix      ins all insert/stream seed 5330 distributiontemplate pdd1 &
         nparticles 6000 massrate 0.1 insert_every 1000 overlapcheck yes &
         all_in no vel constant 0.0 0.0 -1.0 insertion_face inface
```

LIGGGHTS
Input Script Example (3)

```
#the chute
fix      cad all mesh/surface/stress file simple_chute.stl type 1 wear finnie
fix      inface all mesh/surface file insertion_face.stl type 1
fix      granwalls all wall/gran/hertz/history mesh n_meshes 1 meshes cad
```

Geometry actually used as wall

```
#particle distributions for insertion
fix      pts1 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0015
fix      pts2 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0025
fix      pdd1 all particledistribution/discrete 1 2 pts1 0.3 pts2 0.7

#particle insertion
fix      ins all insert/stream seed 5330 distributiontemplate pdd1 &
         nparticles 6000 massrate 0.1 insert_every 1000 overlapcheck yes &
         all_in no vel constant 0.0 0.0 -1.0 insertion_face inface
```

LIGGGHTS
Input Script Example (3)

```
#the chute
fix      cad all mesh/surface/stress file simple_chute.stl type 1 wear finnie
fix      inface all mesh/surface file insertion_face.stl type 1
fix      granwalls all wall/gran/hertz/history mesh n_meshes 1 meshes cad
```

Geometry actually used as wall

```
#particle distributions for insertion
fix      pts1 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0015
fix      pts2 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0025
fix      pdd1 all particledistribution/discrete 1 2 pts1 0.3 pts2 0.7

#particle insertion
fix      ins all insert/stream seed 5330 distributiontemplate pdd1 &
         nparticles 6000 massrate 0.1 insert_every 1000 overlapcheck yes &
         all_in no vel constant 0.0 0.0 -1.0 insertion_face inface
```

LIGGGHTS
Input Script Example (3)

```
#the chute
fix      cad all mesh/surface/stress file simple_chute.stl type 1 wear finnie
fix      inface all mesh/surface file insertion_face.stl type 1
fix      granwalls all wall/gran/hertz/history mesh n_meshes 1 meshes cad
```

Geometry actually used as wall

```
#particle distributions for insertion
fix      pts1 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0015
fix      pts2 all particletemplate/sphere 1 atom_type 1 density constant 2500 &
         radius constant 0.0025
fix      pdd1 all particledistribution/discrete 1 2 pts1 0.3 pts2 0.7

#particle insertion
fix      ins all insert/stream seed 5330 distributiontemplate pdd1 &
         nparticles 6000 massrate 0.1 insert_every 1000 overlapcheck yes &
         all_in no vel constant 0.0 0.0 -1.0 insertion_face inface
```

LIGGGHTS
Input Script Example (4)

```
#apply nve integration to all particles
fix                 integr all nve/sphere

#output settings, include total thermal energy
compute             erot all erotate/sphere
thermo_style        custom step atoms ke c_erot vol
thermo              1000
thermo_modify       lost ignore norm no
compute_modify      thermo_temp dynamic yes

#insert the first particles so that dump is not empty
run       1
dump      dmp all custom 200 post/dump*.chute id type x y z ix iy iz &
          vx vy vz fx fy fz omegax omegay omegaz radius
dump      dumpstress all mesh/gran/VTK 200 post/dump*.vtk stress wear cad

#run simulation
run                 100000 upto

write_restart       restart/chute.restart
```

LIGGGHTS
Input Script Example (4)

```
#apply nve integration to all particles
fix                 integr all nve/sphere

#output settings, include total thermal energy
compute             erot all erotate/sphere
thermo_style        custom step atoms ke c_erot vol
thermo              1000
thermo_modify       lost ignore norm no
compute_modify      thermo_temp dynamic yes

#insert the first particles so that dump is not empty
run       1
dump      dmp all custom 200 post/dump*.chute id type x y z ix iy iz &
          vx vy vz fx fy fz omegax omegay omegaz radius
dump      dumpstress all mesh/gran/VTK 200 post/dump*.vtk stress wear cad

#run simulation
run                 100000 upto

write_restart       restart/chute.restart
```

NVE - microcanonical ensemble

LIGGGHTS
Input Script Example (4)

```
#apply nve integration to all particles
fix                 integr all nve/sphere

#output settings, include total thermal energy
compute             erot all erotate/sphere
thermo_style        custom step atoms ke c_erot vol
thermo              1000
thermo_modify       lost ignore norm no
compute_modify      thermo_temp dynamic yes

#insert the first particles so that dump is not empty
run       1
dump      dmp all custom 200 post/dump*.chute id type x y z ix iy iz &
          vx vy vz fx fy fz omegax omegay omegaz radius
dump      dumpstress all mesh/gran/VTK 200 post/dump*.vtk stress wear cad

#run simulation
run                 100000 upto

write_restart       restart/chute.restart
```

NVE - microcanonical ensemble

Calculate the rotational energy

LIGGGHTS
Input Script Example (4)

```
#apply nve integration to all particles
fix                 integr all nve/sphere

#output settings, include total thermal energy
compute             erot all erotate/sphere
thermo_style        custom step atoms ke c_erot vol
thermo              1000
thermo_modify       lost ignore norm no
compute_modify      thermo_temp dynamic yes

#insert the first particles so that dump is not empty
run       1
dump      dmp all custom 200 post/dump*.chute id type x y z ix iy iz &
          vx vy vz fx fy fz omegax omegay omegaz radius
dump      dumpstress all mesh/gran/VTK 200 post/dump*.vtk stress wear cad

#run simulation
run                 100000 upto

write_restart       restart/chute.restart
```

NVE - microcanonical ensemble

Calculate the rotational energy

Define log output

LIGGGHTS
Input Script Example (4)

```
#apply nve integration to all particles
fix                 integr all nve/sphere

#output settings, include total thermal energy
compute             erot all erotate/sphere
thermo_style        custom step atoms ke c_erot vol
thermo              1000
thermo_modify       lost ignore norm no
compute_modify      thermo_temp dynamic yes

#insert the first particles so that dump is not empty
run       1
dump      dmp all custom 200 post/dump*.chute id type x y z ix iy iz &
          vx vy vz fx fy fz omegax omegay omegaz radius
dump      dumpstress all mesh/gran/VTK 200 post/dump*.vtk stress wear cad

#run simulation
run                 100000 upto

write_restart       restart/chute.restart
```

NVE - microcanonical ensemble

Calculate the rotational energy

Define log output

Define log output frequency

LIGGGHTS
Input Script Example (4)

```
#apply nve integration to all particles
fix                 integr all nve/sphere

#output settings, include total thermal energy
compute             erot all erotate/sphere
thermo_style        custom step atoms ke c_erot vol
thermo              1000
thermo_modify       lost ignore norm no
compute_modify      thermo_temp dynamic yes

#insert the first particles so that dump is not empty
run       1
dump      dmp all custom 200 post/dump*.chute id type x y z ix iy iz &
          vx vy vz fx fy fz omegax omegay omegaz radius
dump      dumpstress all mesh/gran/VTK 200 post/dump*.vtk stress wear cad

#run simulation
run                 100000 upto

write_restart       restart/chute.restart
```

NVE - microcanonical ensemble

Calculate the rotational energy

Define log output

Define log output frequency

Execute specified number of timesteps

LIGGGHTS
Input Script Example (4)

```
#apply nve integration to all particles
fix                 integr all nve/sphere

#output settings, include total thermal energy
compute             erot all erotate/sphere
thermo_style        custom step atoms ke c_erot vol
thermo              1000
thermo_modify       lost ignore norm no
compute_modify      thermo_temp dynamic yes

#insert the first particles so that dump is not empty
run       1
dump      dmp all custom 200 post/dump*.chute id type x y z ix iy iz &
          vx vy vz fx fy fz omegax omegay omegaz radius
dump      dumpstress all mesh/gran/VTK 200 post/dump*.vtk stress wear cad

#run simulation
run                 100000 upto

write_restart       restart/chute.restart
```

NVE - microcanonical ensemble

Calculate the rotational energy

Define log output

Define log output frequency

Execute specified number of timesteps

Define simulation output

LIGGGHTS
Input Script Example (4)

```
#apply nve integration to all particles
fix                 integr all nve/sphere

#output settings, include total thermal energy
compute             erot all erotate/sphere
thermo_style        custom step atoms ke c_erot vol
thermo              1000
thermo_modify       lost ignore norm no
compute_modify      thermo_temp dynamic yes

#insert the first particles so that dump is not empty
run       1
dump      dmp all custom 200 post/dump*.chute id type x y z ix iy iz &
          vx vy vz fx fy fz omegax omegay omegaz radius
dump      dumpstress all mesh/gran/VTK 200 post/dump*.vtk stress wear cad

#run simulation
run                 100000 upto

write_restart       restart/chute.restart
```

NVE - microcanonical ensemble

Calculate the rotational energy

Define log output

Define log output frequency

Execute specified number of timesteps

Define simulation output

upto: advance to this timestep continuing from current status

LIGGGHTS
Input Script Example (4)

```
#apply nve integration to all particles
fix                 integr all nve/sphere
```

NVE - microcanonical ensemble

```
#output settings, include total thermal energy
compute             erot all erotate/sphere
thermo_style        custom step atoms ke c_erot vol
thermo              1000
thermo_modify       lost ignore norm no
compute_modify      thermo_temp dynamic yes
```

Calculate the rotational energy

Define log output

Define log output frequency

```
#insert the first particles so that dump is not empty
run       1
```

Execute specified number of timesteps

```
dump      dmp all custom 200 post/dump*.chute id type x y z ix iy iz &
          vx vy vz fx fy fz omegax omegay omegaz radius
dump      dumpstress all mesh/gran/VTK 200 post/dump*.vtk stress wear cad
```

Define simulation output

```
#run simulation
run                 100000 upto
```

upto: advance to this timestep continuing from current status

```
write_restart       restart/chute.restart
```

Save data to be able to continue simulation

Thank you for your attention!
Questions?

www.cfdem.com | www.particulate-flow.at

christoph.kloss@cfdem.com

Introduction to Dynamical Particle Simulations

Extending LIGGGHTS

How to modify **LIGGGHTS for your own needs**

Daniel Queteschiner*

Christoph Kloss**

***daniel.queteschiner@gmx.at**

****christoph.kloss@cfdem.com**

Extending LIGGGHTS
General Hints and Guidelines

There are 3 main features you may want modify in LIGGGHTS:

Pair Styles

pair-wise (particle-particle) interaction

Computes

mainly used for diagnostics and to gather data for post-processing

Fixes

a “fix” is an operation that is applied to the system during timestepping, e.g., time integration, applying constraint forces to atoms, enforcing boundary conditions, computing diagnostics

Extending LIGGGHTS
General Hints and Guidelines

How to go about modifying LIGGGHTS:

Understanding C++ basics is essential

- If you are not yet familiar with C++, do some basic tutorials

Reuse existing code

- Look for a fix / compute / pair that does something similar you want (**read the manual**)
- Identify the source file containing the code in question
- Understand how it works
- Copy it to a new file and modify it

Read the manual

- Section 10 gives a modification overview

Learning by doing

- Probably the most important part …

Example: FLOWSB & LIGGGHTS One-Way Coupling

Extending LIGGGHTS
Goals

- **Read data from ft12 files**
- **Transform data into velocity field in physical space**
- **Apply drag force to particles**

Extending LIGGGHTS
Steps taken

- Identify any existing mechanism in LIGGGHTS to apply a force on particles:
- **`fix addforce`** or **`fix setforce`**
- Those require per atom variables (fx, fy, fz) as arguments
- Identify any existing mechanism in LIGGGHTS to compute per atom variables, e.g.
- **`compute displace/atom`** (compute_displace_atom.cpp)
- Copy file to compute_force_atom.cpp and modify to calculate Schiller-Naumann force
- *Intermediate step: read velocity field in physical space from formatted file*
- *Intermediate step: read velocity field in physical space from unformatted file*
- Read ft12 files and transform data into velocity field in physical space

  (converted Fortran code to C++)
- Apply scaling to velocity field (wall units – SI / cgs)
- Testing and Debugging

Extending LIGGGHTS
Fortran vs. C++: Pitfalls

Unformatted Sequential Fortran Files

- Files written by Fortran programs contain **data + record size information**
- Byte-Order: Little Endian vs. Big Endian

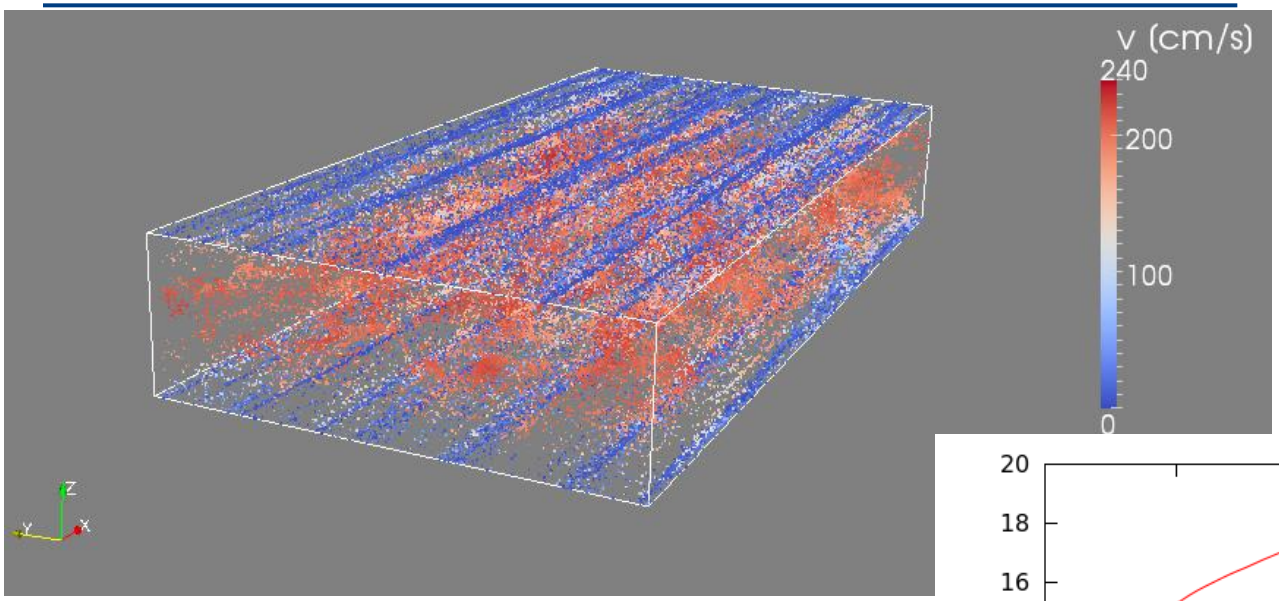
Multidimensional Arrays

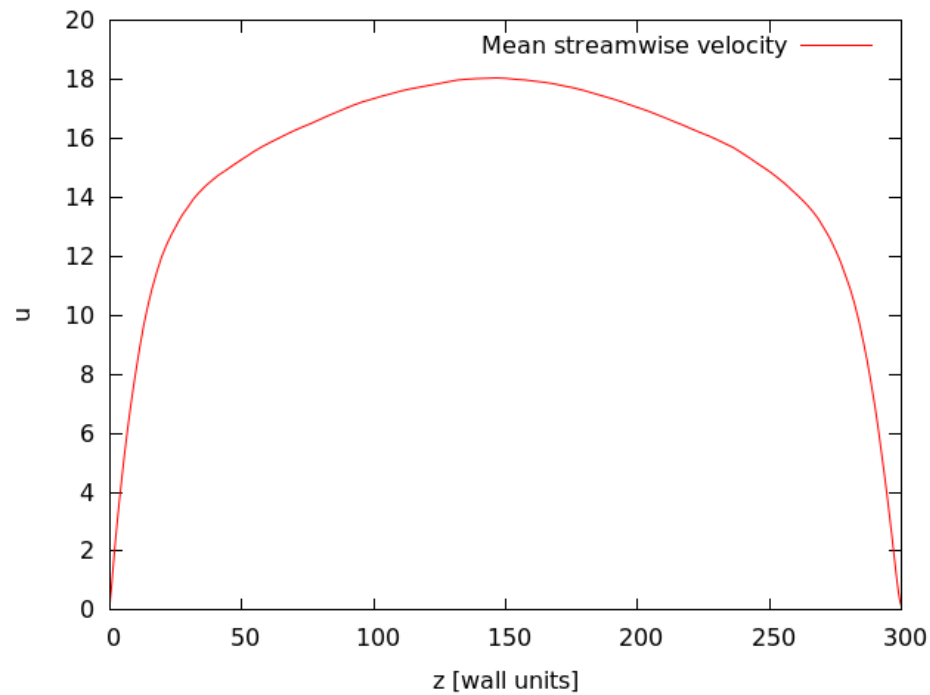
- Indexing starts with 1 (Fortran) or 0 (C++)
  (source of off-by-one errors)
- No negative indexes in C++ (e.g. bordering of grid for interpolation)
- Storage:
  - Column-major order vs. Row-major order

INTEGER A(2,3)	A(1,1)	A(2,1)	A(1,2)	A(2,2)	A(1,3)	A(2,3)
int a[2][3];	a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]

  - Ensure contiguous memory in C++ (array of arrays vs. array of pointers; MPI)
- Redimensioning

Extending LIGGGHTS
Goals achieved





100.000 spherical particles in simulation

6000 ft12 files read (~1 s of simulated time)

2 DEM time-steps executed / file

runtime ca. 18h on a single processor

Thank you for your attention!
Questions?

www.cfdem.com | www.particulate-flow.at

christoph.kloss@cfdem.com