# DESIgn, version 1.3

## Discrete-Element bonded-particle Sea Ice toolbox for LIGGGHTS

# User's Manual

**Agnieszka Herman**

Institute of Oceanography
University of Gdańsk, Poland

Email: agnieszka.herman@ug.edu.pl

http://herman.ocean.ug.edu.pl/LIGGGHTSseaice.html

May 31, 2015

# Contents

# Chapter 1

# Introduction

This document is a supplementary material to the paper "Discrete-Element bonded-particle Sea Ice model DESIgn, version 1.3. Model description and implementation" by Agnieszka Herman (Geoscientific Model Development Discuss., 2015). It contains the user's manual of the model described in that paper.
The DESIgn toolbox presented here is based on LIGGGHTS (LAMMPS Improved for General Granular and Granular Heat Transfer Simulations; `http://www.cfdem.com/liggghtsr-open-source-discrete-element-method-particle-simulation-code`; Kloss and Goniva, 2010, 2011; **?**), which in turn is based on LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator; `http://lammps.sandia.gov/`; Plimpton, 1995). All functionalities of the sea ice toolbox are defined as derived from classes already used in LIGGGHTS. It is worth noting that the present version of the sea ice model is an extension of the previous versions, based exclusively on LAMMPS (Herman, 2012, 2013a,b,c). The new model is called DESIgn; alternatively, treated as an optional toolbox for LIGGGHTS, the name USER-SEAICE is used, in accordance to the naming convention used for other optional, user-designed toolboxes of LAMMPS/LIGGGHTS. This documents describes the version 1.3 of DESIgn.
The code of the toolbox, together with documentation and input files necessary to run the test cases, is attached to the paper cited above (file `DESIgn_4liggghts_v1.3.zip`). It is also available online at `http://herman.ocean.ug.edu.pl/LIGGGHTSseaice.html` (section 'Code and documentation').
Throughout this manual, reference is made to the documentation of LAMMPS and, especially, LIGGGHTS, available on their web sites.

## 1.1 Contents of the sea ice toolbox

- `doc`
  documentation of the toolbox, including this manual and documentation of the sea-ice model commands in the text and html format compliant with that used by LAMMPS and LIGGGHTS
- `src`
  source code of the toolbox:
  - `USER-SEAICE` – files constituting the toolbox
  - `src-seaice` – files that are part of LIGGGHTS, in which modifications are required (see step 4 on the list in Section 1.2 below)
- `examples`
  input files and configurations scripts necessary to run the test cases (see Chapter 4)

- `utils`
  various Matlab scripts that can be used, e.g., to generate data files with initial conditions, to read the model output, to plot the results etc. – see Section 4.2.1 and comments within the respective `*.m` files

## 1.2 System requirements and installation

The DESIgn code has been tested on a few Linux clusters with different architectures, with GNU and Intel compilers, and with MPICH2 and MVAPICH. LIGGGHTS can run on a number of operational systems with a number of compilers – see its documentation for details. Generally, if you are able to compile and run LIGGGHTS without the sea ice toolbox, you should also be able to compile and run it with the sea ice code included, as it has no additional requirements. The DESIgn version 1.3 model has been written for LIGGGHTS version 3.1.0, released on 18 February 2015. Effort will be taken to keep the code up to date and compatible with the future LIGGGHTS versions. Updates will be available at `http://herman.ocean.ug.edu.pl/LIGGGHTSseaice.html`.
To install LIGGGHTS with the sea ice toolbox, do the following steps:

1. Download and unpack the LIGGGHTS code from its web site. In the further description it is assumed that the folder with the code is called `LIGGGHTS`.

2. Download and unpack the zipped sea-ice code (`DESIgn_4liggghts_v1.3.zip`).

3. Copy the folder USER-SEAICE to `LIGGGHTS/src`.

4. This step depends on the LIGGGHTS version you are installing.

   - For LIGGGHTS 3.1.0:
     Replace files `atom.h`, `atom.cpp`, `contact_interface.h`, `style_normal_model.h`, `pair_gran_base.h` and `style_contact_model.h` in `LIGGGHTS/src` with those provided in `src-seaice`.
   - For LIGGGHTS newer than 3.1.0:
     Check whether a corresponding version of `DESIgn_4liggghts_vX.X.zip` is available at `http://herman.ocean.ug.edu.pl/LIGGGHTSseaice.html`.
     If not, modify the code in the affected files by inserting the lines indicated in Table 1.1 at proper places, using the files in `src-seaice` folder as a template (search for the code preceded by the comment "// used by USER-SEAICE").

5. Go to `LIGGGHTS/src` and type:
   `install USER-SEAICE`
   This will copy all sea-ice toolbox files into the `LIGGGHTS/src` directory.

6. Compile and build LIGGGHTS in the usual way, i.e., by typing:
   `make your-make-script`
   where `your-make-script` is the name of the compilation script suitable for your machine and the compiler you're using (copy and modify one of the scripts available in the `MAKE` directory). As a result, an executable named `lmp_your-make-script` will be created.

7. Finally, you may want to copy the `*.html` and `*.txt` files from the `doc` folder of the sea ice toolbox to `LIGGGHTS/doc`, so that you may use the documentation of the toolbox together with the documentation of LIGGGHTS.

Table 1.1: Changes in the code of LIGGGHTS, necessary for the sea ice toolbox

| File | Code that should be added |
|---|---|
| atom.h | `double *thickness,**tilt;` |
| | `int disk_flag,thickness_flag,tilt_flag;` |
| atom.cpp | `thickness = NULL;` |
| | `tilt = NULL;` |
| | `disk_flag = thickness_flag = tilt_flag = 0;` |
| contact_interface.h | `double hmin;` |
| pair_gran.h | `if (!atom->sphere_flag && !atom->disk_flag)` |
| | `error->all(FLERR,"pair granular requires atom style sphere or disk");` |
| pair_gran_base.h | `int *disk_flag = atom->disk_flag;` |
| | `if (disk_flag>0) {` |
| | `double *thickness = atom->thickness;` |
| | `for (int ii = 0; ii< inum; ii++) {` |
| | ... (whole content of the loop) ... |
| | } |
| | `} else {` |
| style_contact_model.h | `GRAN_MODEL(HERTZ_DISK, TANGENTIAL_HISTORY, ...)` |
| | `GRAN_MODEL(HERTZ_DISK, TANGENTIAL_NO_HISTORY, ...)` |
| | `GRAN_MODEL(HERTZ_STIFFNESS_DISK, TANGENTIAL_HISTORY, ...)` |
| | `GRAN_MODEL(HERTZ_STIFFNESS_DISK, TANGENTIAL_NO_HISTORY, ...)` |
| style_normal_model.h | `#include "normal_model_hertz_disk.h"` |
| | `#include "normal_model_hertz_stiffness_disk.h"` |

# Chapter 2

# Technical documentation

As already mentioned in the Introduction, LAMMPS and LIGGGHTS have been developed for efficient modeling of large systems of interacting objects in general, and granular materials in particular. Thanks to that, the present discrete-element (DEM) sea ice model could be implemented relatively easily, by making use of the existing modeling framework.
In this chapter, the most important technical aspects of the sea ice toolbox are presented. Section 2.1 describes briefly those elements of LIGGGHTS that are crucial for the functioning of the sea ice toolbox, including the granular particles, the Hertz contact mechanics model and the treatment of grain–grain and grain–wall interactions, as well as the numerical solvers used to integrate the momentum equations in time. Subsequently, all new elements of the sea ice toolbox are described in Section 2.2.
In principle, readers not interested in technical aspects of the model code may omit this chapter and skip to the next one, containing the practical details related to configuring and running the model. However, at least basic knowledge of the material presented below is advisable for a better understanding of the model functioning and its possibilities and limitations.

## 2.1 LIGGGHTS functionalities essential for the sea ice toolbox

### 2.1.1 Granular particles

The main atom (particle) style in LIGGGHTS for simulating granular materials is `sphere`. In the contact mechanics models implemented in LIGGGHTS (see below), spherical geometry of interacting objects is assumed, in which the radius is the only parameter describing the size and shape of these objects. Throughout the code, the spherical geometry is used to calculate: (i) the moment of inertia of the particles; (ii) the contact forces in function of the overlap between particles, assuming circular contact area; (iii) moments and forces related to the presence of elastic bonds between atoms. Consequently, in the bonded-particle model available in the commercial version of LIGGGHTS (see Section 2.1.7 below), bonds are represented as elastic rods with a circular cross-section.
(**Note**: if LIGGGHTS is run in a two-dimensional mode, i.e., when the `enforce2d` fix is used, *some* quantities are calculated as if the particles were *disks with unit thickness*. This is the case, for example, whenever the mass or volume of the atoms is calculated, but the computation of quantities dependent on the moment of inertia of the atoms is performed in the same way in 2D and 3D, see, e.g., the documentation of the `compute erotate/sphere` command. Most importantly, the formulae used to calculate pairwise interaction forces are suitable for spherical particle geometry, both in 2D and 3D.)
**Relevant commands**: `atom_style sphere`

### 2.1.2 Contact mechanics and global material properties

For spherical granular particles interacting with each other or with elastic walls, Hookean and Hertzian contact mechanics models are implemented in LIGGGHTS. They are used to calculate the normal and tangential components of the contact forces, with optional history, cohesion and/or rolling friction effects, and are available in two alternative versions that differ with respect to the calculation of the model coefficients, and thus the treatment of polydispersity. The forces are calculated based on prescribed material properties of the particles, including their Young modulus, restitution coefficient etc. – see the LIGGGHTS documentation and the Supplementary Note S1 accompanying the main paper (Herman, 2015) for details.
**Relevant commands**: `pair_style gran`, `gran model hertz`, `gran model hertz/stiffness`, `gran tangential history`

### 2.1.3 Velocity Verlet solver

Application of LIGGGHTS to granular systems, i.e., to finite-size particles, implies that the model equations have to be solved not only for position and velocity of the center of mass of the particles, but also for their angular velocity. LIGGGHTS provides velocity Verlet solvers for various kinds of systems, including the canonical ensemble (NVT, conserving the amount of moles, volume and temperature) and the microcanonical ensemble (NVE, conserving the amount of moles, volume and energy). The version of the NVE solver suitable for finite-size atoms is relevant for problems discussed here.
**Relevant commands**: `fix nve/sphere`, `run_style verlet`

### 2.1.4 Two-dimensional computations

By default, LIGGGHTS performs computations in three dimensions. Accordingly, all atoms are described by three components of location, velocity and – in the case of finite-size atoms – angular velocity; all forces and torques have three components as well.
LIGGGHTS enables computations in two dimensions, in which the vertical components of the velocity and forces acting on atoms are zeroed at each time step, so that the vertical coordinate of the atoms remains constant. Notably, this does not mean a reduction in memory requirements and computational time, as all calculations are performed in usual way and the zeroing of the z-components is added as an additional operation at each time step.
**Relevant commands**: `dimension`, `fix enforce2d`

### 2.1.5 Forces and torques acting on atoms

LIGGGHTS provides a very flexible way of specifying forces and torques acting on individual atoms or groups of atoms. The forces and torques can be constant or space- and time-varying, they may be functions of the properties and positions of the atoms, or generally of any user-specified variables, and they may act within the whole model domain or within user-specified subregions.
**Relevant commands**: `fix addforce`, `fix addtorque`

### 2.1.6 Computation of global and per-atom properties

Several global characteristics of the modeled system can be computed at selected time intervals and for selected spatial regions and groups/types of atoms – including the translational and rotational kinetic energy and their sum (temperature), pressure, stress, etc.
**Relevant commands**: `compute ke`, `compute erotate/sphere`, `compute temp/sphere`, `compute pressure`, `compute stress/atom`, `compute pair/gran/local`, `compute bond/local`

### 2.1.7 Elastic bonds

At present, elastic bonds for granular particles are *not* included in the open-source, public distribution of LIGGGHTS, but they are available in the commercial version (LIGGGHTS Premium). The new functionalities described further in Section 2.2.5 are based on experimental, preliminary code downloaded from the model repository in 2013. It included basic definitions and functions enabling the usage of bonds with granular particles. Not surprisingly considering the preliminary nature of the code, it contained some important bugs, but nevertheless provided invaluable starting point for the bond-related code in the sea-ice toolbox.
Importantly, the granular-bonds code makes use of LIGGGHTS functionalities suitable for other types of connected atoms, including point particles (see optional packages MC and MOLECULE). These include: specifying the coefficients defining the bonds' properties, creating bonds between neighbouring atoms during a simulation, or removing broken bonds from the bond list.
**Relevant commands**: `fix bond/break`, `fix bond/create`, `atom_style bond`, `bond_coeff`, `bond_style`

## 2.2 New functionalities

Without exception, all new elements of the sea ice model are based on existing ones (confirming the immense extendability and flexibility of LAMMPS and LIGGGHTS). As shown in Table 2.1, the new elements were created based on the existing classes for granular/spherical particles. For example, files `atom_vec_disk.h` and `atom_vec_disk.cpp` are modifications of `atom_vec_sphere.h` and `atom_vec_sphere.cpp`, etc.

### 2.2.1 Disk-shaped particles

Disk-shaped atoms – the basic atom type of the sea ice toolbox – are characterized by the same set of properties as spheres plus, additionally, thickness and, if the quasi-3D wave effects are to be taken into account, tilt (i.e., the angle between the axis of symmetry of the disk and the vertical axis of the coordinate system). All functions of the toolbox assume disk-shape geometry of atoms (variable `atom->disk_flag`, set in `atom_vec_disk.cpp`, must be equal 1).
In all variants of the disk-shaped atoms (files `atom_vec_disk*.cpp`), the differences in respect to spherical atoms relate to: (i) the size of the buffer variable used to pass the atom-related information between processors; this size depends on the number of properties characterizing atoms and is larger for disks than for spheres; (ii) the calculation of the mass of atoms based on their size (functions `create_atom`, `data_atom`, `data_atom_hybrid`, `pack_data` and `pack_data_hybrid` in `atom_vec_disk*.cpp`); (iii) the calculation of the moments of inertia of atoms; (iv) the calculation of the Hertzian repulsive force between atoms in contact with each other.
**Relevant files**: `atom_vec_disk.*`, `atom_vec_bond_gran_disk.*`, `atom_vec_disk_waves.*`
**Relevant commands**: `atom_style disk`, `atom_style hybrid disk bond/gran/disk`, `atom_style hybrid disk disk/waves bond/gran/disk`

### 2.2.2 Two-dimensional computations

If the quasi-3D wave effects are not taken into account, the standard `fix enforce2d` command can be used for sea ice simulations. With the quasi-3D effects, positions of the centers of mass of the atoms, as well as their translational velocities are calculated in 2D, but their angular velocities are calculated in 3D (see the equations in the main paper). The `fix enforce2d/seaice` command handles this option and performs the zeroing out of the proper variables.
**Relevant files**: `fix_enforce2d_seaice_waves.*`
**Relevant commands**: `fix enforce2d/seaice`

Table 2.1: New elements of LIGGGHTS included in the sea ice toolbox. Entries in the third column shown in italics were part of the experimental bond package and are not included in the open-source LIGGGHTS version available online.

| Element | Class name | Based on class |
|---|---|---|
| Disk-shaped atoms | AtomVecDisk | AtomVecSphere |
| Disk-shaped atoms with elastic bonds | AtomVecBondGranDisk | *AtomVecBondGran* |
| Disk-shaped atoms with quasi-3D wave effects | AtomVecDiskWaves | AtomVecDisk |
| Hertz contact model for disks | NormalModel<HERTZ_DISK> | NormalModel<HERTZ> |
| Hertz/stiffness contact model for disks | NormalModel <HERTZ_STIFFNESS_DISK> | NormalModel <HERTZ_STIFFNESS> |
| NVE solver for disk-shaped atoms | FixNVEDisk | FixNVESphere |
| Elastic bonds for disk-shaped atoms | BondGranDisk | *BondGran* |
| Elastic bonds for disk-shaped atoms (quasi-3D) | BondGranDiskWaves | BondGranDisk |
| Computation of per-atom rotational kinetic energy of disk-shaped atoms | ComputeERotateDiskAtom | ComputeERotateSphere |
| Computation of global rotational kinetic energy of disk-shaped atoms | ComputeErotateDisk | ComputeERotateSphereAtom |
| Computation of total kinetic energy of disk-shaped atoms | ComputeTempDisk | ComputeTempSphere |
| Fix to break bonds between disk-shaped atoms | FixBondBreakGranDisk | FixBondBreak (MC package) |
| Fix to create bonds between disk-shaped atoms | FixBondCreateGranDisk | *FixBondCreateGran* |
| Fix to propagate bonds between disk-shaped atoms | FixBondPropagateGranDisk | *FixBondPropagateGran* |
| Fix for simulations with quasi-3D wave effects | FixEnforce2DSeaiceWaves | FixEnforce2D |
| Fix to add Coriolis force | FixSeaiceCoriolis | FixAddForce |
| Fix to add current-induced force | FixSeaiceCurrent | FixAddForce |
| Fix to add wind-induced force | FixSeaiceWind | FixAddForce |
| Fix to add wave-induced force | FixSeaiceWaves | FixAddForce |

### 2.2.3 Computation of global and per-atom properties

Some global or local characteristics of the system – like for example the translational kinetic energy – are not affected by the change of the shape of atoms from spherical to cylindrical, and therefore they can be calculated by the existing fixes. Others require some (usually straightforward) modifications and are handled by fixes analogous to the ones implemented in the standard version of LIGGGHTS.
**Relevant files**: `compute_erotate_disk.*`, `compute_erotate_disk_atom.*`, `compute_bond_gran_disk_local.*`
**Relevant commands**: `compute erotate/disk`, `compute erotate/disk/atom`, `compute bond/gran/disk/local`

### 2.2.4 External forcing

Any required type of forcing acting on the ice 'grains' can be applied in a straightforward manner by using a combination of `variable` and `fix addforce` commands. In the majority of real-world cases, however, the same types of forcing are relevant, including the air–ice and water–ice

tangential and normal stresses (skin and form drag), and the Coriolis force. Also, in most cases, standard formulae are used for these forces. Thus, in order to make the model configuration easier, these formulae have been implemented in the sea-ice code and can be used with proper `fix` commands that accept respective coefficients, constant or variable (e.g., the latitude or the friction coefficients) as parameters. The implemented formulae are given in Section 6 of the main paper (Herman, 2015). The Coriolis force is handled by the `fix seaice/coriolis` command, which accepts only one argument – latitude (constant or variable). The forces due to the wind and currents are handled by the `fix seaice/wind` and the `fix seaice/current` commands, respectively. Additionally, the second of these commands may include the effects due to an oscillating surface current.
The list of external-forcing types implemented as fixes can be easily extended by copying the code of one of the implemented forcing (`fix_seaice_*.h` and `fix_seaice_*.cpp` files) and making necessary changes to adjust them to a new forcing type.
Certain properties of sea ice, atmosphere and ocean, occuring in the expressions for the above-described forces, are treated as global model parameters, i.e., they have constant values during the whole computation. These include the air density, water density, and surface-waves characteristics. They must be defined in the configuration files; otherwise, the programme exits with an error message. If necessary, i.e., if spatial and/or temporal variability of any of these variables is to be taken into account, it can be achieved easily with little programming, in a way analogous to the way other variables are treated.
**Relevant files**: `fix_seaice_wind.*`, `fix_seaice_current.*`, `fix_seaice_coriolis.*`, `fix_seaice_waves.*`
**Relevant commands**: `fix seaice/wind`, `fix seaice/current`, `fix seaice/coriolis`, `fix seaice/waves`

### 2.2.5 Elastic bonds for disks

#### General bond-related issues

The treatment of bonds in the modeled system has two interrelated aspects. First, atoms must be able to store information about 'their' bonds and include bond-related forces and torques in equations describing their motion. Second, bonds themselves must be able to store information about atoms they connect and to calculate stresses acting on them.
The usage of bonds requires a hybrid atom type (`atom_style hybrid disk bond/gran/disk`), where the second argument, that cannot be used separately, is responsible for handling any additional, bond-related variables. Two very important of those variables are `atom->nbondtypes` and `atom->bond_per_atom`, both set in the constructor of the `AtomVecBondGranDisk` class (file `atom_vec_bond_gran_disk.cpp`). The first describes the number of different bond types that the given atom can handle (e.g., bonds with different material properties, see below); the second – the maximum allowed number of bonds connected to that single atom. For two-dimensional systems, relevant to sea ice problems, the highest possible number of neighbors that a disk-shaped atom can have equals 6 for monodisperse systems. However, this number increases fast with increasing polydispersity and can be very high for systems with a very wide grain size distribution. In the present version of the toolbox, `bond_per_atom` has the same value for all atoms (even though, obviously, the number of potential neighbors of an atom increases with its perimeter, and thus radius).
Bonds themselves are handled by the `BondGranDisk` class. A crucial number that has to be set in the constructor of this class is the number of elements (`n_granhistory`) on the list (`bondhistlist`) that stores all relevant information about a bond from previous time steps, as well as any additional information needed by the given bond model. For example, in 3D systems this number equals at least 12 (6 forces, 6 torques). In the sea ice toolbox, in 2D version with random-thickness bonds it equals 6 (4 forces, 1 torque, 1 bond thickness), and in quasi-3D version with random-thickness bonds – 11 (5 forces, 5 torques, and 1 bond thickness).

Obviously, memory considerations suggest adjusting the size of the lists to the requirements of the modeled system. On the other hand, from the point of view of keeping the code consistent and facilitating further development of the model, it is reasonable that, e.g., the two components of the tangential torque have the same index on the list independently on the particular case. To this end, a solution was chosen in which the elements of `bondhistlist` are referred to by indices `j0`, `j1`, ..., so that for example the components of the tangential torque are stored always under `j9` and `j10`. In this way, the same variable can be referenced always by the same index, and the class constructors handle 'packing' of these indices into `bondhistlist`.
**Relevant files**: `atom_vec_bond_gran_disk.*`, `bond_gran_disk.*`, `bond_gran_disk_waves.*`
**Relevant commands**: `atom_style hybrid disk bond/gran/disk`, `bond_style gran/disk`

#### Bond properties

Contrary to bonds in the default granular model based on spherical particles, bonds connecting disk-shaped particles have cuboidal shape, i.e., they are described by length, width and thickness. Their moments of inertia and area moments of inertia have to be calculated accordingly. In the present version of the toolbox, the width $2R_{ij}$ and length $b_{ij}$ of any individual bond is calculated from the radii of atoms this bond connects ($r_i$, $r_j$):

$$R_{ij} = \lambda_R \min\{r_i, r_j\}, \tag{2.1}$$
$$b_{ij} = \lambda_b (r_i + r_j). \tag{2.2}$$

If different values of $\lambda_R$ and/or $\lambda_b$ are required, this can be achieved by specifying more than one bond type with different coefficients. The bond thicknesses are drawn from a uniform distribution with a prescribed mean and width.
**Relevant files**: —
**Relevant commands**: `bond_coeff`

#### Creating and destroying bonds

There are two ways to create bonds between atoms. They can be created at selected stages of the simulation by means of the `fix bond/create/gran/disk` command. It bonds atoms of prescribed types that lie within a prescribed distance from each other. Alternatively, bonds can be read from a file with initial conditions by means of the standard LIGGGHTS `read_data` command (search for the 'Bonds' section in the documentation of that command). Both methods require that the bond coefficients are defined with the `bond_coeff` command (see above).
In a typical simulation, bonds are broken if the stresses acting on them exceed their strength. However, it is also possible to remove some or all bonds during a computation. This is handled by the `fix bond/break/gran/disk` command.
The update of the bond lists is handled by `fix bond/propagate/gran/disk`, which is turned on automatically in simulations with bonded atoms, i.e., it doesn't have to occur in the configuration script.
**Relevant files**: `fix_bond_create_gran_disk.*`, `fix_bond_break_gran_disk.*`
**Relevant commands**: `fix bond/create/gran/disk`, `read_data`, `fix bond/break/gran/disk`

### 2.2.6 Quasi-3D wave effects

Taking into account quasi-3D effects related to the curvature of the sea surface requires a hybrid atom type (`atom_style hybrid disk disk/waves`), where the second argument, that cannot be used separately, is responsible for handling any additional, wave-related variables. If this option is used, the atoms have an additional property, called `tilt`, and their `atom->tilt_flag` has to be set to 1 (file `atom_vec_disk_waves.cpp`). The forces acting on atoms still have 2 components, but torques have 3 components, enabling 3D rotation of the disks.

For simulations with bonds, `bond_style gran/disk/waves` has to be used, so that apart from the bending moment around the vertical axis, the remaining two components of the bending moment, as well as the twisting moments can be properly taken into account in the bonds equations.
The `fix enforce2d/seaice/waves` command ensures that the proper set of variables is zeroed out after each time step.
The forces resulting from the presence of waves, i.e., from the curvature of the sea surface, are calculated when the `fix seaice/waves` command is active an if the wave characteristics are defined with either `fix property/global regWaves` or `fix property/global jonswap`.
**Relevant files**: `atom_vec_disk_waves.*`, `bond_gran_disk_waves.*`, `fix_seaice_waves*.*`, `fix_enforce2d_seaice_waves.*`
**Relevant commands**: `atom_style hybrid disk disk/waves`, `fix seaice/waves`, `fix enforce2d/seaice/waves`

### 2.2.7 Known problems

Different components of DESIgn have been tested to a various degree, depending on how often they have been used in model configurations that I've been using in my research. Some components certainly have more or less serious bugs, and they are more likely to occur in less frequently used commands. For example, I usually initialize the model from input files prepared in Matlab, and therefore I almost never use the `create_atoms` and `fix bond/create` commands.
I'll make efforts to remove any bugs in the future versions of DESIgn, and I'll be grateful for any
These are the known issues in the present toolbox version 1.3:

- The `set density` command works for spheres and, in 2D, for unit-thickness disks, i.e., it is unsuitable for the sea ice model. Similarly, there is no possibility to set the thickness of ice using the `set` command, as it does not accept `thickness` as a parameter.
- The `compute pair/gran/local` command correctly outputs forces between atoms, but incorrectly assumes spherical geometry when the `contactArea` keyword is used. (This does not influence the computation itself, only the output of this particular quantity.)
- There is no disk-specific code for handling atom interaction with walls. In other words, the respective forces are calculated for a spherical geometry of the objects. This may be important for some model configurations, but in my computations I always 'made' walls out of a set of immobile ('frozen') atoms, representing, e.g., fast ice. The `fix freeze` command can be used for that purpose.
- The `fix bond/create/gran/disk` command has some problems and behaves in a rather weird way in some situations. Again, I almost never use that command, as I prefer preparing input files with initial conditions in Matlab, where I can have a better control over all relevant variables. But of course, the `fix bond/create/gran/disk` needs extensive testing and bug-fixing to make it suitable for cases in which bonds may be created during a simulation due to freezing.

# Chapter 3

# Sea ice model commands

This chapter contains a full list of commands from the sea ice toolbox. The description is identical with that in the *.html and *.txt files available in the `doc` folder.
In the description of the commands, there are many links to other parts of the LIGGGHTS documentation. They do not function properly in the pdf version. In order to be able to use these links, it is advisable to copy the *.html documentation files into the `doc` folder of LIGGGHTS.

## atom_style disk command

## atom_style hybrid disk disk/waves command

## atom_style hybrid disk bond/gran/disk command

## atom_style hybrid disk disk/waves bond/gran/disk command

**Syntax:**

```
atom_style disk
atom_style hybrid disk additional_style
```

- additional_style = *disk/waves* and/or *bond/gran/disk*

**Examples:**

```
atom_style disk
atom_style hybrid disk disk/waves
atom_style hybrid disk bond/gran/disk
atom_style hybrid disk disk/waves bond/gran/disk
```

**Description:**

These atom styles are part of the USER-SEAICE package; they are only enabled if LIGGGHTS was built with that package.

They define the style of atoms to use in a sea ice simulation. The general rules for the usage of these atom styles are the same as for other styles, see the description of the atom_style command.

The basic atom style for sea ice simulations, without bonds and quasi-3D surface-waves effects, is *disk*. For simulations with surface waves, the additional style *disk/waves* is required. Simulations with bonds require the additional style *bond/gran/disk*. All granular styles store atom IDs and types, coordinates, velocities, diameter, mass and angular velocity (i.e., the set of attributes of the *sphere* style). These are the additional attributes of each disk-style:

| | |
|---|---|
| *disk* | thickness |
| *bond/gran/disk* | bonds |
| *disk/waves* | tilt |

In order to use the read_data command with these atom styles, each line in the Atoms section in the input file has to contain the following variables:

For *atom_style disk*: 8 columns (atom ID, type, diameter, thickness, density, 3 spatial coordinates).

For *atom_style hybrid disk bond/gran/disk*: 9 columns (atom ID, type, 3 spatial coordinates, diameter, thickness, density, molecule ID).

The molecule ID is meaningless in the context of granular systems, but is used for consistency of the read_data command with other bond styles. The *disk/waves* sub-style does not require any additional data to be read from file, i.e., the Atoms section contains the same columns as with *atom_style disk*.

---

If the *bond/gran/disk* is part of the hybrid style used, the following settings are required and/or strongly recommended:

```
newton off off
communicate single cutoff value vel yes
```

Due to the first setting, forces due to pairwise and bonded interactions between atoms belonging to different processors are calculated by each processor separately, instead of being communicated between processors; see newton command for details. As a result of the second command, velocities, necessary to calculate pairwise interactions, are stored by ghosts atoms; see communicate command for details. For simulations with bonds, the value of the ghost cutoff distance *value* should be larger than the longest bond in the system, so that the bond information is correctly passed between processors. Finally, if very wide particle size distributions are used, it is necessary to set the *one* option of the neigh_modify command to a sufficiently high value, higher than the expected maximum number of neighbors a single atom can have.

**Related commands:**

read_data, pair_style, bond_style, fix_nve/disk

**Default:**

atom_style atomic

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## bond_style gran/disk command

## bond_style gran/disk/waves command

### Syntax:

```
bond_style gran/disk
bond_style gran/disk/waves
```

### Examples:

```
bond_style gran/disk
bond_style gran/disk/waves
bond_coeff 1 0.8 1.0 0.0 0.9 8e9 2.5 1 1e6 1e5 1e6
```

### Description:

These bond styles are part of the USER-SEAICE package; they are only enabled if LIGGGHTS was built with that package.

The *gran/disk* style, in combination with atom_style hybrid disk bond/gran/disk can be used in a general context to model bonded, disk-shaped granular particles in 2D. The *gran/disk/waves* style is specifically dedicated to simulation of wave-induced bending and twisting moments acting on ice floes on the sea surface. It requires atom_style hybrid disk disk/waves bond/gran/disk.

The following coefficients must be defined for each bond type via the bond_coeff command as in the example above. The syntax of the *bond_coeff* command in this case is:

```
bond_coeff N lambdaR thb delthb lambdab Eb lambdans breakmode sigmamaxc
sigmamaxt taumax
```

where:

- N = bond type
- lambdaR = bond-radius coefficient (Rij=lambdaR*min(ri,rj); number between 0 and 1)
- thb = mean bond thickness (in m)
- delthb = width of the bond-thickness distribution (in m)
- lambdab = bond-length coefficient (bij=lambdab*(ri+rj); number between 0 and 1)
- Eb = bond elastic modulus (in Pa)
- lambdans = ratio of the bond normal and shear stiffness kn/kt
- breakmode = mode of bond breaking
- sigmamaxc = bond strength (max normal stress in compression; in Pa)
- sigmamaxt = bond strength (max normal stress in tension; in Pa)

- taumax = bond strength (max shear stress; in Pa)

At present, only *breakmode*=1 is allowed, which means stress-induced bond breaking that occurs if at least one of the three breaking criteria is fullfilled. The bond-thickness distribution is uniform, with mean equal *thb* and width equal *delthb* (may be zero).

A constant damping coefficient of 0.995 is used in the formulae for the forces and torques acting on the bonds.

---

If granular bonds are used, the following settings are required and/or strongly recommended:

```
newton off off
communicate single cutoff value vel yes
special_bonds lj/coul 0 1 1 extra Ne
```

Due to the first setting, forces due to pairwise and bonded interactions between atoms belonging to different processors are calculated by each processor separately, instead of being communicated between processors; see newton command for details. As a result of the second command, velocities, necessary to calculate pairwise interactions, are stored by ghosts atoms; see communicate command for details. For simulations with bonds, the value of the ghost cutoff distance *value* should be larger than the longest bond in the system, so that the bond information is correctly passed between processors. The third command is an artefact from other bond types, in which interactions may be computed not only between atoms directly connected with a bond, but also between atoms connected via other atoms; in the case of granular bonds, changing the weights for indirect interactions from 1 to 0 doesn't influence the results, but unnecessarily increases the amount of memory; the same effect occurs if the *special_bonds* command is not used at all; the value *Ne* specifies the number of extra space saved for bonds that may be created during the simulation with the fix bond/create/gran/disk command; it may be zero if bonds are read from an input file at the beginning of the simulation and the information about the maximum number of neighbors is read from that file.

**Related commands:**

atom_style hybrid disk bond/gran/disk, bond_coeff, fix bond/create/gran/disk, read_data

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## compute bond/gran/disk/local command

### Syntax:

```
compute ID group-ID bond/gran/disk/local keyword1 keyword2 ...
```

- ID, group-ID are documented in compute command
- bond/gran/disk/local = style name of this compute command
- one or more keywords may be appended
- keyword = *length* or *thickness* or *forcen* or *forcet* or *torquen* or *torqueth* or *torquetz*
- *length* = bond length
- *thickness* = bond thickness
- *forcen* = amplitude of the normal force acting on the bond
- *forcet* = amplitude of the tangential force acting on the bond
- *torquen* = amplitude of the normal torque acting on the bond
- *torqueth* = amplitude of the horizontal tangential torque acting on the bond
- *torquetz* = amplitude of the vertical tangential torque acting on the bond

### Examples:

```
compute 1 all bond/gran/disk/local thickness
compute 1 all bond/gran/disk/local length forcen forcet torquetz
```

### Description:

This command is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It defines a computation that calculates properties of individual bond interactions for granular bonds and disk-shaped atoms (*atom_style hybrid disk bond/gran/disk* or *atom_style hybrid disk disk/waves bond/gran/disk* and *bond_style gran/disk* or *bond_style gran/disk/waves*). The number of datums generated, aggregated across all processors, equals the number of bonds in the system, modified by the group parameter as explained below.

The keywords *torquen* and *torqueth* are allowed only in computations with waves, as they describe moments due to twisting of the bonds and their bending out of the horizontal xy-plane.

The local data stored by this command is generated by looping over all the atoms owned on a processor and their bonds. A bond will only be included if both atoms in the bond are in the specified compute group. Any bonds that have been broken are not included.

Note that as atoms migrate from processor to processor, there will be no consistent ordering of the entries within the local vector or array from one timestep to the next. The only consistency

that is guaranteed is that the ordering on a particular timestep will be the same for local vectors or arrays generated by other compute commands. For example, bond output from the compute property/local command can be combined with data from this command and output by the dump local command in a consistent way. Here is an example of how to do this:

```
compute 1 all property/local batom1 batom2 btype
compute 2 all bond/gran/disk/local forcen forcet
dump 1 all local 1000 tmp.dump index c_1[1] c_1[2] c_1[3] c_2[1] c_2[2]
```

**Output info:**

See the documentation of the compute bond/local for details.

The amplitude of the forces and torques is calculated differently for purely two-dimensional simulations and for quasi-three-dimensional simulations.

**Restrictions:** none

**Related commands:**

compute bond/local, dump local, compute property/local

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## compute erotate/disk command

### Syntax:

```
compute ID group-ID erotate/disk
```

- ID, group-ID are documented in compute command
- erotate/disk = style name of this compute command

### Examples:

```
compute 1 all erotate/disk
```

### Description:

This command is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It defines a computation that calculates the rotational kinetic energy of a group of disk-shaped particles. The only difference in respect to the compute erotate/sphere command concerns the moments of inertia used to calculate the rotational energy. For a disk with radius r, thickness h and mass m, the moment of inertia around its symmetry axis equals Iz = 1/2 mr^2, and the moment of inertia around the remaining two axes equals Ix = Iy = 1/12 m(3r^2+h^2).

### Output info:

This compute calculates a global scalar that can be used by any command that uses a global scalar value from a compute as input. See Section_howto 15 for an overview of LAMMPS output options.

### Restrictions:

This compute requires that atoms store a set of properties as defined by the atom_style disk command. All particles in the group must be finite-size disks.

### Related commands:

compute erotate/sphere

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## compute erotate/disk/atom command

### Syntax:

```
compute ID group-ID erotate/disk/atom
```

- ID, group-ID are documented in compute command
- erotate/disk/atom = style name of this compute command

### Examples:

```
compute 1 all erotate/disk/atom
```

### Description:

This command is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It defines a computation that calculates the rotational kinetic energy for each disk-shaped particle in a group. The only difference in respect to the compute erotate/sphere/atom command concerns the moments of inertia used to calculate the rotational energy. For a disk with radius r, thickness h and mass m, the moment of inertia around its symmetry axis equals Iz = 1/2 mr^2, and the moment of inertia around the remaining two axes equals Ix = Iy = 1/12 m(3r^2+h^2).

The value of the rotational kinetic energy will be 0.0 for atoms not in the specified compute group.

### Output info:

This compute calculates a per-atom vector, which can be accessed by any command that uses per-atom values from a compute as input. See Section_howto 15 for an overview of LAMMPS output options.

### Restrictions:

This compute requires that atoms store a set of properties as defined by the atom_style disk command. All particles in the group must be finite-size disks.

### Related commands:

dump custom, compute erotate/sphere/atom

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## fix bond/break/gran/disk command

### Syntax:

```
fix ID group-ID bond/break/gran/disk Nevery bondtype Rmax keyword values ...
```

- ID, group-ID are documented in fix command
- bond/break/gran/disk = style name of this fix command
- Nevery = attempt bond breaking every this many steps
- bondtype = type of bonds to break
- Rmax = bond longer than Rmax can break (distance units)
- zero or more keyword/value pairs may be appended to args
- keyword = *prob*
- *prob* values = fraction seed
- fraction = break a bond with this probability if otherwise eligible
- seed = random number seed (positive integer)

### Examples:

```
fix 5 all bond/break/gran/disk 1 1 2.0 prob 0.5 49829
```

### Description:

This fix is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It breaks bonds between pairs of atoms during a simulation according to specified criteria. It is a modification of fix bond/break command (part of the MC package), modified for disk-shaped atoms with bond_style gran/disk. Once the bond is broken it will be permanently deleted.

Typically, individual bonds are broken and deleted during a simulation if their breaking criteria, specified in the bond_coeff command, are fullfilled. This command may be used if, for some reason, additional removal of bonds is required, for example to imitate sea ice melting (in which case, the breaking probability may depend on a global variable representing temperature). The parameter *Rmax* is an artefact of other, molecular bond types; it may be set to zero in order to remove its influence from the list of breaking criteria listed below. Note also that it is calculated as a distance between the edges of the bonded atoms, not between their centers.

A check for possible bond breakage is performed every *Nevery* timesteps. If two bonded atoms I,J are further than a distance *Rmax* of each other, if the bond is of type *bondtype*, and if both I and J are in the specified fix group, then I,J is labeled as a "possible" bond to break.

For details of the algorithm and the usage of this command, see the documentation of fix bond/break.

---

**Restart, fix_modify, output, run start/stop, minimize info:**

See the documentation of fix bond/break.

**Restrictions:**

**Related commands:**

atom_style hybrid disk bond/gran/disk, bond_style gran/disk, fix bond/create/gran/disk

**Default:**

The option defaults are prob = 1.0.

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## fix bond/create/gran/disk command

### Syntax:

```
fix ID group-ID bond/create/gran/disk Nevery itype jtype Rmin bondtype maxbond
keyword values ...
```

- ID, group-ID are documented in fix command
- bond/create/gran/disk = style name of this fix command
- Nevery = attempt bond creation every this many steps
- itype,jtype = atoms of itype can bond to atoms of jtype
- Rmin = 2 atoms separated by less than Rmin can bond (distance units)
- bondtype = type of created bonds
- maxbond = max # of bonds a single atom can have
- zero or more keyword/value pairs may be appended to args
- keyword = *iparam* or *jparam* or *prob*
- *iparam* values = maxbond, newtype
- maxbond = max # of bonds of bondtype the itype atom can have
- newtype = change the itype atom to this type when maxbonds exist
- *jparam* values = maxbond, newtype
- maxbond = max # of bonds of bondtype the jtype atom can have
- newtype = change the jtype atom to this type when maxbonds exist
- *prob* values = fraction seed
- fraction = create a bond with this probability if otherwise eligible
- seed = random number seed (positive integer)

### Examples:

```
fix 5 all bond/create/gran/disk 1 1 1 5.0 1 7
```

### Description:

This fix is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It creates bonds between pairs of disk-shaped atoms during a simulation according to specified criteria. It is a modification of fix bond/create command (part of the MC package), modified for disk-shaped atoms with bond_style gran/disk .

A check for possible new bonds is performed every *Nevery* timesteps. If two atoms I,J are within a distance *Rmin* of each other, if I is of atom type *itype*, if J is of atom type *jtype*, if both I and J are in the specified fix group, if a bond does not already exist between I and J, and if both I and J meet their respective *maxbond* requirement (explained below), then I,J is labeled as a "possible"

bond pair. Note that the distance is measured between the edges (not between centers) of the disks.

For details of the algorithm and the usage of this command, see the documentation of fix bond/create.

---

**Restart, fix_modify, output, run start/stop, minimize info:**

See the documentation of fix bond/create.

**Restrictions:**

See the documentation of fix bond/create.

**Related commands:**

atom_style hybrid disk bond/gran/disk, bond_style gran/disk, fix bond/break/gran/disk

**Default:**

The option defaults are iparam = (0,itype), jparam = (0,jtype), and prob = 1.0.

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

---

## fix enforce2d/seaice/waves command

### Syntax:

```
fix ID group-ID enforce2d/seaice/waves
```

- ID, group-ID are documented in fix command
- enforce2d/seaice/waves = style name of this fix command

### Examples:

```
fix 5 all enforce2d/seaice/waves
```

### Description:

This fix is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It should be used in sea ice simulations with quasi-3D wave effects, i.e., with the atom_style hybrid disk disk/waves and fix seaice/waves commands. Analogously to fix enforce2d, it zeroes out the z-dimension velocity and force on each atom in the group. However, to allow for wave-induced deviations of the disks orientation from the horizontal plane, this fix does not zero out the x- and y-components of the angular velocity, momentum and torque.

---

### Restart, fix_modify, output, run start/stop, minimize info:

See documentation of the fix enforce2d command.

**Restrictions:** none

**Related commands:** fix enforce2d, atom_style hybrid disk disk/waves, fix seaice/waves

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

---

## fix nve/disk command

### Syntax:

```
fix ID group-ID nve/disk
```

- ID, group-ID are documented in fix command
- nve/disk = style name of this fix command

### Examples:

```
fix 1 all nve/disk
```

### Description:

This fix is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

Similarly to the fix nve/sphere command, it performs constant NVE integration to update position, velocity, and angular velocity for particles in the group each timestep, but instead of spherical, disk-shaped particle geometry is assumed.

See documentation of the fix nve/sphere and fix nve commands for details.

---

### Restart, fix_modify, output, run start/stop, minimize info:

See documentation of the fix nve/sphere command.

### Restrictions:

This fix requires that atoms store a set of properties characterizing disk-shaped particles (as defined by the atom_style disk command).

### Related commands:

fix nve, fix nve/sphere

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## fix seaice/coriolis command

### Syntax:

```
fix ID group-ID seaice/coriolis phi keyword value ...
```

- ID, group-ID are documented in fix command
- seaice/coriolis = style name of this fix command
- phi = latitude
- phi can be a variable (see below)
- zero or more keyword/value pairs may be appended to args
- keyword = *region*
- *region* value = region-ID
- region-ID = ID of region atoms must be in to have added force

### Examples:

```
fix f1 all seaice/coriolis  60.0
fix f1 all seaice/coriolis -70.0 region regID
```

### Description:

This fix is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It adds Coriolis force to the x and y components of force for each disk-shaped atom in the group.

It is assumed that the Coriolis force is horizontal, i.e., it does not contribute to the force along the z-axis. It is also assumed that the Coriolis force does not contribute to the torque acting on the disks.

This fix requires a global property, *rhoIce*, that can be set with the fix property/global command:

```
fix pg1 all property/global rhoIce scalar 910.0
```

No default value for this constant is predefined and the lack of the above definition in the input script will produce an error.

The values of *phi* (in degrees) are positive in the northern hemisphere and negative in the southern hemisphere. *Phi* can be specified as an equal-style or atom-style variable. If the value is a variable, it should be specified as v_name, where name is the variable name. In this case, the variable will be evaluated at each timestep. For more details concerning specifying variable forcing, see the documentation of the fix addforce command.

If the *region* keyword is used, the atom must also be in the specified geometric region in order to have the Coriolis force added to it.

---

**Restart, fix_modify, output, run start/stop, minimize info:**

See the documentation of the fix addforce command.

**Restrictions:** none

**Related commands:**

fix addforce, fix seaice/current, fix seaice/waves, fix seaice/wind

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## fix seaice/current command

### Syntax:

```
fix ID group-ID seaice/current Ux Uy Cds Cdf keyword value ...
```

- ID, group-ID are documented in fix command
- seaice/current = style name of this fix command
- Ux,Uy = x and y components of surface current (in m/s)
- Cds,Cdf = ice-water skin and form drag coefficients
- any of Ux,Uy,Cds,Cdf can be a variable (see below)
- zero or more keyword/value pairs may be appended to args
- keyword = *wavecur* = wave-induced surface current
- *wavecur* value = *jonswap* or *regWaves*
- *jonswap* = Jonswap spectrum
- *regWaves* = regular waves
- keyword = *region*
- *region* value = region-ID
- region-ID = ID of region atoms must be in to have added force

### Examples:

```
fix NW all seaice/current 0.25 -0.25 0.002 0.001
fix Nwv all seaice/current 0.0 -0.2 0.002 0.0018 wavecur regWaves
fix periodic all seaice/current v_acost 0.0 0.002 0.002 region regID
```

### Description:

This fix is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It adds current-generated force and torque to forces/torques acting on each disk-shaped atom in the group.

It is assumed that the current is horizontal, i.e., it does not contribute to the force along the z-axis and to the torque along the x- and y-axes.

The skin drag acting on a given disk is proportional to its upper surface area. The form drag is proportional to the disk diameter times its freeboard, calculated from its thickness and relative density. In the code, a quadratic formula is used for the forces and a linearized formula for the torque.

Any of the 4 quantities defining the current speed components and drag coefficients can be specified as an equal-style or atom-style variable. If the value is a variable, it should be specified as v_name, where name is the variable name. In this case, the variable will be evaluated each timestep. For more details concerning specifying variable forcing, see the documentation of the fix addforce command.

The *wavecur* option makes it possible to take into account horizontal components of the wave-induced surface current (i.e., the oscillatory motion at the mean water surface, not the mean drift due to waves).

Without the *wavecur* option, this fix requires one global variable, defining the value of water density, that can be set with the fix property/global command:

```
fix pg1 all property/global rhoWater scalar 1025.0
```

If the *wavecur* option is used, an additional global variable is required, specifying the wave characteristics, for example:

```
fix pg2 all property/global regWaves matrix 3 0.75 0.15 0.15 25.0 18.0 20.0
0.0 10.0 -10.0 0.0 20.0 -30.0
```

for a superposition of 3 regular waves, or:

```
fix pg3 all property/global jonswap vector 0.5 12.0 0.0 0.33 20.0 20 0.05 0.5
5 -20.0 20.0
```

for a Jonswap spectrum (see fix seaice/waves for a detailed description of the syntax of these commands and the meaning of their arguments). No default values for these constants are predifined and the lack of the above definitions in the input script will produce an error.

If the *region* keyword is used, the atom must also be in the specified geometric region in order to have the force added to it.

---

**Restart, fix_modify, output, run start/stop, minimize info:**

See the documentation of the fix addforce command.

**Restrictions:** none

**Related commands:**

fix addforce, fix seaice/coriolis, fix seaice/wind, fix seaice/waves

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## fix seaice/waves command

### Syntax:

```
fix ID group-ID seaice/waves wavetype keyword value
```

- ID, group-ID are documented in fix command
- seaice/waves = style name of this fix command
- wavetype = type of the wave input (*jonswap* or *regWaves*)
- zero or one keyword/value pair may be appended to args
- keyword = *region*
- *region* value = region-ID
- region-ID = ID of region atoms must be in to have added force

### Examples:

```
fix f3v all property/global jonswap vector 0.75 20. 0. 0.33 50. 20 0.05 0.3 5
-14. 14.
fix f3  all seaice/waves jonswap
fix w2v all property/global regWaves matrix 2 0.75 0.15 25. 18. -5. 5. 0. 20.
```

### Description:

This fix is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It adds wave-generated force and torque (due to the curvature of the sea surface) to forces/torques acting on each disk-shaped atom in the group.

It is assumed that the wave-induced torque acts in the horizontal plane and results from unbalanced buoyancy forces present if the upper surface of a disk-shaped particle is not parallel to the local sea surface.

This fix requires atom_style hybrid disk disk/waves and, if bonds are used, bond_style gran/disk/waves.

The *wavetype* option is used to specify the type of wave forcing used in the simulations: either *jonswap* for a Jonswap spectrum or *regWaves* for a superposition of regular waves. The wave characteristics for these options have to be specified with a global property *jonswap* or *regWaves*, respectively, as in the examples above. The general syntax of these commands is:

```
fix ID group-ID property/global regWaves matrix N a1 a2 ... aN T1 T2 ... TN
th1 th2 ... thN ph1 ph2 ... phN
```

```
fix ID group-ID property/global jonswap vector Hs Tp thm gamma m Nf fmin fmax
Nd dmin dmax
```

- N = number of elementary waves
- a1, a2, ..., aN = a list of N wave amplitudes (in m)
- T1, T2, ..., TN = a list of N wave periods (in s)
- th1, th2, ..., thN = a list of N wave directions (in degrees; -180<=thn<=180)
- ph1, ph2, ..., phN = a list of N wave phase angles (in degrees; -180<=phn<=180)
- Hs = significant wave height (in m)
- Tp = peak period (in s)
- thm = mean wave direction (in degrees; -180<=thm<=180)
- gamma = peak enhancement factor
- m = exponent in the directional distribution (cos^m(th=thm))
- Nf = number of frequencies
- fmin = minimum frequency (in Hz)
- fmax = maximum frequency (in Hz)
- Nd = number of directions
- dmin = minimum angle (in degrees; -180<=dmin<=180)
- dmax = maximum angle (in degrees; -180<=dmax<=180; dmax>dmin)

In the case of the *jonswap* option, the spectrum specified by the set of parameters *Hs*, *Tp*, *thm*, *gamma* and *m* is expressed as a set of *Nf***Nd* sine waves with random phases and with frequencies logarithmically distributed between *fmin* and *fmax*.

The values *th1*, *th2*, ..., and *thm* describe directions to which waves are propagating; zero means waves propagating to the 'east' (positive x-direction), 90 degrees to the 'north' (positive y-direction), -90 degrees to the 'south', and so on.

No default values for the *regWaves* or *jonswap* global properties are predifined and the lack of the above definitions in the input script will produce an error.

If the *region* keyword is used, the atom must also be in the specified geometric region in order to have the force added to it.

---

**Restart, fix_modify, output, run start/stop, minimize info:**

See the documentation of the fix addforce command.

**Restrictions:** none

**Related commands:**

fix addforce, fix seaice/coriolis, fix seaice/current, fix seaice/wind

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## fix seaice/wind command

### Syntax:

```
fix ID group-ID seaice/wind U10x U10y Cds Cdf keyword value ...
```

- ID, group-ID are documented in fix command
- seaice/wind = style name of this fix command
- U10x,U10y = x and y components of wind speed (in m/s)
- Cds,Cdf = air-ice skin and form drag coefficients
- any of U10x,U10y,Cds,Cdf can be a variable (see below)
- zero or more keyword/value pairs may be appended to args
- keyword = *region*
- *region* value = region-ID
- region-ID = ID of region atoms must be in to have added force

### Examples:

```
fix NW all seaice/wind 5.0 -5.0 0.002 0.001
fix storm all seaice/wind v_ux v_uy 0.002 0.002 region regID
```

### Description:

This fix is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It adds wind-generated force to the x and y components of force for each disk-shaped atom in the group.

It is assumed that the wind is horizontal, i.e., it does not contribute to the force along the z-axis. It is also assumed that the wind does not contribute to the torque acting on the disks.

This fix requires two global variables, defining the values of water density and air density, that can be set with the fix property/global commands:

```
fix m2 all property/global rhoWater scalar 1025.0
fix m3 all property/global rhoAir scalar 1.27
```

No default values for these constants are predefined and the lack of the above definitions in the input script will produce an error.

The skin drag acting on a given disk is proportional to its upper surface area. The form drag is proportional to the disk diameter times its freeboard, calculated from its relative density and thickness.

Any of the 4 quantities defining the wind speed components and drag coefficients can be specified as an equal-style or atom-style variable. If the value is a variable, it should be specified as v_name, where name is the variable name. In this case, the variable will be evaluated each timestep, and its value(s) used to determine the instantaneous wind speed/drag coefficient. For more details concerning specifying variable forcing, see the documentation of the fix addforce command.

If the *region* keyword is used, the atom must also be in the specified geometric region in order to have force added to it.

---

**Restart, fix_modify, output, run start/stop, minimize info:**

See the documentation of the fix addforce command.

**Restrictions:** none

**Related commands:**

fix addforce, fix seaice/coriolis, fix seaice/current, fix seaice/waves

**Default:** none

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## gran model hertz/disk

### Syntax:

```
pair_style gran model hertz/disk [other model_type/model_name pairs as
described here ] keyword values
```

- zero or more keyword/value pairs may be appended

```
  limitForce values = 'on' or 'off'
    on = ensures that the normal force is never attractive (an artefact that
can occur at the end of a collision).
    off = standard implementation that might lead to attractive forces.
  tangential_damping values = 'on' or 'off'
    on = activates tangential damping
    off = no tangential damping
```

### Description:

This pair style is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It requires *atom_style disk* and it is analogous to *gran model hertz* with the difference that the formulae used to calculate the interaction forces between atoms take into account disk-shaped geometry of the atoms. See gran model hertz for a documentation of the Hertzian model for spherical particles, and the Supplementary Note S1 to Herman (2015) for the derivation of the equations.

As in the case of other granular models, it is mandatory to use multiple fix property/global commands with this granular model:

```
fix id all property/global youngsModulus peratomtype value_1 value_2 ...
    (value_i=value for Youngs Modulus of atom type i)
fix id all property/global poissonsRatio peratomtype value_1 value_2 ...
    (value_i=value for Poisson ratio of atom type i)
fix id all property/global coefficientRestitution peratomtypepair n_atomtypes
value_11 value_12 .. value_21 value_22 .. .
    (value_ij=value for the coefficient of restitution between atom type i and
j; n_atomtypes is the number of atom types you want to use in your simulation)
fix id all property/global coefficientFriction peratomtypepair n_atomtypes
value_11 value_12 .. value_21 value_22 .. .
    (value_ij=value for the (static) coefficient of friction between atom type
i and j; n_atomtypes is the number of atom types you want to use in your
simulation)
```

### Related commands:

atom_style disk, gran model hertz

**Default:**

*tangential_damping* = 'on' *limitForce* = 'off'

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

## gran model hertz/stiffness/disk

### Syntax:

```
pair_style gran model hertz/stiffness/disk [other model_type/model_name pairs
as described here ] keyword values
```

- zero or more keyword/value pairs may be appended

```
  limitForce values = 'on' or 'off'
    on = ensures that the normal force is never attractive (an artefact that
can occur at the end of a collision).
    off = standard implementation that might lead to attractive forces.
  tangential_damping values = 'on' or 'off'
    on = activates tangential damping
    off = no tangential damping
```

### Description:

This pair style is part of the USER-SEAICE package; it is only enabled if LIGGGHTS was built with that package.

It requires *atom_style disk* and it is analogous to *gran model hertz/stiffness* with the difference that the formulae used to calculate the interaction forces between atoms take into account disk-shaped geometry of the atoms. See gran model hertz/stiffness for a documentation of the Hertzian model for spherical particles and the Supplementary Nore S1 to Herman (2015) for the derivation of the equations.

As in the case of other granular models, it is mandatory to use multiple fix property/global commands with this granular model:

```
fix id all property/global kn peratomtypepair n_atomtypes value_11 value_12 ..
value_21 value_22 .. .
    (value_ij=value for k_n between atom type i and j; n_atomtypes is the
number of atom types you want to use in your simulation)
fix id all property/global kt peratomtypepair n_atomtypes value_11 value_12 ..
value_21 value_22 .. .
    (value_ij=value for k_t between atom type i and j; n_atomtypes is the
number of atom types you want to use in your simulation)
fix id all property/global gamman peratomtypepair n_atomtypes value_11
value_12 .. value_21 value_22 .. .
    (value_ij=value for gamma_n between atom type i and j; n_atomtypes is the
number of atom types you want to use in your simulation)
fix id all property/global gammat peratomtypepair n_atomtypes value_11
value_12 .. value_21 value_22 .. .
    (value_ij=value for gamma_t between atom type i and j; n_atomtypes is the
number of atom types you want to use in your simulation)
```

### Related commands:

atom_style disk, gran model hertz/stiffness

**Default:**

*tangential_damping* = 'on' *limitForce* = 'off'

**(Herman)** Agnieszka Herman, Geosci. Model Devel., submitted (2015).

# Chapter 4

# Examples and utilities

## 4.1 Examples

The `examples` folder of DESIgn contains input files for a number of cases that illustrate the functioning of various elements of the model. Some of them are discussed in the main paper presenting the model (Herman, 2015), others – those, in which bonds are not used – were presented in Herman (2013a,b,c).
The presented examples intentionally use relatively small numbers of grains (and, consequently, bonds), so that they don't require extensive computational resources and can be treated as test cases and as a starting point for more advanced and more realistic model settings.
This is the list of examples included with the code of the toolbox:

`convergence` – **biaxial compression without bonds**
: Simulation of a set of $N$ grains (floes) within a shrinking periodic domain. It can be used to study the jamming phase transition in ice (as in Herman, 2013b), but is also very useful for preparing initial conditions for other, more complex simulations, as it provides an easy method of obtaining high ice concentrations (by first shrinking the domain, then letting the system relax with the forcing turned off).

`tensilebreak` – **uniaxial tensile stress**
: Simulation of a rectangular sample of densely packed, fully connected grains subject to a tensile stress obtained by 'freezing' the grains along the lower boundary and moving the grains along the opposite boundary with a prescribed, monotonously increasing velocity until terminal failure of the material.

`compressivebreak` – **uniaxial compressive stress**
: Simulation similar to the previous one, but under compressive stress obtained by changing the sign of the velocity (and thus strain).

`shearbreak` – **shear stress**
: Simulation similar to the previous ones, but with shear strain instead of the normal strain (configuration analogous to that in Herman, 2013c).

`wind` – **cluster formation without bonds due to wind and current**
: Simulation of a set of grains (floes) within a rectangular, periodic model domain, under the action of wind, current and the Coriolis force (configuration analogous to that in Herman, 2013b).

`wavebreak` – **ice breaking due to surface waves**
: Simulation of a rectangular sample of densely packed, fully connected grains subject to stresses related to the curvature of the sea surface.

## 4.2 Utilities

The functions and scripts included in the `utils` folder of the sea ice toolbox can be divided into two categories, as described in the two sections below.

### 4.2.1 Generation of initial conditions

`LIGGGHTSinit.m`
: Function to prepare initial conditions for sea ice simulations, i.e., files that can be read with the `read_data` command. It produces a sample of $N$ disks with prescribed distributions of radii and velocities, randomly distributed within a prescribed domain, without or with bonds, according to the information provided by the user.

`LIGGGHTSinit_fromdump.m`
: Function similar to the previous one, but used to prepare initial conditions from the results of previous LIGGGHTS simulation.

`example_usage_of_LIGGGHTSinit.m`
: Script presenting example usage of the two above functions.

### 4.2.2 Analysis and visualization of the results

`readdump.m`
: Function to read files produced with the `dump` command (bonds, pairs, or per-atom quantities, as described in the LIGGGHTS documentation).

`read_thermo.m`
: Function to read files produced with the `thermo` and `thermo_style` commands (global quantities).

`assembleresults.m`
: Function to combine data read by the two previous functions, for example for plotting purposes.

`principal_stress.m`
: Function to calculate the components of principal stress, the principal angle etc. from the stress output of LIGGGHTS.

`plotfloepositions.m`
: Function to generate figure with positions of the grains, colored according to a selected characteristics.

`floecount.m`
: Function to calculate the number and sizes of floes based on the bond information.

`floeshape.m`
: Function (experimental!) to estimate contours of the floes.

`example_result_analysis.m`
: Script presenting the usage of some of the functions listed above.

# Bibliography

Herman, A. (2012). Influence of ice concentration and floe-size distribution on cluster formation in sea ice floes. *Cent. Europ. J. Phys.*, 10:715–722.

Herman, A. (2013a). Molecular-dynamics simulation of contact and force networks in fragmented sea ice under shear deformation. *Proc. 3rd Int. Conf. Particle-Based Methods*, pages 659–669. 18–20 Sep. 2013, Stuttgart, Germany.

Herman, A. (2013b). Numerical modeling of force and contact networks in fragmented sea ice. *Annals Glaciology*, 54:114–120.

Herman, A. (2013c). Shear-jamming in two-dimensional granular materials with power-law grain-size distribution. *Entropy*, 15:4802–4821.

Kloss, C. and Goniva, C. (2010). LIGGGHTS: a new open source discrete element simulation software. In *Proc. 5th Int. Conf. Discrete Element Methods, London, UK, August 2010.*

Kloss, C. and Goniva, C. (2011). LIGGGHTS – open source discrete element simulations of granular materials based on Lammps. In *Suppl. Proc.: Materials Fabrication, Properties, Characterization, and Modeling*, volume 2, pages 781–788.

Plimpton, S. (1995). Fast parallel algorithms for short-range molecular dynamics. *J. Comp. Phys.*, 117:1–19.