# A New Object-Oriented Approach for Integrating Discrete Element Method into Modelica

Christian Richter[1] Jürgen Weber[2] Florian Ohser[3] Thomas Beutlich[4]

[1]Chair of Construction Machines, TU Dresden, Germany, christian.richter1@tu-dresden.de
[2]Chair of Fluid-Mechatronic Systems, TU Dresden, Germany, weber@ifd.tu-dresden.de
[3]ESI ITI GmbH, Germany, florian.ohser@esi-group.com
[4]ESI ITI GmbH, Germany, thomas.beutlich@esi-group.com

## Abstract

In this paper a new library for co-simulation of discrete element method and Modelica models is presented. For this a component-based approach is used that allows closed modeling and visualization of discrete element systems in a modelica tool. Translation into a native DEM description language and co-simulation is done by a separate compiler and backend. Usage and functionality are shown in a simple use case of a bucket excavator digging a hole.

*Keywords: discrete element method, co-simulation, construction machines*

## 1 Introduction

Working process of construction and conveying machines is characterized by the interaction with granular materials. In order to allow prospective analysis of machine behavior under real operating conditions, coupled simulations are increasingly used. In these cases, particle-mechanical behavior is reproduced by using discrete element method (DEM). Up to now the creation and calculation of coupled simulations between system models and DEM is very expensive and time-consuming. This effort can be significantly reduced by using the new library presented in this work, which uses a new component-oriented modeling approach for discrete element systems.

### 1.1 Discrete Element Method

The discrete element method (DEM) is a numerical method for simulating the behavior and motion of large numbers of discrete, interacting objects (Cundall, 1971). In most cases, as done here, these objects are referred as particles. Basis of the method is the calculation of forces acting between the particles or between a particle and an adjacent surface. The basic calculation cycle should be explained briefly below.

After insertion every particle has an initial position and velocity. The simulation loop starts by determining all particle-particle and particle-wall contacts. After that the forces and torques acting on every particle have to be calculated. These forces result on the one hand from field forces like gravity and on the other hand from the particle deformation as a consequence of collision. For that different contact-models and force-deformation laws are used. Figure 1 shows an example of such a contact model. By summing up all single forces and torques, the translational and angular acceleration of each particle can be obtained. The last step is solving the equations of motion. For that the new positions and velocities are resolved by integrating translational and angular acceleration. The whole loop is repeated for a predetermined number of iterations.

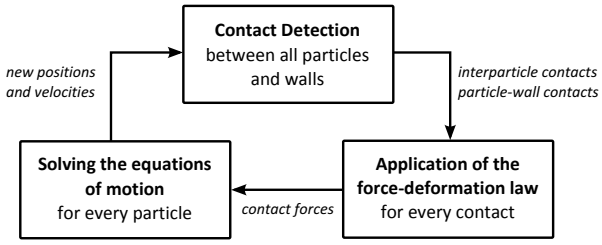


**Figure 1.** DEM Computation Loop.

### 1.2 LIGGGHTS®

One of the most used non-proprietary software applications for discrete element simulations is LIGGGHTS® (LAMMPS improved for general granular and granular heat transfer simulations) (Kloss and Goniva, 2011). Main advantages of it are:

- Open source
- Large number of available contact models
- Extensive import and export capabilities for geometry and results
- Various implementations and methods for parallelization of computation (MPI, OpenMP, CUDA)

Besides these points it also has some disadvantages:

- Command-oriented modeling-paradigm
- Complicated syntax
- Elaborate parametrization
- No graphical user interfaces

- No integrated visualization and post-processing capabilities

All these disadvantages will be eliminated with the solution presented here.

### 1.3 Earlier Solutions and Classification

Coupled simulation of Modelica-based machine models and discrete element method has been a big field of research in recent years. Several solutions have been developed till now. For a better understanding of the differences between them, classification shown in figure 2 (Geimer et al., 2006) should be used.

Core idea and principle of Modelica is to use an equation-based approach for behavioral description and linkage of different models from different physical domains. This is called a *classic simulation*. While this works fine for some domains, like hydraulics or mechanics, this won't work for discrete element systems. To ensure fast contact detection or force computation the specialization of another simulation tool is necessary.

For coupling two different simulation tools special interfaces must be developed. In 2010 we started with the software-framework *SARTURIS* providing a network based coupling of both domains(Kunze et al., 2010). Another coupling technique using functional mock-up units (FMU) was presented in 2012 (Kunze et al., 2012). Based on the functional mock-up interface, a FMU describes a non-proprietary data format containing encapsulated simulation models (Blochwitz et al., 2012). FMU's can be exported and imported by many simulation tools and used for simulation coupling. Referring to figure 2 both solutions are *co-simulations*. The biggest drawback is that distributed modeling, as well as coupling of different input and output values, is very time-consuming and error-prone.

The solution presented in this paper allows a closed modeling and an automatic coupling of DEM and Modelica. A similar approach was used in (Elmqvist et al., 2015). Additionally, the new library uses a component-based modeling paradigm for discrete element models.

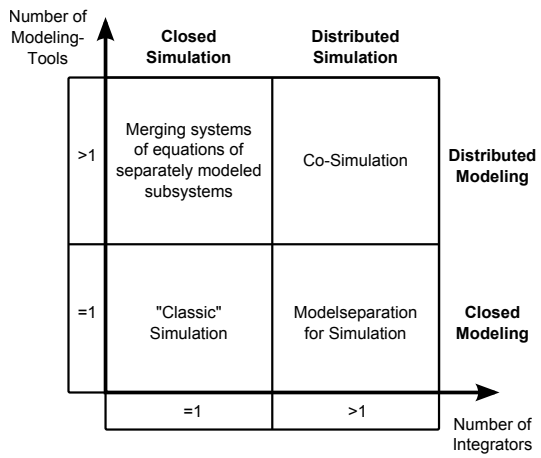


**Figure 2.** Classification of coupled simulations.

## 2 Object-oriented design for DEM

As already mentioned LIGGGHTS® follows a command-oriented modeling-paradigm. This is typical for all DEM applications. Usually the user writes an input script containing the whole simulation process. The software reads in this script and executes all commands in sequential order.

One of the core ideas of Modelica is to use an object-oriented design (OOD) for models. For transforming LIGGGHTS® functions into an OOD first an object-oriented analysis (OOA) must be done. According to (Coad and Yourdon, 1991) an object is defined as a real world entity related to the problem domain, with "crisply defined boundaries". Objects are encapsulated with attributes and behaviour. For identifying all objects it's helpful to start writing down all functionalities that should be included in future objects. After that object classes and their design parameters have to be defined fulfilling all these functionalities. One principle is that all objects should be self-explaining and easy to understand for the user. The following table shows a selection of defined classes and some of their functions.

**Table 1.** Selection of DEM object-classes and related functionalities.

| *Object* | *Functionalities* |
|---|---|
| SimulationBox | set timestep size |
| | set contact model |
| | set boundaries of spatial domain |
| | get total particle count/mass |
| SingleParticle | generate a single particle |
| | set diameter |
| | define material settings |
| ParticleSet | load saved particle configurations |
| ParticleSource | generate a particle stream |
| ParticleSink | remove particles |
| | set particle rate / mass rate |
| | define material settings |
| RigidBody | set position and velocity |
| | get forces on body |
| RegionSensor | get particle count/mass in region |

As you can see not all of these objects are real world entities, so it would be better to speak of a component-oriented than of an object-oriented design. In order to keep the terminology as simple as possible it was decided to continue speaking of an object-oriented approach.

After classes, functions and parameters are defined they can be implemented in Modelica. Figure 3 shows the structure of the new library and design of the single object models.


DOI 10.3384/ecp17132895

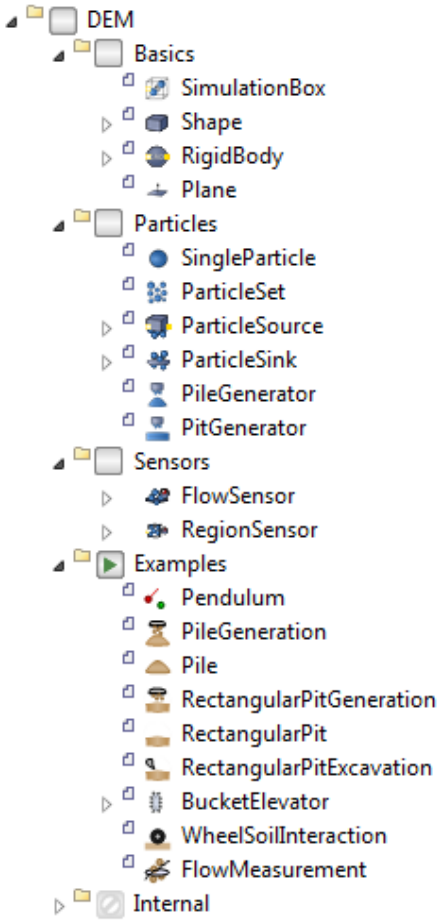


**Figure 3.** DEM Library in SimulationX®.

# 3 Simulation coupling

## 3.1 System architecture

Above library allows the closed modeling of machine and process models. In order to perform a distributed simulation, models have to be subsequently separated again. For a better understanding on how this is done figure 4 shows the system architecture of all simulation components. This structure is divided into front- and back-end.

The front-end essentially consists of the library and a material database, which will be explained more in detail in section 4.2. Each library object contains an internal network client, which is capable to connect and communicate via TCP/IP to a server.

The server is the root node of back-end-structure. It receives the messages coming from the components and forwards them to a special DEM-Slave with an attached compiler. The compiler is collects information about all elements in the model and translates them into LIGGGHTS command sequences.

LIGGGHTS itself is not used as an executable but as a shared library with a custom API. Data exchange is much more simplified this way. Furthermore we modified some basic LIGGGHTS function, e.g. for moving meshes, particles sources and sinks during simulation runtime.

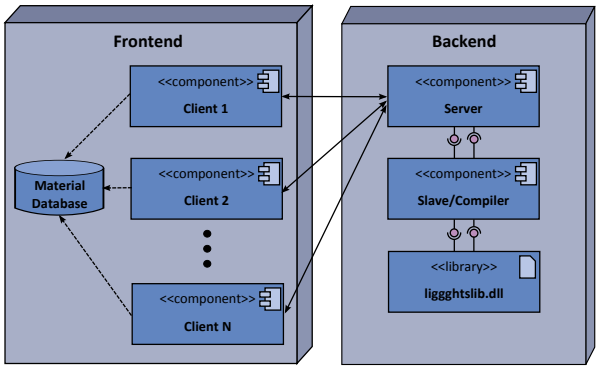


**Figure 4.** System architecture

## 3.2 Communication

For communication and data exchange between front- and back-end C-functions accessed by external objects are used. Every object has a *TcpClient*, which is responsible for connecting to the server as well as sending and receiving data. All data is stored in *DataPackages* acting like a send and receive buffer.

```
TcpClient client = TcpClient();
DataPackage outPkg = DataPackage();
DataPackage inPkg  = DataPackage();
```

During initialization all clients are connecting to the server. After connection has successful established initial data is exchanged. This may be for example some positional or geometric information.

```
parameter Boolean isConnected = false;
parameter Boolean isInitialized = false;

parameter String address = "localhost";
parameter Integer port = 1234;

initial algorithm
  if isInitialized == false then
        isConnected := connectToHost(
            client, address, port);
  end if;

  setData(outPkg, {/*integer values*/},
                  {/*real values*/},
                  {/*string values*/});

  if isConnected then
        sendPackage(client, outPkg);
        recvPackage(client, inPkg);
  end if;

  isInitialized:=true;
```

After initialization the main loop starts. Communication between front- and back-end occurs at discrete equidistant time values. For this we use a *sample*-function. At every communication event current model values are pushed to the server. After sending all output data the model waits for the data coming fom the server. At the end of the simulation loop some final data is transferred to the server.

```
parameter Real tc(quantity="Basics.Time",
    displayUnit="s") = 0.0001;
Boolean commTrigger(start=false,fixed=true)
     = sample(0, tc);

algorithm
  if isConnected then
    when commTrigger then
      // set current data to outPkg
      // send and receive packages
      // extract data from inPkg
    elsewhen terminal() then
      // send final information
    end when;
  end if;
```

# 4 Pre-processing

## 4.1 Basic simulation settings

All basic simulation and communication settings are defined in the *SimulationBox*. Similar to the *World* component in *Modelica.Mechanics.MultiBody* package every DEM model must contain one SimulationBox. This is ensured by an *outer* construct in the code.

```
outer DEM.Basics.SimulationBox simBox;
```

This way all objects have access to basic parameters like host address and port.

```
equation
  address = simBox.address;
  port = simBox.port;
```

In order to keep computation costs low it's necessary to define boundaries for the DEM space. These boundaries can be fixed (particles will be removed if they leave the spatial domain) or dynamic growing.

## 4.2 Material definitions

The parameterization of the material properties of DEM models is very complicated and presents a problem that has not been completely solved. In order to increase the operating convenience of the library, a material database has been created, which contains parameter sets for the most realistic description of different granular materials. The valid parameter sets were determined by comparing laboratory measurements and simulation. Various calibration tests were used. Among other things, the shear force, the angle of inclination as well as the transit time of different granular substances were investigated. The selection of the materials to be examined followed the possible future application areas of the total solution. Sand and gravel (construction machinery), hard coal, brown coal, iron ore and potash (mining and conveyor technology) as well as corn and wheat (agricultural machinery and food technology) were investigated.

For the representation of large rocks or boulders a function was implemented, which allows the use of Multisphere materials. In this case, a composite of several spheres is formed, which are inseparably connected to each other. This function was not provided in the original work package. It has been implemented since it means a considerable added value for the user and thus for the marketing of the final product. The interpolation of a stone by a Multisphere object is shown in figure 5.

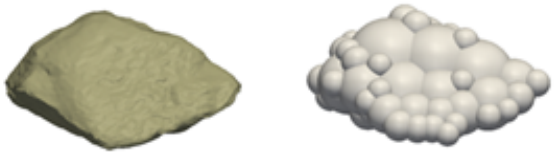

**Figure 5.** Multisphere approximation of a stone

# 5 Post-processing

## 5.1 Visualization

Besides representing time-dependent state values in diagrams, 3D visualization is an important part of modern post processing. For this the *ModelicaServices* package comes with some models for animation and visualization of certain predefined shapes such as cylinders, boxes or imported STL- and DXF-geometries. The implementation of this package can vary from one Modelica tool to another.

These capabilities are very limited to basic shapes and not sufficient for the visualization of large particle systems. Though there is an animation body for spheres, it's not very advising to use it, because for $n$ particles it would be necessary to create $n$ animation submodels. This would increase the number of internal equations and downgrade performance.

In our implementation we created a new animation body called *DEMPoints*. We propose to extend *ModelicaServices* with such a model.

For large-scale systems up to one million particles 3D representation itself takes a lot of computation costs. For that reason it's possible to switch between the options *splats*, *diamonds* or *spheres*, which supply different levels of details and performance.

## 5.2 Sensors

In discrete element simulations it's often necessary to measure particle specific values. For that we enhanced regular LIGGGHTS capabilities by some special sensor functions. Different shaped *RegionSensors* can be used to evaluate the number and mass of particles in a specific volumetric region. Sensor position and size can change during simulation runtime. It's also possible to attach sensors to rigid bodies. One use case would be the measurement of bucket filling level during the digging process of an excavator.

To check if a particle is inside a specific region we use a very simple and efficient algorithm. Consider there's a cuboid region sensor with the position vector $\mathbf{x}_S$, orientation matrix $\mathbf{R}_S$ and dimensions $lx$, $ly$ and $lz$. Now we want


DOI 10.3384/ecp17132895

to find out if a particle $P$ is inside the sensor region. First thing we have to do is calculating the absolute difference vector of the particles global position $\mathbf{x}_P$ and the position of the sensor $\mathbf{x}_S$ (eq. 1). After that we transform this absolute difference vector into the relative coordinates of the sensor (eq. 2).

$$\mathbf{x}_{PS_{abs}} = \mathbf{x}_P - \mathbf{x}_S \tag{1}$$

$$\mathbf{x}_{PS_{rel}} = \mathbf{R}_S \cdot \mathbf{x}_{PS_{abs}} \tag{2}$$

To determine if the particle is inside or not we have to check the following logic equation.

$$\begin{aligned} inside_{cuboid} = &|\mathbf{x}_{PS_{rel},x}| < 0.5 \cdot lx \wedge \\ &|\mathbf{x}_{PS_{rel},y}| < 0.5 \cdot ly \wedge \\ &|\mathbf{x}_{PS_{rel},z}| < 0.5 \cdot lz \end{aligned} \tag{3}$$

For spherical region sensors equation 2 can be omitted. Checking is done as shown in equation 4 where $r$ is the radius of the sphere.

$$\begin{aligned} inside_{sphere} = &|\mathbf{x}_{PS_{abs},x}| < r \wedge \\ &|\mathbf{x}_{PS_{abs},y}| < r \wedge \\ &|\mathbf{x}_{PS_{abs},z}| < r \end{aligned} \tag{4}$$

Just mention that there's a second sort of sensors called *FlowSensors*. They are used for measuring the number and mass of particles passing two dimensional surfaces. Computation algorithm for these kind of sensors is basically the same like for contact detection und shouldn't be explained here.

# 6 Use cases

## 6.1 Bucket Excavator

As first use case a bucket excavator digging a hole should be simulated. The excavator itself was modeled as multibody system, which can easily be extended by hydraulic or electric components. For all parts which should interact with the granular material – in this case just the bucket – the new library component *CADPart* was used. As next step as pit of size 6.0 x 2.0 x 1.0 meters was generated by using the *PitGenerator* element. The new library has a direct interface to a database containing predefined materials, as described in section 4.2. So, the material chosen for the pit was *gravel*. Figure 6 shows the 3D view of a running simulation. As you can see particles are visualized directly in SimulationX®.

## 6.2 Loaded Truck

The in figure 7 shown model of the truck allows to determine the hydraulic forces in the main cylinder of the truck during the loading and unloading of bulk material by taking the elastic suspension into account. It is also possible to determine the influence of the moving bulk material of the drivability during different maneuvers and demonstrates additional features and the capabilities of the developed library. In the background is a *ParticleSource*,

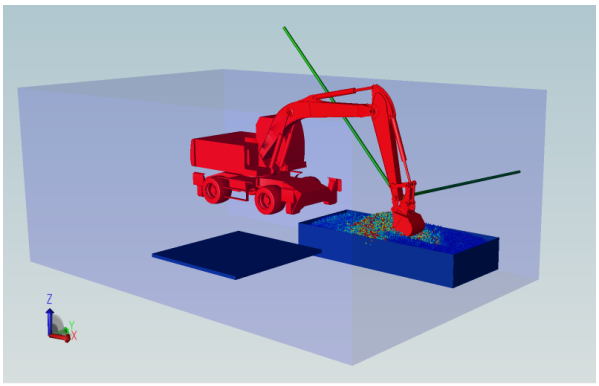

**Figure 6.** Excavator simulation

which the truck was being loaded with during the simulation. For the *ParticleSource* it is also possible to chose a predefined material of the database. Additionally, the feature is demonstrated that accelerated, rotating and automatically increasing simulation rooms are supported during co-simulation. With the *RegionSensors* the number of particles and the mass of the load can be evaluated which interacts with the *CADParts*. The ground is a *Plane* for the DEM simulation without any feedback to the system simulation.

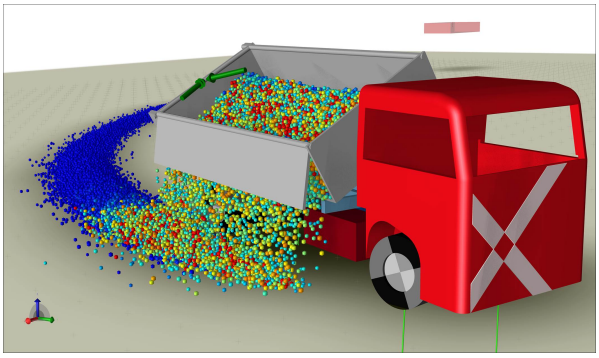

**Figure 7.** Loaded truck simulation

# 7 Conclusion and Outlook

In this work, a new concept was presented allowing the closed modelling of machine models and discrete element systems in one simulation tool. For that the command-oriented modeling technique many DEM applications work with was transferred into an object-oriented design approach. This approach allows to perform DEM simulations for inexperienced users who are not familiar with the DEM. But even for very experienced users, the new library will make it much easier to build up DEM models, run coupled simulations and analyze and document the results.

By supporting additional LIGGGHTS® features and additional wizards the modeling could be simplified, the possibilities expanded and the usebility of DEM models improved.

Additional LIGGGHTS® features could be the breaking and bonding of material. This field of DEM simulation and the determination of the appropriate material parameters is currently the content of some research-projects. But the results of that projects are far away to be used in coupled simulation. Additional wizards could be developed for example to allow the user to vary material parameters, create their own materials, or generate a multisphere body. Furthermore, the analysis possibilities of the LIGGGHTS® results in SimulationX could be expanded further in order to increase the added value of the coupled simulation. For this and for all other enhancements, we are looking forward to the feedback of future users and interested parties.

## Acknowledgements

This work is part of the project DEM-4-X funded by the BMWi (Federal Ministry for Economic Affairs and Energy, Project No.: 2055606KM4). The authors are deeply grateful for the financial support.

## References

Torsten Blochwitz, Martin Otter, Johan Åkesson, Martin Arnold, Christoph Clauß, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauß, Dietmar Neumerkel, Hans Olsson, and Antoine Viel. Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. In Martin Otter and Dirk Zimmer, editors, *Proceedings of the* 9th *International Modelica Conference*, Munich, Germany, September 2012. doi:10.3384/ecp12076173.

Peter Coad and Edward Yourdon. Object oriented analysis. 1991.

Peter A. Cundall. A computer model for simulating progressive, large-scale movements in blocky rock systems. In *Proc. Symp. Int. Rock Mech.*, volume 2, Nancy, 1971.

Hilding Elmqvist, Axel Goteman, Vilhelm Roxling, and Toheed Ghandriz. Generic Modelica Framework for MultiBody Contacts and Discrete Element Method. In Peter Fritzson and Hilding Elmqvist, editors, *Proceedings of the* 11th *International Modelica Conference*, Versailles, France, September 2015. doi:10.3384/ecp15118427.

Marcus Geimer, Thomas Krüger, and Peter Linsel. Co-Simulation, gekoppelte Simulation oder Simulationskopplung? Ein Versuch der Begriffsvereinheitlichung. *O+P Zeitschrift für Fluidtechnik - Aktorik, Steuerelektronik und Sensorik*, 50(11-12):572–576, 2006.

Christoph Kloss and Christoph Goniva. *Open Source Discrete Element Simulations of Granular Materials Based on Lammps*, volume 2, pages 781–788. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2011. ISBN 9781118062142. doi:10.1002/9781118062142.ch94.

Günther Kunze, Andre Katterfeld, and Tina Grüning. Simulation maschineller Erdbauprozesse. In *15. Fachtagung Schüttgutfördertechnik*, Munich, Germany, October 2010.

Günther Kunze, Andre Katterfeld, Christian Richter, Hendrik Otto, and Christian Schubert. Plattform- und Sofwareunabhängige Simulation der Erdstoff-Maschine Interaktion. In *5. Fachtagung Baumaschinentechnik*, Dresden, Germany, September 2012.