Study of Rigid Body Models in the PySPH Framework

Mid-Semester Review

Anirudh Jonnalagadda [1]

Internal Guide: Dr. Sukratu Barve [1]
External Guide: Dr. Prabhu Ramachandran [2]

[1]Center for Modeling and Simulation
Savitribai Phule Pune University

[2]Department of Aerospace Engineering
Indian Institute of Technology, Bombay

March 14, 2016

Picking up from Pre-Project

- Work done during Pre-Project:
  - Establish proficiency Python programming
  - Identify Potential Topics
  - Obtain (beginner's) understanding of the pySPH framework

- Work done during Pre-Project:
  - Establish proficiency Python programming
  - Identify Potential Topics
  - Obtain (beginner's) understanding of the pySPH framework
- Objectives and Tentative Roadmap

Picking up from Pre-Project

- Work done during Pre-Project:
  - Establish proficiency Python programming
  - Identify Potential Topics
  - Obtain (beginner's) understanding of the pySPH framework
- Objectives and Tentative Roadmap
  - Understand the Smoothed Particle Hydrodynamics (SPH) methodology
  - Understand implementation of the SPH method in pySPH

Picking up from Pre-Project

- Work done during Pre-Project:
  - Establish proficiency Python programming
  - Identify Potential Topics
  - Obtain (beginner's) understanding of the pySPH framework
- Objectives and Tentative Roadmap
  - Understand the Smoothed Particle Hydrodynamics (SPH) methodology
  - Understand implementation of the SPH method in pySPH
  - Understand the Discrete Element Method for modelling Rigid Body Interactions
    - Validate current Proof of Concept Rigid Body Collision Model

Picking up from Pre-Project

- Work done during Pre-Project:
  - Establish proficiency Python programming
  - Identify Potential Topics
  - Obtain (beginner's) understanding of the pySPH framework
- Objectives and Tentative Roadmap
  - Understand the Smoothed Particle Hydrodynamics (SPH) methodology
  - Understand implementation of the SPH method in pySPH
  - Understand the Discrete Element Method for modelling Rigid Body Interactions
    - Validate current Proof of Concept Rigid Body Collision Model
  - Understand mechanism of Rigid Body - Fluid Coupling (currently done using WCSPH)
    - Validate current Proof of Concept Rigid Body-Fluid Coupled Model

Picking up from Pre-Project

- Work done during Pre-Project:
  - Establish proficiency Python programming
  - Identify Potential Topics
  - Obtain (beginner's) understanding of the pySPH framework
- Objectives and Tentative Roadmap
  - Understand the Smoothed Particle Hydrodynamics (SPH) methodology
  - Understand implementation of the SPH method in pySPH
  - Understand the Discrete Element Method for modelling Rigid Body Interactions
    - Validate current Proof of Concept Rigid Body Collision Model
  - Understand mechanism of Rigid Body - Fluid Coupling (currently done using WCSPH)
    - Validate current Proof of Concept Rigid Body-Fluid Coupled Model
  - Implement physically consistent models

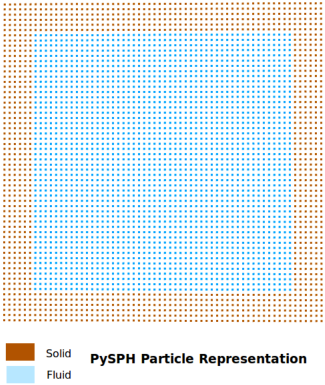
Picking up from Pre-Project

- Work done during Pre-Project:
  - Establish proficiency Python programming
  - Identify Potential Topics
  - Obtain (beginner's) understanding of the pySPH framework
- Objectives and Tentative Roadmap
  - Understand the Smoothed Particle Hydrodynamics (SPH) methodology
  - Understand implementation of the SPH method in pySPH
  - Understand the Discrete Element Method for modelling Rigid Body Interactions
    - Validate current Proof of Concept Rigid Body Collision Model
  - Understand mechanism of Rigid Body - Fluid Coupling (currently done using WCSPH)
    - Validate current Proof of Concept Rigid Body-Fluid Coupled Model
  - Implement physically consistent models
  - Implement Solid-Fluid coupling using IISPH technique

- SPH is a numerical method developed for problems pertaining to Astrophysics.

- SPH is a numerical method developed for problems pertaining to Astrophysics.
- It is a Meshfree Particle Method where the domain discretization is performed as:
  - Spatial Discretization with Particle Representation : Domain representation with particles
  - Numerical Discretization with Particle Approximation : Operators and functions of the governing equations are approximated using the particles.

- SPH is a numerical method developed for problems pertaining to Astrophysics.
- It is a Meshfree Particle Method where the domain discretization is performed as:
  - Spatial Discretization with Particle Representation : Domain representation with particles
  - Numerical Discretization with Particle Approximation : Operators and functions of the governing equations are approximated using the particles.
- The SPH methodology uses the Lagrangian Formulation of the governing equations (i.e. all the governing equations are expressed in terms of the Substantial Derivative)



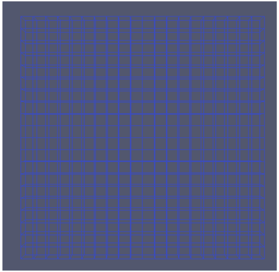
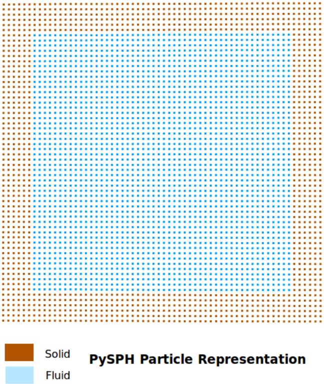


Figure : Discretization in Meshfree and Grid Based Methods

The SPH Method: Spatial Discretization



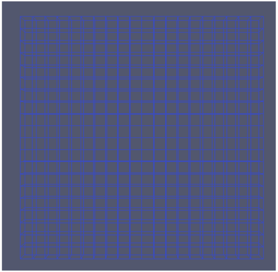


Figure : Discretization in Meshfree and Grid Based Methods

- Particle Distribution can be completely arbitrary
- No explicit interconnectivity between particles

- Consider a Scalar Field $A(\mathbf{r}, t)$,

$$A(\mathbf{r}, t) = (A * \delta)(\mathbf{r}, t)$$

- Consider a Scalar Field A($\mathbf{r}$,t),

$$A(\mathbf{r}, t) = (A * \delta)(\mathbf{r}, t) = \int_{\Omega} A(\mathbf{r'}, t)\delta(\mathbf{r} - \mathbf{r'})d^3\mathbf{r'}$$

The SPH Method: Numerical Discretization

- Consider a Scalar Field $A(\mathbf{r}, t)$,

$$A(\mathbf{r}, t) = (A * \delta)(\mathbf{r}, t) = \int_{\Omega} A(\mathbf{r'}, t)\delta(\mathbf{r} - \mathbf{r'})d^3\mathbf{r'}$$

- This δ is approached through an Interpolation Kernel $w_h(\mathbf{r} - \mathbf{r'})$.

The SPH Method: Numerical Discretization

- Consider a Scalar Field A($\mathbf{r}$,t),

$$A(\mathbf{r}, t) = (A * \delta)(\mathbf{r}, t) = \int_{\Omega} A(\mathbf{r'}, t)\delta(\mathbf{r} - \mathbf{r'})d^3\mathbf{r'}$$

- This δ is approached through an Interpolation Kernel $w_h(\mathbf{r} - \mathbf{r'})$.
- Kernel/Continuous Approximation of A is,

$$A(\mathbf{r}, t) \approx [A]_c(\mathbf{r}, t) = \int_{\Omega} A(\mathbf{r'}, t)w_h(\mathbf{r} - \mathbf{r'})d^3\mathbf{r'}$$

- Consider a Scalar Field A($\mathbf{r}$,t),

$$A(\mathbf{r}, t) = (A * \delta)(\mathbf{r}, t) = \int_{\Omega} A(\mathbf{r'}, t)\delta(\mathbf{r} - \mathbf{r'})d^3\mathbf{r'}$$

- This δ is approached through an Interpolation Kernel $w_h(\mathbf{r} - \mathbf{r'})$.
- Kernel/Continuous Approximation of A is,

$$A(\mathbf{r}, t) \approx [A]_c(\mathbf{r}, t) = \int_{\Omega} A(\mathbf{r'}, t)w_h(\mathbf{r} - \mathbf{r'})d^3\mathbf{r'}$$

- The Kernel is a Real function and has non-zero values on a compact/infinite sized support
- The support depends on the **Smoothing Length**, h.

- Kernel Properties (for $[A]_c$ to be First Order Accurate)
  - It's integral over the domain should be 1
  - It's first moment should be 0;

The SPH Method: Numerical Discretization

- Kernel Properties (for $[A]_c$ to be First Order Accurate)
  - It's integral over the domain should be 1
  - It's first moment should be 0; satisfied by taking a symmetrical Kernel

The SPH Method: Numerical Discretization

- Kernel Properties (for $[A]_c$ to be First Order Accurate)
  - It's integral over the domain should be 1
  - It's first moment should be 0; satisfied by taking a symmetrical Kernel
- Particle Approximation of A: $[A]_c$ approximated by a Riemann Sum

$$[A]_c(\mathbf{r}, t) = \int_{\Omega} A(\mathbf{r'}, t) w_h(\mathbf{r} - \mathbf{r'}) d^3\mathbf{r'}$$

$$\approx \sum_b A(\mathbf{r_b}, t)\ w_h(|\mathbf{r} - \mathbf{r_b}|)\ V_b$$

- Kernel Properties (for $[A]_c$ to be First Order Accurate)
  - It's integral over the domain should be 1
  - It's first moment should be 0; satisfied by taking a symmetrical Kernel
- Particle Approximation of A: $[A]_c$ approximated by a Riemann Sum

$$[A]_c(\mathbf{r}, t) = \int_\Omega A(\mathbf{r'}, t) w_h(\mathbf{r} - \mathbf{r'}) d^3\mathbf{r'}$$

$$\approx \sum_b A(\mathbf{r_b}, t)\ w_h(|\mathbf{r} - \mathbf{r_b}|)\ V_b = [A]_d$$

The SPH Method: Numerical Discretization

- Kernel Properties (for $[A]_c$ to be First Order Accurate)
  - It's integral over the domain should be 1
  - It's first moment should be 0; satisfied by taking a symmetrical Kernel
- Particle Approximation of A: $[A]_c$ approximated by a Riemann Sum

$$[A]_c(\mathbf{r}, t) = \int_\Omega A(\mathbf{r'}, t) w_h(\mathbf{r} \text{ - } \mathbf{r'}) d^3\mathbf{r'}$$

$$\approx \sum_b A(\mathbf{r_b}, t)\ w_h(|\mathbf{r} - \mathbf{r_b}|)\ V_b = [A]_d$$

$$\therefore [A]_d = \sum_b A(\mathbf{r_b}, t)\ w_h(|\mathbf{r_{ab}}|)\ V_b,$$

$$\forall a \in \Omega, \mathbf{r}_{ab} = \mathbf{r}_a - \mathbf{r}_b$$

- The properties of each particle **'a'** depends on the **'b'** particles in its neighbourhood defined by the Smoothing Length **'h'**

- The properties of each particle **'a'** depends on the **'b'** particles in its neighbourhood defined by the Smoothing Length **'h'**
- A Nearest Neighbour Particle Search (NNPS) algorithm takes care of which particles **'b'** influence the properties of **'a'**

The SPH Method

- The properties of each particle **'a'** depends on the **'b'** particles in its neighbourhood defined by the Smoothing Length **'h'**
- A Nearest Neighbour Particle Search (NNPS) algorithm takes care of which particles **'b'** influence the properties of **'a'**
  - pySPH implements a Linked List NNPS algorithm by default

- The properties of each particle **'a'** depends on the **'b'** particles in its neighbourhood defined by the Smoothing Length **'h'**
- A Nearest Neighbour Particle Search (NNPS) algorithm takes care of which particles **'b'** influence the properties of **'a'**
  - pySPH implements a Linked List NNPS algorithm by default
- The Finite Element Method and SPH
  - Both Methods represent the field variables as linear combinations of basis functions

- The properties of each particle **'a'** depends on the **'b'** particles in its neighbourhood defined by the Smoothing Length **'h'**
- A Nearest Neighbour Particle Search (NNPS) algorithm takes care of which particles **'b'** influence the properties of **'a'**
  - pySPH implements a Linked List NNPS algorithm by default
- The Finite Element Method and SPH
  - Both Methods represent the field variables as linear combinations of basis functions
  - FEM constructs these basis functions from the grid and SPH constructs them based on the kernel

The SPH Method

- The properties of each particle **'a'** depends on the **'b'** particles in its neighbourhood defined by the Smoothing Length **'h'**
- A Nearest Neighbour Particle Search (NNPS) algorithm takes care of which particles **'b'** influence the properties of **'a'**
  - pySPH implements a Linked List NNPS algorithm by default
- The Finite Element Method and SPH
  - Both Methods represent the field variables as linear combinations of basis functions
  - FEM constructs these basis functions from the grid and SPH constructs them based on the kernel
  - The FEM basis functions are stationary, while those of SPH move along with the particles.
    i.e. if **'B'** are the SPH Basis functions then $\frac{D\mathbf{B}}{Dt} = 0$

The Discrete Element Method

- It is a numerical scheme which allows finite rotations and displacements of discrete bodies

The Discrete Element Method

- It is a numerical scheme which allows finite rotations and displacements of discrete bodies
- The bodies interact with their nearest neighbours through Contact Laws; the interactions create new or destroy earlier contacts.

The Discrete Element Method

- It is a numerical scheme which allows finite rotations and displacements of discrete bodies
- The bodies interact with their nearest neighbours through Contact Laws; the interactions create new or destroy earlier contacts.
- **Soft Contact** Method
  - The bodies are allowed to "overlap"
  - The amount and rate of the overlap gives incremental contact forces
  - These contact forces are applied to Newton's laws to obtain new out of balance forces and moments
  - From the forces and moments, the velocities and displacements are evaluated

- DEM terminology:
  - Contact Search: The procedure of keeping track of the bodies that are in contact with a given body in a time-step
  - Boxing: Time saving mechanism wherein the working area is divided into squares/cubes (for 2D or 3D)

- DEM terminology:
  - Contact Search: The procedure of keeping track of the bodies that are in contact with a given body in a time-step
  - Boxing: Time saving mechanism wherein the working area is divided into squares/cubes (for 2D or 3D)
- Calculation of Contact Force
  - Ascertain if contact is present
  - Calculate the amount of overlap
  - Use the overlap in the contact model and compute the contact force

- DEM terminology:
  - Contact Search: The procedure of keeping track of the bodies that are in contact with a given body in a time-step
  - Boxing: Time saving mechanism wherein the working area is divided into squares/cubes (for 2D or 3D)
- Calculation of Contact Force
  - Ascertain if contact is present
  - Calculate the amount of overlap
  - Use the overlap in the contact model and compute the contact force
- Since the bodies are represented as collections of particles in pySPH, explicit Boxing and Contact Search is not needed
- the NNPS keeps track of all those elements which are in contact and the smoothing length effectively computes the amount of overlap

- Models the contacts as a Spring and Dashpot system

- Models the contacts as a Spring and Dashpot system
- Current PoC model implementation

  Repulsive Force, $f_{i,s} = -k(d - |\mathbf{r_{ij}}|)\frac{\mathbf{r_{ij}}}{|\mathbf{r_{ij}}|}$

  Damping Force, $f_{i,d} = \eta\ \mathbf{v_{ij}}$

  Shear Force, $f_{i,t} = k_t\ \mathbf{v}_{\mathbf{ij},t}$

  where,

  k = Spring Coefficient,

  k_t = Coefficient of Shear,

  η = Damping Coefficient,

  d = diameter of elements comprising the body

  $\mathbf{r_{ij}} = \mathbf{r}_i - \mathbf{r}_j$,

  $\mathbf{v_{ij}} = \mathbf{v}_i - \mathbf{v}_j$

  Relative Tangential Velocity,

$$\mathbf{v}_{\mathbf{ij},t} = \mathbf{v_{ij}} - \left(\mathbf{v_{ij}} \cdot \frac{\mathbf{r_{ij}}}{|\mathbf{r_{ij}}|}\right)\frac{\mathbf{r_{ij}}}{|\mathbf{r_{ij}}|}$$

- Total Contact Force,

$$F_c = \sum_{i \in RigidBody} (f_{i,s} + f_{i,d} + f_{i,t})$$

- Total Torque due to contact,

$$T_C = \sum_{i \in RigidBody} [r_i \times (f_{i,s} + f_{i,d} + f_{i,t})]$$

Newton's Equations for Rigid Body Dynamics

Rigid Body "I" $= \{particles : r_{ij} = 0, \; \forall t, i, j \in I\}$

Newton's Equations for Rigid Body Dynamics

Rigid Body "I" $= \{particles : r_{ij} = 0,\ \forall t, i, j \in I\}$

$$M_I \frac{d\mathbf{V_I}}{dt} = \sum_{k \in I} m_k \frac{d\mathbf{v_k}}{dt}$$

$$I_I \frac{d\Omega_\mathbf{I}}{dt} = \sum_{k \in I} m_k \left(\mathbf{r_k} - \mathbf{R_I}\right) \times \frac{d\mathbf{v_k}}{dt}$$

where,

$M_I, \mathbf{V_I}, \mathbf{R_I} =$ Mass, Velocity and Center of Gravity of "I",
$I_I, \Omega_\mathbf{I} =$Inertial Tensor(MoI), Angular Velocity of "I"
$m_k, \mathbf{v_k}, \mathbf{r_k} =$ Mass, Velocity and Position of k^{th} particle

- Test Cases: Zhang et.al[1] performed experimental studies to validate a coupled Finite Volume Particle (FVP) DEM to simulate interactions between fluids and solid bodies

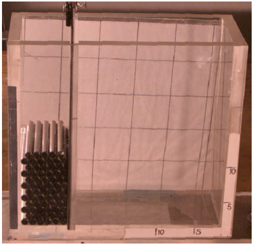


Figure : Experimental Setup

- Tank Dimensions: l = h = 26cm , w = 10cm
- Cylinder Properties: l = 9.9cm ,d = 1cm, $\rho = 2.7 \times 10^3 \frac{kg}{m^3}$

[1] Zhang et.al, Simulation of solid-fluid mixture flow using moving particle methods, J. Comput.Phys. 228 (2009) 25522565, http://dx.doi.org/10.1016/j.jcp.2008.12.005

Validation of RigidBodyCollision model: Expected Profiles

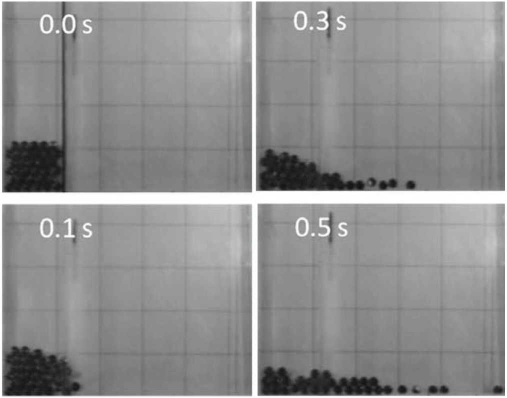


Figure : Evolution of the system at various times (6 Cylinder stack)

Validation of RigidBodyCollision model: Expected Profiles

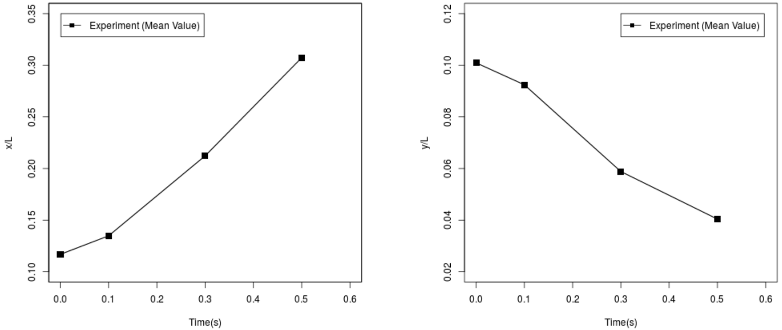


Figure : 6 Cylinder System Center of Mass

Validation of RigidBodyCollision model

- Tweaking Standard Examples

Validation of RigidBodyCollision model

- Tweaking Standard Examples
- Diving-in example: Simple Pendulum

Validation of RigidBodyCollision model

- Tweaking Standard Examples
- Diving-in example: Simple Pendulum
- Case Setup
  - Setup is done in pure Python, using Numpy, and PySPH libraries
  - Particles are created using the "collapsing_cylinder_2d.py" program
  - Simulation is run using the "rigid.py" program

Validation of RigidBodyCollision model

- Tweaking Standard Examples
- Diving-in example: Simple Pendulum
- Case Setup
  - Setup is done in pure Python, using Numpy, and PySPH libraries
  - Particles are created using the "collapsing_cylinder_2d.py" program
  - Simulation is run using the "rigid.py" program
- Current Model results (Best outcome)
  - k=8, d=2.5, η =0.01, k_t=0.1

Validation of RigidBodyCollision model

- Tweaking Standard Examples
- Diving-in example: Simple Pendulum
- Case Setup
  - Setup is done in pure Python, using Numpy, and PySPH libraries
  - Particles are created using the “collapsing_cylinder_2d.py” program
  - Simulation is run using the “rigid.py” program
- Current Model results (Best outcome)
  - k=8, d=2.5, η =0.01, k_t=0.1
- Conclusion: The current PoC model is not physically consistent

Making RogidBodyCollision model Physically Consistent

- Current Model implements a simple Linear Spring and Dashpot Collision Model

Making RogidBodyCollision model Physically Consistent

- Current Model implements a simple Linear Spring and Dashpot Collision Model
- Canelas,et.al.[2]implemented a Modified, Non-Linear Hertzian model in their recently published work (Feb 2016)

[2]R.B.Canelas, et.al., SPH-DCDEM model for Arbitrary geometries in free surface solid-fluid flows, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.01.006

PoA for Remaining Semester

- Implement the Non-Linear Hertzian model in pySPH
- Validate the Model

PoA for Remaining Semester

- Implement the Non-Linear Hertzian model in pySPH
- Validate the Model
- Perform Validation Study for the Solid-Fluid Coupled Model

PoA for Remaining Semester

- Implement the Non-Linear Hertzian model in pySPH
- Validate the Model
- Perform Validation Study for the Solid-Fluid Coupled Model
- Perform Validation Study for the Solid-Fluid Coupled Model with Fluid equations solved with the IISPH formulation

Thank you!

Questions?