NetDEM

Generated by Doxygen 1.9.2

Chapter 1

Welcome

NetDEM is a C++ program library targeted at the computational mechanics of irregular granular materials and utilizing machine learning tools to boost the computational effciency.

It is currently capable of performing basic and general DEM simulations, with following features:

- Sphere and triangle facets contact solver
- GJK contact solver for convex particles
- SDF contact solver for arbitrary (convex and concave) particles
- Hybrid OpenMP and MPI parallel computing
- Integrated `mlpack` machine learning environment

The supported particle shapes include `sphere`, `cylinder`, `poly-super-ellipsoid`, `poly-super-quadrics`, `spherical harmonics`, `triangle mesh`, `level set`, etc.

1.1 Get started

1.1.1 Install

1.1.1.1 Prerequisites:

The compilation requires `gcc`, `autoconf`, `automake`, `cmake`, `mpi`, `boost`, which can be obtained using

```
# For MacOS: use brew install, such as
brew install gcc autoconf automake cmake openmpi boost

# For Ubuntu: use apt-get install, such as
sudo apt install build-essential
sudo apt-get install -y autoconf-archive automake cmake texinfo
sudo apt-get install openmpi-bin libopenmpi-dev libboost-all-dev
```

1.1.1.2 Compile and build:

```
make sync_submodule
make
```

If some third-party libraries have not been or cannot be downloaded successfully, you can delete them and do a `git checkout contrib` and `make sync_submodule` again.

1.1.1.3 Test the installation:

```
./scripts/run_tests.sh
```

1.1.2 Examples & tutorials

Some preliminary examples are located under directory `examples/`, which can be run with, e.g.,
`./build/bin/netdem_example_random_packing 1`

For more details please refer to the website.

1.1.3 Visualize & post-process

Results can be dumped as VTK files, which can be visualized in paraview.

We have a side-repository that provides some matlab or python scripts for post-process (e.g., VTK io, stress-strain, spherical histograms of contact anisotropy).

1.2 Support

This code is under active development. Please join us if you have an interest to contribute.

If you need help using NetDEM, or have found a bug, please open an issue or submit a pull request.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 netdem Namespace Reference

Namespaces

- namespace Math

Classes

- class BodyForce
- class BondedSpheres
- class BondedVoronois
- class BondEntry
- struct BondEntryData
- class BondEntryParser
- class BondGeometries
- class BondSolverPP
- class BondSolverPW
- class BreakageAnalysisPD
- class Cell
- class CellManager
- class CollisionEntry
- struct CollisionEntryData
- class CollisionEntryParser
- class CollisionGeometries
- class CollisionSolverPP
- class CollisionSolverPW
- class Command
- class CommandCreate
- class ContactForces
- class ContactModel
- class ContactModelFactory
- class ContactPP
- struct ContactPPData
- class ContactPPParser
- class ContactPW
- struct ContactPWData

- class ContactPWParser
- class ContactSolverFactory
- class ContactSolverSettings
- class Cork
- class Cylinder
- class DataDumper
- class DeformableParticle
- class DeformationAnalysis
- class DEMFragment
- class DEMObjectPool

  particles and contacts are frequently added to or removed from the scene. The pool strategy is used to avoid the frequently construction and de-construction of object instances. When a particle or wall needs to be added, an instances will be obtained from the pool. When a particle or wall needs to be removed,it is recycled and stored in the pool. to do: object pool need to be improved
- class DEMProfiler
- class DEMSolver
- class Distribution
- class Domain
- class DomainManager
- class DomainSplittor
- class Ellipsoid
- class FEMSimulator
- class GeneralNet
- class Gravity
- class HertzMindlin
- class InputProcessor
- class LevelSet
- class LevelSetFunction
- class LevelSetSplittor
- class LinearSpring
- class Membrane
- class MembraneWall
- class MiniMap
- class Modifier
- class ModifierManager
- class MPIDataDefine
- class MPIManager
- struct my_pair
- class NeighPofP
- class NeighPofW
- class NeighWofP
- class PackGenerator
- struct pair_hash
- class ParallelBond
- class Particle
- struct ParticleData
- struct ParticleEnergy
- class ParticleEnergyCalculator
- class ParticleParser
- class ParticleStrengthParameters
- class PeriDigmBlock
- class PeriDigmBoundaryCondition
- class PeriDigmDamageModel
- class PeriDigmDEMCoupler
- class PeriDigmDiscretization

- class PeriDigmMaterial
- class PeriDigmSettings
- class PeriDigmSimulator
- class Plane
- class Polybezier
- class PolySuperEllipsoid
- class PolySuperQuadrics
- class RegressionNet
- class Scene
- class SDFCalculator
- class Shape
- class ShapeFactory
- class Simplex
- class Simulation
- class SolverANNPP
- class SolverANNPPlane
- class SolverANNPW
- class SolverBooleanPP
- class SolverBooleanPW
- class SolverGJKPP
- class SolverGJKPW
- class SolverSDFPP
- class SolverSDFPW
- class SolverSpherePlane
- class SolverSphereSphere
- class SolverSphereTriangle
- class Sphere
- class SphericalHarmonics
- class SphericalVoronoi
- class STLModel
- class STLReader
- class TetMesh
- class TetMeshSplittor
- class Triangle
- class TriMesh
- class VolumeBased
- class Voronoi
- class Wall
- class WallBoxPlane
- class WallBoxPlate
- class WallDispControl
- class WallServoControl
- class WSCVTSampler

Typedefs

- typedef long long int64t
- using size_t = std::size_t
- using Vec2i = std::array< int, 2 >
- using Vec3i = std::array< int, 3 >
- using Vec4i = std::array< int, 4 >
- using Vec2d = std::array< double, 2 >
- using Vec3d = std::array< double, 3 >
- using Vec4d = std::array< double, 4 >

- using Mat2d = std::array< std::array< double, 2 >, 2 >
- using Mat3d = std::array< std::array< double, 3 >, 3 >
- template<size_t N>
  using VecNi = std::array< int, N >
- template<size_t N>
  using VecNd = std::array< double, N >
- template<size_t Nr, size_t Nc>
  using MatNd = std::array< std::array< double, Nc >, Nr >
- template<typename T >
  using VecXT = std::vector< T >
- template<typename T , size_t N>
  using VecNT = std::array< T, N >

Enumerations

- enum TimerType {
  linked_list , contacts , particles , walls ,
  pre_modifiers , mid_modifiers , post_modifiers , mpi_communication ,
  custom , num_timers }
- enum class LayerName {
  IdentityLayer , LayerNorm , Linear , ReLU ,
  LeakyReLU , FlexibleReLU , ELU , Softmax ,
  LogSoftMax , LSTM }

Functions

- double GetMSE (const arma::mat &pred, const arma::mat &Y)
- double GetMAE (const arma::mat &pred, const arma::mat &Y)
- arma::mat GetLabels (const arma::mat &ann_outputs)
- void cgal_tetmesh (const VecXT< Vec3d > &vv, const VecXT< Vec3i > &ff, VecXT< Vec3d > *const tv, VecXT< Vec4i > *const tt, double mesh_size)
- void cgal_smooth_mesh (VecXT< Vec3d > *const vv, VecXT< Vec3i > *const ff, int num_iters)
- void cgal_alpha_shape (VecXT< Vec3d > *vv_out, VecXT< Vec3i > *ff_out, const VecXT< Vec3d > &vv↩
  _in, double alpha=0.7)
- void STDToEigen (const VecXT< VecXT< double > > &std_mat, Eigen::MatrixXd *eigen_mat)
- void STDToEigen (const Mat3d &std_mat, Eigen::Matrix3d *eigen_mat)
- void STDToEigen (const VecXT< double > &std_vec, Eigen::VectorXd *eigen_vec)
- void STDToEigen (const Vec3d &std_vec, Eigen::Vector3d *eigen_vec)
- void EigenToSTD (VecXT< VecXT< double > > *const std_mat, const Eigen::MatrixXd &eigen_mat)
- void EigenToSTD (Mat3d *const std_mat, const Eigen::Matrix3d &eigen_mat)
- void EigenToSTD (VecXT< double > *const std_vec, const Eigen::VectorXd &eigen_vec)
- void EigenToSTD (Vec3d *const std_vec, const Eigen::Vector3d &eigen_vec)
- Mat3d EigenVector (const Mat3d &mat)
- VecXT< double > EigenSolve (const VecXT< VecXT< double > > &a, const VecXT< double > &b)
- Vec3d EigenSolve (Mat3d const &a, const Vec3d &b)
- void STDToEigen (const VecXT< Vec3d > &std_mat, Eigen::MatrixXd *eigen_mat)
- void STDToEigen (const VecXT< Vec3i > &std_mat, Eigen::MatrixXi *eigen_mat)
- void STDToEigen (const VecXT< Vec4i > &std_mat, Eigen::MatrixXi *eigen_mat)
- void EigenToSTD (VecXT< Vec3d > *const std_mat, const Eigen::MatrixXd &eigen_mat)
- void EigenToSTD (VecXT< Vec3i > *const std_mat, const Eigen::MatrixXi &eigen_mat)
- void EigenToSTD (VecXT< Vec4i > *const std_mat, const Eigen::MatrixXi &eigen_mat)
- void EigenToSTD (VecXT< int > *const std_vec, const Eigen::VectorXi &eigen_vec)
- void igl_remove_unreferenced_vertices (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f)

- void igl_remove_duplicate_vertices (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f)
- void igl_remove_duplicate_vertices (VecXT< Vec3d > *const v)
- void igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- void igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- void igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, double dist_pc_to_plane, Vec3d const &dir_n, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- void igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, double dist_pc_to_plane, Vec3d const &dir_n, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- void igl_mesh_refine (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f, int num_refines)
- void igl_mesh_decimate (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f, int num_facets)
- int igl_facet_components (const VecXT< Vec3i > &fi, VecXT< int > *const fc)
- void igl_reorient_facets (const VecXT< Vec3d > &v, VecXT< Vec3i > *f)
- bool igl_check_winding (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f)
- void igl_convex_hull (const VecXT< Vec3d > &v0, VecXT< Vec3d > *const v1, VecXT< Vec3i > *const f1)
- void igl_tetmesh_boundary (const VecXT< Vec4i > &tt, VecXT< Vec3i > *const ff, VecXT< int > *const fj)
- void igl_tetmesh_boundary (const VecXT< Vec4i > &tt, VecXT< Vec3i > *const ff)
- VecXT< int > igl_points_inside_mesh (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f, const VecXT< Vec3d > &v_querry)
- void igl_marching_cubes (VecXT< Vec3d > *const vv, VecXT< Vec3i > *const ff, VecXT< VecXT< VecXT< double > > > const &sdf, Vec3d const &corner, Vec3d const &spacing, double iso_value)
- void PrintWarning (std::string const &info)
- void PrintError (std::string const &info)
- void PrintDebug (std::string const &info)
- std::string my_to_string (int value)
- std::string my_to_string (double value)
- VecXT< VecXT< double > > ImportDataTxtToVec (std::string const &filename, int lines_to_skip=0)
- bool FileExist (std::string const &filename)
- std::ostream & operator<< (std::ostream &os, Vec3i const &obj)
- std::ostream & operator<< (std::ostream &os, Vec3d const &obj)
- std::ostream & operator<< (std::ostream &os, Vec4d const &obj)
- std::ostream & operator<< (std::ostream &os, Mat3d const &obj)
- Vec3d operator+ (Vec3d const &lhs, double rhs)
- Vec3d operator+ (double lhs, Vec3d const &rhs)
- Vec3i operator+ (Vec3i const &lhs, int rhs)
- Vec3i operator+ (int lhs, Vec3i const &rhs)
- Vec3d operator- (Vec3d const &lhs, double rhs)
- Vec3d operator- (double lhs, Vec3d const &rhs)
- Vec3d operator* (Vec3d const &lhs, double rhs)
- Vec3d operator* (double lhs, Vec3d const &rhs)
- Vec3d operator/ (Vec3d const &lhs, double rhs)
- Vec3d operator/ (double lhs, Vec3d const &rhs)
- Vec3d operator+ (Vec3d const &lhs, Vec3d const &rhs)
- Vec3d operator- (Vec3d const &lhs, Vec3d const &rhs)
- Vec3d operator* (Vec3d const &lhs, Vec3d const &rhs)
- Vec3d operator/ (Vec3d const &lhs, Vec3d const &rhs)

6.1.1 Typedef Documentation

6.1.1.1 int64t

```
typedef long long netdem::int64t
```

6.1.1.2 Mat2d

```
using netdem::Mat2d = typedef std::array<std::array<double, 2>, 2>
```

6.1.1.3 Mat3d

```
using netdem::Mat3d = typedef std::array<std::array<double, 3>, 3>
```

6.1.1.4 MatNd

```
template<size_t Nr, size_t Nc>
using netdem::MatNd = typedef std::array<std::array<double, Nc>, Nr>
```

6.1.1.5 size_t

```
using netdem::size_t = typedef std::size_t
```

6.1.1.6 Vec2d

```
using netdem::Vec2d = typedef std::array<double, 2>
```

6.1.1.7 Vec2i

```
using netdem::Vec2i = typedef std::array<int, 2>
```

6.1.1.8 Vec3d

```
using netdem::Vec3d = typedef std::array<double, 3>
```

6.1.1.9 Vec3i

```
using netdem::Vec3i = typedef std::array<int, 3>
```

6.1.1.10 Vec4d

```
using netdem::Vec4d = typedef std::array<double, 4>
```

6.1.1.11 Vec4i

```
using netdem::Vec4i = typedef std::array<int, 4>
```

6.1.1.12 VecNd

```
template<size_t N>
using netdem::VecNd = typedef std::array<double, N>
```

6.1.1.13 VecNi

```
template<size_t N>
using netdem::VecNi = typedef std::array<int, N>
```

6.1.1.14 VecNT

```
template<typename T , size_t N>
using netdem::VecNT = typedef std::array<T, N>
```

6.1.1.15 VecXT

```
template<typename T >
using netdem::VecXT = typedef std::vector<T>
```

6.1.2 Enumeration Type Documentation

6.1.2.1 LayerName

```
enum class netdem::LayerName  [strong]
```

Enumerator

IdentityLayer	
LayerNorm	
Linear	
ReLU	
LeakyReLU	
FlexibleReLU	
ELU	
Softmax	
LogSoftMax	
LSTM	

6.1.2.2 TimerType

```
enum netdem::TimerType
```

Enumerator

linked_list	
contacts	
particles	
walls	
pre_modifiers	
mid_modifiers	
post_modifiers	
mpi_communication	
custom	
num_timers	

6.1.3 Function Documentation

6.1.3.1 cgal_alpha_shape()

```
void netdem::cgal_alpha_shape (
            VecXT< Vec3d > * vv_out,
            VecXT< Vec3i > * ff_out,
            const VecXT< Vec3d > & vv_in,
            double alpha = 0.7 )
```

6.1.3.2 cgal_smooth_mesh()

```
void netdem::cgal_smooth_mesh (
            VecXT< Vec3d > *const vv,
            VecXT< Vec3i > *const ff,
            int num_iters )
```

6.1.3.3 cgal_tetmesh()

```
void netdem::cgal_tetmesh (
            const VecXT< Vec3d > & vv,
            const VecXT< Vec3i > & ff,
            VecXT< Vec3d > *const tv,
            VecXT< Vec4i > *const tt,
            double mesh_size )
```

6.1.3.4 EigenSolve() [1/2]

```
VecXT< double > netdem::EigenSolve (
            const VecXT< VecXT< double > > & a,
            const VecXT< double > & b )
```

6.1.3.5 EigenSolve() [2/2]

```
Vec3d netdem::EigenSolve (
            Mat3d const & a,
            const Vec3d & b )
```

6.1.3.6 EigenToSTD() [1/8]

```
void netdem::EigenToSTD (
            Mat3d *const std_mat,
            const Eigen::Matrix3d & eigen_mat )  [inline]
```

6.1.3.7 EigenToSTD() [2/8]

```
void netdem::EigenToSTD (
            Vec3d *const std_vec,
            const Eigen::Vector3d & eigen_vec )  [inline]
```

6.1.3.8 EigenToSTD() [3/8]

```
void netdem::EigenToSTD (
            VecXT< double > *const std_vec,
            const Eigen::VectorXd & eigen_vec )  [inline]
```

6.1.3.9 EigenToSTD() [4/8]

```
void netdem::EigenToSTD (
            VecXT< int > *const std_vec,
            const Eigen::VectorXi & eigen_vec )  [inline]
```

6.1.3.10 EigenToSTD() [5/8]

```
void netdem::EigenToSTD (
            VecXT< Vec3d > *const std_mat,
            const Eigen::MatrixXd & eigen_mat )  [inline]
```

6.1.3.11 EigenToSTD() [6/8]

```
void netdem::EigenToSTD (
            VecXT< Vec3i > *const std_mat,
            const Eigen::MatrixXi & eigen_mat )  [inline]
```

6.1.3.12 EigenToSTD() [7/8]

```
void netdem::EigenToSTD (
            VecXT< Vec4i > *const std_mat,
            const Eigen::MatrixXi & eigen_mat )  [inline]
```

6.1.3.13 EigenToSTD() [8/8]

```
void netdem::EigenToSTD (
            VecXT< VecXT< double > > *const std_mat,
            const Eigen::MatrixXd & eigen_mat )  [inline]
```

6.1.3.14 EigenVector()

```
Mat3d netdem::EigenVector (
            const Mat3d & mat )  [inline]
```

6.1.3.15 FileExist()

```
bool netdem::FileExist (
            std::string const & filename )
```

6.1.3.16 GetLabels()

```
arma::mat netdem::GetLabels (
            const arma::mat & ann_outputs )
```

6.1.3.17 GetMAE()

```
double netdem::GetMAE (
            const arma::mat & pred,
            const arma::mat & Y )
```

6.1.3.18 GetMSE()

```
double netdem::GetMSE (
            const arma::mat & pred,
            const arma::mat & Y )
```

6.1.3.19 igl_check_winding()

```
bool netdem::igl_check_winding (
            const VecXT< Vec3d > & v,
            const VecXT< Vec3i > & f )
```

6.1.3.20 igl_convex_hull()

```
void netdem::igl_convex_hull (
            const VecXT< Vec3d > & v0,
            VecXT< Vec3d > *const v1,
            VecXT< Vec3i > *const f1 )
```

6.1.3.21 igl_facet_components()

```
int netdem::igl_facet_components (
            const VecXT< Vec3i > & fi,
            VecXT< int > *const fc )
```

6.1.3.22 igl_marching_cubes()

```
void netdem::igl_marching_cubes (
            VecXT< Vec3d > *const vv,
            VecXT< Vec3i > *const ff,
            VecXT< VecXT< VecXT< double > > > const & sdf,
            Vec3d const & corner,
            Vec3d const & spacing,
            double iso_value )
```

6.1.3.23 igl_mesh_decimate()

```
void netdem::igl_mesh_decimate (
            VecXT< Vec3d > *const v,
            VecXT< Vec3i > *const f,
            int num_facets )
```

6.1.3.24 igl_mesh_intersect() [1/4]

```
void netdem::igl_mesh_intersect (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab )
```

6.1.3.25 igl_mesh_intersect() [2/4]

```
void netdem::igl_mesh_intersect (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab,
            VecXT< int > *const jab )
```

6.1.3.26 igl_mesh_intersect() [3/4]

```
void netdem::igl_mesh_intersect (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            double dist_pc_to_plane,
            Vec3d const & dir_n,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab )
```

6.1.3.27 igl_mesh_intersect() [4/4]

```
void netdem::igl_mesh_intersect (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            double dist_pc_to_plane,
            Vec3d const & dir_n,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab,
            VecXT< int > *const jab )
```

6.1.3.28 igl_mesh_refine()

```
void netdem::igl_mesh_refine (
            VecXT< Vec3d > *const v,
            VecXT< Vec3i > *const f,
            int num_refines )
```

6.1.3.29 igl_points_inside_mesh()

```
VecXT< int > netdem::igl_points_inside_mesh (
            const VecXT< Vec3d > & v,
            const VecXT< Vec3i > & f,
            const VecXT< Vec3d > & v_querry )
```

6.1.3.30 igl_remove_duplicate_vertices() [1/2]

```
void netdem::igl_remove_duplicate_vertices (
            VecXT< Vec3d > *const v )
```

6.1.3.31 igl_remove_duplicate_vertices() [2/2]

```
void netdem::igl_remove_duplicate_vertices (
            VecXT< Vec3d > *const v,
            VecXT< Vec3i > *const f )
```

6.1.3.32 igl_remove_unreferenced_vertices()

```
void netdem::igl_remove_unreferenced_vertices (
            VecXT< Vec3d > *const v,
            VecXT< Vec3i > *const f )
```

6.1.3.33 igl_reorient_facets()

```
void netdem::igl_reorient_facets (
            const VecXT< Vec3d > & v,
            VecXT< Vec3i > * f )
```

6.1.3.34 igl_tetmesh_boundary() [1/2]

```
void netdem::igl_tetmesh_boundary (
            const VecXT< Vec4i > & tt,
            VecXT< Vec3i > *const ff )
```

6.1.3.35 igl_tetmesh_boundary() [2/2]

```
void netdem::igl_tetmesh_boundary (
            const VecXT< Vec4i > & tt,
            VecXT< Vec3i > *const ff,
            VecXT< int > *const fj )
```

6.1.3.36 ImportDataTxtToVec()

```
VecXT< VecXT< double > > netdem::ImportDataTxtToVec (
            std::string const & filename,
            int lines_to_skip = 0 )
```

6.1.3.37 my_to_string() [1/2]

```
std::string netdem::my_to_string (
            double value )  [inline]
```

6.1.3.38 my_to_string() [2/2]

```
std::string netdem::my_to_string (
            int value )  [inline]
```

6.1.3.39 operator*() [1/3]

```
Vec3d netdem::operator* (
            double lhs,
            Vec3d const & rhs )  [inline]
```

6.1.3.40 operator*() [2/3]

```
Vec3d netdem::operator* (
            Vec3d const & lhs,
            double rhs )  [inline]
```

6.1.3.41 operator*() [3/3]

```
Vec3d netdem::operator* (
            Vec3d const & lhs,
            Vec3d const & rhs )  [inline]
```

6.1.3.42 operator+() [1/5]

```
Vec3d netdem::operator+ (
            double lhs,
            Vec3d const & rhs )  [inline]
```

6.1.3.43 operator+() [2/5]

```
Vec3i netdem::operator+ (
            int lhs,
            Vec3i const & rhs )  [inline]
```

6.1.3.44 operator+() [3/5]

```
Vec3d netdem::operator+ (
            Vec3d const & lhs,
            double rhs )  [inline]
```

6.1.3.45 operator+() [4/5]

```
Vec3d netdem::operator+ (
            Vec3d const & lhs,
            Vec3d const & rhs )  [inline]
```

6.1.3.46 operator+() [5/5]

```
Vec3i netdem::operator+ (
            Vec3i const & lhs,
            int rhs )  [inline]
```

6.1.3.47 operator-() [1/3]

```
Vec3d netdem::operator- (
            double lhs,
            Vec3d const & rhs )  [inline]
```

6.1.3.48 operator-() [2/3]

```
Vec3d netdem::operator- (
            Vec3d const & lhs,
            double rhs )  [inline]
```

6.1.3.49 operator-() [3/3]

```
Vec3d netdem::operator- (
            Vec3d const & lhs,
            Vec3d const & rhs )  [inline]
```

6.1.3.50 operator/() [1/3]

```
Vec3d netdem::operator/ (
            double lhs,
            Vec3d const & rhs )  [inline]
```

6.1.3.51 operator/() [2/3]

```
Vec3d netdem::operator/ (
            Vec3d const & lhs,
            double rhs )  [inline]
```

6.1.3.52 operator/() [3/3]

```
Vec3d netdem::operator/ (
            Vec3d const & lhs,
            Vec3d const & rhs )  [inline]
```

6.1.3.53 operator<<() [1/4]

```
std::ostream & netdem::operator<< (
            std::ostream & os,
            Mat3d const & obj )  [inline]
```

6.1.3.54 operator<<() [2/4]

```
std::ostream & netdem::operator<< (
            std::ostream & os,
            Vec3d const & obj )  [inline]
```

6.1.3.55 operator<<() [3/4]

```
std::ostream & netdem::operator<< (
            std::ostream & os,
            Vec3i const & obj )  [inline]
```

6.1.3.56 operator<<() [4/4]

```
std::ostream & netdem::operator<< (
            std::ostream & os,
            Vec4d const & obj )  [inline]
```

6.1.3.57 PrintDebug()

```
void netdem::PrintDebug (
            std::string const & info )  [inline]
```

6.1.3.58 PrintError()

```
void netdem::PrintError (
            std::string const & info )  [inline]
```

6.1.3.59 PrintWarning()

```
void netdem::PrintWarning (
            std::string const & info )  [inline]
```

6.1.3.60 STDToEigen() [1/7]

```
void netdem::STDToEigen (
            const Mat3d & std_mat,
            Eigen::Matrix3d * eigen_mat )  [inline]
```

6.1.3.61 STDToEigen() [2/7]

```
void netdem::STDToEigen (
            const Vec3d & std_vec,
            Eigen::Vector3d * eigen_vec )  [inline]
```

6.1.3.62 STDToEigen() [3/7]

```
void netdem::STDToEigen (
            const VecXT< double > & std_vec,
            Eigen::VectorXd * eigen_vec )  [inline]
```

6.1.3.63 STDToEigen() [4/7]

```
void netdem::STDToEigen (
            const VecXT< Vec3d > & std_mat,
            Eigen::MatrixXd * eigen_mat )  [inline]
```

6.1.3.64 STDToEigen() [5/7]

```
void netdem::STDToEigen (
            const VecXT< Vec3i > & std_mat,
            Eigen::MatrixXi * eigen_mat )  [inline]
```

6.1.3.65 STDToEigen() [6/7]

```
void netdem::STDToEigen (
            const VecXT< Vec4i > & std_mat,
            Eigen::MatrixXi * eigen_mat )  [inline]
```

6.1.3.66 STDToEigen() [7/7]

```
void netdem::STDToEigen (
            const VecXT< VecXT< double > > & std_mat,
            Eigen::MatrixXd * eigen_mat )  [inline]
```

6.2 netdem::Math Namespace Reference

Namespaces

- namespace Quaternion

Functions

- template<typename T >
  int Sign (T val)
- double NormLen (Vec2d const &val)
- double NormLen (Vec3d const &val)
- double NormLen (double val_0, double val_1)
- double NormLen (double val_0, double val_1, double val_2)
- double NormLen (double val_0, double val_1, double val_2, double val_3)
- double Determinant (Mat2d const &mat)
- double Determinant (Mat3d const &mat)
- Mat2d Inverse (Mat2d const &m_val)
- Mat3d Inverse (Mat3d const &m_val)
- template<size_t r, size_t cr, size_t c>
  MatNd< r, c > Dot (MatNd< r, cr > const &m_1, MatNd< cr, c > const &m_2)
- template<size_t r, size_t cr, size_t c>
  MatNd< r, c > DotTransportLHS (MatNd< cr, r > const &m_1, MatNd< cr, c > const &m_2)
- template<size_t r, size_t cr, size_t c>
  MatNd< r, c > DotTransportRHS (MatNd< r, cr > const &m_1, MatNd< c, cr > const &m_2)
- Vec3d Cross (Vec3d const &val_1, Vec3d const &val_2)
- double Dot (Vec3d const &val_1, Vec3d const &val_2)
- double Dot (VecXT< double > const &val_1, VecXT< double > const &val_2)
- void Normalize (Vec3d *const val)
- Vec3d Rotate (Vec3d const &val_old, double rot_angle_cos, double rot_angle_sin, Vec3d const &rot_axis)
- Vec3d Rotate (Vec3d const &val_old, double rot_angle, Vec3d const &rot_axis)
- Vec3d Rotate (Vec3d const &val_old, Vec4d const &quat)
- Vec3d Rotate (Vec3d const &val_old, Mat3d const &rot_mat)
- Vec3d CartesianToSpherical (Vec3d const &vert_cart)
- Vec3d SphericalToCartesian (Vec3d const &vert_sph)

Variables

- constexpr double PI = 3.14159265358979323846264338327950288419 71
- constexpr double Infinity = 1.0e15

6.2.1 Function Documentation

6.2.1.1 CartesianToSpherical()

```
Vec3d netdem::Math::CartesianToSpherical (
            Vec3d const & vert_cart )  [inline]
```

6.2.1.2 Cross()

```
Vec3d netdem::Math::Cross (
            Vec3d const & val_1,
            Vec3d const & val_2 )  [inline]
```

6.2.1.3 Determinant() [1/2]

```
double netdem::Math::Determinant (
            Mat2d const & mat )  [inline]
```

6.2.1.4 Determinant() [2/2]

```
double netdem::Math::Determinant (
            Mat3d const & mat )  [inline]
```

6.2.1.5 Dot() [1/3]

```
template<size_t r, size_t cr, size_t c>
MatNd< r, c > netdem::Math::Dot (
            MatNd< r, cr > const & m_1,
            MatNd< cr, c > const & m_2 )  [inline]
```

6.2.1.6 Dot() [2/3]

```
double netdem::Math::Dot (
            Vec3d const & val_1,
            Vec3d const & val_2 )  [inline]
```

6.2.1.7 Dot() [3/3]

```
double netdem::Math::Dot (
            VecXT< double > const & val_1,
            VecXT< double > const & val_2 )  [inline]
```

6.2.1.8 DotTransportLHS()

```
template<size_t r, size_t cr, size_t c>
MatNd< r, c > netdem::Math::DotTransportLHS (
            MatNd< cr, r > const & m_1,
            MatNd< cr, c > const & m_2 )  [inline]
```

6.2.1.9 DotTransportRHS()

```
template<size_t r, size_t cr, size_t c>
MatNd< r, c > netdem::Math::DotTransportRHS (
            MatNd< r, cr > const & m_1,
            MatNd< c, cr > const & m_2 )  [inline]
```

6.2.1.10 Inverse() [1/2]

```
Mat2d netdem::Math::Inverse (
            Mat2d const & m_val )  [inline]
```

6.2.1.11 Inverse() [2/2]

```
Mat3d netdem::Math::Inverse (
            Mat3d const & m_val )  [inline]
```

6.2.1.12 Normalize()

```
void netdem::Math::Normalize (
            Vec3d *const val )  [inline]
```

6.2.1.13 NormLen() [1/5]

```
double netdem::Math::NormLen (
            double val_0,
            double val_1 )  [inline]
```

6.2.1.14 NormLen() [2/5]

```
double netdem::Math::NormLen (
            double val_0,
            double val_1,
            double val_2 )  [inline]
```

6.2.1.15 NormLen() [3/5]

```
double netdem::Math::NormLen (
            double val_0,
            double val_1,
            double val_2,
            double val_3 )  [inline]
```

6.2.1.16 NormLen() [4/5]

```
double netdem::Math::NormLen (
            Vec2d const & val )  [inline]
```

6.2.1.17 NormLen() [5/5]

```
double netdem::Math::NormLen (
            Vec3d const & val )  [inline]
```

6.2.1.18 Rotate() [1/4]

```
Vec3d netdem::Math::Rotate (
            Vec3d const & val_old,
            double rot_angle,
            Vec3d const & rot_axis )  [inline]
```

6.2.1.19 Rotate() [2/4]

```
Vec3d netdem::Math::Rotate (
            Vec3d const & val_old,
            double rot_angle_cos,
            double rot_angle_sin,
            Vec3d const & rot_axis )  [inline]
```

6.2.1.20 Rotate() [3/4]

```
Vec3d netdem::Math::Rotate (
            Vec3d const & val_old,
            Mat3d const & rot_mat )  [inline]
```

6.2.1.21 Rotate() [4/4]

```
Vec3d netdem::Math::Rotate (
            Vec3d const & val_old,
            Vec4d const & quat )  [inline]
```

6.2.1.22 Sign()

```
template<typename T >
int netdem::Math::Sign (
            T val )  [inline]
```

6.2.1.23 SphericalToCartesian()

```
Vec3d netdem::Math::SphericalToCartesian (
            Vec3d const & vert_sph )  [inline]
```

6.2.2 Variable Documentation

6.2.2.1 Infinity

```
constexpr double netdem::Math::Infinity = 1.0e15  [constexpr]
```

6.2.2.2 PI

```
constexpr double netdem::Math::PI = 3.14159265358979323846264338327950288419 71  [constexpr]
```

6.3 netdem::Math::Quaternion Namespace Reference

Functions

- Vec4d FromRodrigues (double rot_angle, Vec3d const &rot_axis)
- std::tuple< double, Vec3d > ToRodrigues (Vec4d const &quat)
- Vec4d FromMatrix (Mat3d const &rot_mat)
- Mat3d ToMatrix (Vec4d const &quat)
- Vec4d Multiply (Vec4d const &p, Vec4d const &q)
- Vec4d Add (Vec4d const &p, Vec4d const &q)
- Vec4d Conjugate (Vec4d const &p)
- void Normalize (Vec4d *const q)

6.3.1 Function Documentation

6.3.1.1 Add()

```
Vec4d netdem::Math::Quaternion::Add (
            Vec4d const & p,
            Vec4d const & q )  [inline]
```

6.3.1.2 Conjugate()

```
Vec4d netdem::Math::Quaternion::Conjugate (
            Vec4d const & p )  [inline]
```

6.3.1.3 FromMatrix()

```
Vec4d netdem::Math::Quaternion::FromMatrix (
            Mat3d const & rot_mat )  [inline]
```

6.3.1.4 FromRodrigues()

```
Vec4d netdem::Math::Quaternion::FromRodrigues (
            double rot_angle,
            Vec3d const & rot_axis )  [inline]
```

6.3.1.5 Multiply()

```
Vec4d netdem::Math::Quaternion::Multiply (
            Vec4d const & p,
            Vec4d const & q )  [inline]
```

6.3.1.6 Normalize()

```
void netdem::Math::Quaternion::Normalize (
            Vec4d *const q )  [inline]
```

6.3.1.7 ToMatrix()

```
Mat3d netdem::Math::Quaternion::ToMatrix (
            Vec4d const & quat )  [inline]
```

6.3.1.8 ToRodrigues()

```
std::tuple< double, Vec3d > netdem::Math::Quaternion::ToRodrigues (
            Vec4d const & quat )  [inline]
```

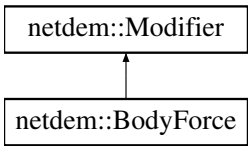
Chapter 7

Class Documentation

7.1 netdem::BodyForce Class Reference

`#include <body_force.hpp>`

Inheritance diagram for netdem::BodyForce:



Public Member Functions

- BodyForce (Vec3d const &b)
- BodyForce (double b_x, double b_y, double b_z)
- void SetParticlesFromScene ()
- void SetParticles (const VecXT< int > &id_list)
- void SetParticles (int num_ids,...)
- Modifier * Clone () const override
- void Execute () override
- void Update () override

Public Attributes

- VecXT< int > particle_id_list
- VecXT< Particle * > particle_list
- Vec3d unit_force {0, 0, 0}
- bool use_particles_in_scene {false}

7.1.1 Detailed Description

To add body force to the partilces. This is a pre-modifier, which will be excuted at the begining of a DEM cycle.

- particle_list: reference to the particles onto which the body force will be applied.
- unit_force[3]: the unit body force is each dimension. For example, the grabity force can be defined as {0, 0, -9.81}

7.1.2 Constructor & Destructor Documentation

7.1.2.1 BodyForce() [1/2]

```
netdem::BodyForce::BodyForce (
            Vec3d const & b )
```

7.1.2.2 BodyForce() [2/2]

```
netdem::BodyForce::BodyForce (
            double b_x,
            double b_y,
            double b_z )
```

7.1.3 Member Function Documentation

7.1.3.1 Clone()

```
Modifier * netdem::BodyForce::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.1.3.2 Execute()

```
void netdem::BodyForce::Execute ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.1.3.3 SetParticles() [1/2]

```
void netdem::BodyForce::SetParticles (
            const VecXT< int > & id_list )
```

7.1.3.4 SetParticles() [2/2]

```
void netdem::BodyForce::SetParticles (
            int num_ids,
             ... )
```

7.1.3.5 SetParticlesFromScene()

```
void netdem::BodyForce::SetParticlesFromScene ( )
```

7.1.3.6 Update()

```
void netdem::BodyForce::Update ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.1.4 Member Data Documentation

7.1.4.1 particle_id_list

```
VecXT<int> netdem::BodyForce::particle_id_list
```

7.1.4.2 particle_list

```
VecXT<Particle *> netdem::BodyForce::particle_list
```

7.1.4.3 unit_force

```
Vec3d netdem::BodyForce::unit_force {0, 0, 0}
```

7.1.4.4 use_particles_in_scene

```
bool netdem::BodyForce::use_particles_in_scene {false}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/body_force.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/body_force.cpp

7.2 netdem::BondedSpheres Class Reference

```
#include <bonded_spheres.hpp>
```

Public Member Functions

- BondedSpheres ()
- BondedSpheres (BondedSpheres const &bp)
- BondedSpheres (BondedSpheres const &&bp)
- BondedSpheres & operator= (BondedSpheres const &bp)
- BondedSpheres & operator= (BondedSpheres const &&bp)
- void SetBondModel (ContactModel *cnt_model)
- void Translate (double pos_x, double pos_y, double pos_z)
- void RotateByRodrigues (double rot_angle, double rot_axis_x, double rot_axis_y, double rot_axis_z)
- Vec3d GetCentroid ()
- void InitFromSTL (std::string const &filename, double sphere_size)
- void InitFromSTL (STLModel const &stl_model, double sphere_size)
- void InitFromGrid (double corner_x, double corner_y, double corner_z, double len_x, double len_y, double len_z, double sphere_size)
- void MakePorosity (double porosity)
- void InitBonds ()
- void ImportToScene (Scene *const scene) const

Public Attributes

- Sphere sphere
- VecXT< Particle > particle_list
- VecXT< ContactPP > contact_list
- VecXT< Vec2i > bond_pair_list
- ContactModel * bond_model {nullptr}

Private Member Functions

- void RefreshPointers ()

7.2.1 Constructor & Destructor Documentation

7.2.1.1 BondedSpheres() [1/3]

```
BondedSpheres::BondedSpheres ( )
```

7.2.1.2 BondedSpheres() [2/3]

```
BondedSpheres::BondedSpheres (
            BondedSpheres const & bp )
```

7.2.1.3 BondedSpheres() [3/3]

```
BondedSpheres::BondedSpheres (
            BondedSpheres const && bp )
```

7.2.2 Member Function Documentation

7.2.2.1 GetCentroid()

```
Vec3d BondedSpheres::GetCentroid ( )
```

7.2.2.2 ImportToScene()

```
void BondedSpheres::ImportToScene (
            Scene *const scene ) const
```

7.2.2.3 InitBonds()

```
void BondedSpheres::InitBonds ( )
```

7.2.2.4 InitFromGrid()

```
void BondedSpheres::InitFromGrid (
            double corner_x,
            double corner_y,
            double corner_z,
            double len_x,
            double len_y,
            double len_z,
            double sphere_size )
```

7.2.2.5 InitFromSTL() [1/2]

```
void BondedSpheres::InitFromSTL (
            std::string const & filename,
            double sphere_size )
```

7.2.2.6 InitFromSTL() [2/2]

```
void BondedSpheres::InitFromSTL (
            STLModel const & stl_model,
            double sphere_size )
```

7.2.2.7 MakePorosity()

```
void BondedSpheres::MakePorosity (
            double porosity )
```

7.2.2.8 operator=() [1/2]

```
BondedSpheres & BondedSpheres::operator= (
            BondedSpheres const && bp )
```

7.2.2.9 operator=() [2/2]

```
BondedSpheres & BondedSpheres::operator= (
            BondedSpheres const & bp )
```

7.2.2.10 RefreshPointers()

```
void BondedSpheres::RefreshPointers ( )  [private]
```

7.2.2.11 RotateByRodrigues()

```
void BondedSpheres::RotateByRodrigues (
            double rot_angle,
            double rot_axis_x,
            double rot_axis_y,
            double rot_axis_z )
```

7.2.2.12 SetBondModel()

```
void BondedSpheres::SetBondModel (
            ContactModel * cnt_model )
```

7.2.2.13 Translate()

```
void BondedSpheres::Translate (
            double pos_x,
            double pos_y,
            double pos_z )
```

7.2.3 Member Data Documentation

7.2.3.1 bond_model

```
ContactModel* netdem::BondedSpheres::bond_model {nullptr}
```

7.2.3.2 bond_pair_list

```
VecXT<Vec2i> netdem::BondedSpheres::bond_pair_list
```

7.2.3.3 contact_list

```
VecXT<ContactPP> netdem::BondedSpheres::contact_list
```

7.2.3.4 particle_list

```
VecXT<Particle> netdem::BondedSpheres::particle_list
```

7.2.3.5 sphere

```
Sphere netdem::BondedSpheres::sphere
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/bonded_spheres.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/bonded_spheres.cpp

7.3 netdem::BondedVoronois Class Reference

```
#include <bonded_voronois.hpp>
```

Public Member Functions

- BondedVoronois ()
- BondedVoronois (BondedVoronois const &bp)
- BondedVoronois (BondedVoronois const &&bp)
- BondedVoronois & operator= (BondedVoronois const &bp)
- BondedVoronois & operator= (BondedVoronois const &&bp)
- void SetBondModel (ContactModel *cnt_model)
- void Translate (double pos_x, double pos_y, double pos_z)
- void RotateByRodrigues (double rot_angle, double rot_axis_x, double rot_axis_y, double rot_axis_z)
- Vec3d GetCentroid ()
- void InitFromSTL (std::string const &filename, int num_voros)
- void InitFromSTL (STLModel const &stl_model, int num_voros)
- void MakePorosity (double porosity)
- void InitBonds ()
- void RefreshPointers ()
- void SaveAsVTK (std::string const &file_name)
- void ImportToScene (Scene *const scene) const

Public Attributes

- VecXT< TriMesh > trimesh_list
- VecXT< Particle > particle_list
- VecXT< ContactPP > contact_list
- VecXT< Vec2i > bond_pair_list
- int cvt_max_iters {1000}
- double cvt_tol {1.0e-3}
- ContactModel * bond_model {nullptr}

Private Member Functions

- VecXT< Vec3d > FindSharedVertices (STLModel const &stl_1, STLModel const &stl_2)
- Vec3d PolyNormal (VecXT< Vec3d > const &verts)
- void PolySortVertices (VecXT< Vec3d > *const pos_vec, Vec3d const &dir_n)
- std::tuple< Vec3d, double > PolyCentroid (const VecXT< Vec3d > &verts)

7.3.1 Constructor & Destructor Documentation

7.3.1.1 BondedVoronois() [1/3]

```
BondedVoronois::BondedVoronois ( )
```

7.3.1.2 BondedVoronois() [2/3]

```
BondedVoronois::BondedVoronois (
            BondedVoronois const & bp )
```

7.3.1.3 BondedVoronois() [3/3]

```
BondedVoronois::BondedVoronois (
            BondedVoronois const && bp )
```

7.3.2 Member Function Documentation

7.3.2.1 FindSharedVertices()

```
VecXT< Vec3d > BondedVoronois::FindSharedVertices (
            STLModel const & stl_1,
            STLModel const & stl_2 )  [private]
```

7.3.2.2 GetCentroid()

```
Vec3d BondedVoronois::GetCentroid ( )
```

7.3.2.3 ImportToScene()

```
void BondedVoronois::ImportToScene (
            Scene *const scene ) const
```

7.3.2.4 InitBonds()

```
void BondedVoronois::InitBonds ( )
```

7.3.2.5 InitFromSTL() [1/2]

```
void netdem::BondedVoronois::InitFromSTL (
            std::string const & filename,
            int num_voros )
```

7.3.2.6 InitFromSTL() [2/2]

```
void BondedVoronois::InitFromSTL (
            STLModel const & stl_model,
            int num_voros )
```

7.3.2.7 MakePorosity()

```
void BondedVoronois::MakePorosity (
            double porosity )
```

7.3.2.8 operator=() [1/2]

```
BondedVoronois & BondedVoronois::operator= (
            BondedVoronois const && bp )
```

7.3.2.9 operator=() [2/2]

```
BondedVoronois & BondedVoronois::operator= (
            BondedVoronois const & bp )
```

7.3.2.10 PolyCentroid()

```
tuple< Vec3d, double > BondedVoronois::PolyCentroid (
            const VecXT< Vec3d > & verts )  [private]
```

7.3.2.11 PolyNormal()

```
Vec3d BondedVoronois::PolyNormal (
            VecXT< Vec3d > const & verts )  [private]
```

7.3.2.12 PolySortVertices()

```
void BondedVoronois::PolySortVertices (
            VecXT< Vec3d > *const pos_vec,
            Vec3d const & dir_n )  [private]
```

7.3.2.13 RefreshPointers()

```
void BondedVoronois::RefreshPointers ( )
```

7.3.2.14 RotateByRodrigues()

```
void BondedVoronois::RotateByRodrigues (
            double rot_angle,
            double rot_axis_x,
            double rot_axis_y,
            double rot_axis_z )
```

7.3.2.15 SaveAsVTK()

```
void BondedVoronois::SaveAsVTK (
            std::string const & file_name )
```

7.3.2.16 SetBondModel()

```
void BondedVoronois::SetBondModel (
            ContactModel * cnt_model )
```

7.3.2.17 Translate()

```
void BondedVoronois::Translate (
            double pos_x,
            double pos_y,
            double pos_z )
```

7.3.3 Member Data Documentation

7.3.3.1 bond_model

```
ContactModel* netdem::BondedVoronois::bond_model {nullptr}
```

7.3.3.2 bond_pair_list

```
VecXT<Vec2i> netdem::BondedVoronois::bond_pair_list
```

7.3.3.3 contact_list

```
VecXT<ContactPP> netdem::BondedVoronois::contact_list
```

7.3.3.4 cvt_max_iters

```
int netdem::BondedVoronois::cvt_max_iters {1000}
```

7.3.3.5 cvt_tol

```
double netdem::BondedVoronois::cvt_tol {1.0e-3}
```

7.3.3.6 particle_list

```
VecXT<Particle> netdem::BondedVoronois::particle_list
```

7.3.3.7 trimesh_list

```
VecXT<TriMesh> netdem::BondedVoronois::trimesh_list
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/bonded_voronois.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/bonded_voronois.cpp

7.4 netdem::BondEntry Class Reference

```
#include <bond_entry.hpp>
```

Public Member Functions

- void UpdateForces (ContactPP *const cnt, double dt)
- void UpdateForces (ContactPW *const cnt, double dt)
- void UpdateLocalForces (ContactPP *const cnt, double dt)
- void UpdateLocalForces (ContactPW *const cnt, double dt)
- void UpdateGlobalForces ()

Public Attributes

- BondGeometries cnt_geoms
- ContactForces cnt_forces
- ContactModel * cnt_model {nullptr}

7.4.1 Member Function Documentation

7.4.1.1 UpdateForces() [1/2]

```
void BondEntry::UpdateForces (
            ContactPP *const cnt,
            double dt )
```

7.4.1.2 UpdateForces() [2/2]

```
void BondEntry::UpdateForces (
            ContactPW *const cnt,
            double dt )
```

7.4.1.3 UpdateGlobalForces()

```
void BondEntry::UpdateGlobalForces ( )
```

7.4.1.4 UpdateLocalForces() [1/2]

```
void BondEntry::UpdateLocalForces (
            ContactPP *const cnt,
            double dt )
```

7.4.1.5 UpdateLocalForces() [2/2]

```
void BondEntry::UpdateLocalForces (
            ContactPW *const cnt,
            double dt )
```

7.4.2 Member Data Documentation

7.4.2.1 cnt_forces

```
ContactForces netdem::BondEntry::cnt_forces
```

7.4.2.2 cnt_geoms

```
BondGeometries netdem::BondEntry::cnt_geoms
```

7.4.2.3 cnt_model

```
ContactModel* netdem::BondEntry::cnt_model {nullptr}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_entry.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_entry.cpp

7.5 netdem::BondEntryData Struct Reference

```
#include <bond_entry_data.hpp>
```

Public Attributes

- double pos [3] {0, 0, 0}
- double dir_n [3] {1, 0, 0}
- double dir_s [3] {0, 1, 0}
- double dir_t [3] {0, 0, 1}
- double branch_1 [3] {1, 0, 0}
- double branch_2 [3] {1, 0, 0}
- double pos_ini [3] {0, 0, 0}
- double dir_n_ini [3] {1, 0, 0}
- double dir_s_ini [3] {0, 1, 0}
- double dir_t_ini [3] {0, 0, 1}
- double pos_1_ini [3] {0, 0, 0}
- double pos_2_ini [3] {0, 0, 0}
- double quat_1_ini [4] {1, 0, 0, 0}
- double quat_2_ini [4] {1, 0, 0, 0}
- double radius {0}
- double fc_n {0}
- double fc_s {0}
- double fc_t {0}
- double mc_n {0}
- double mc_s {0}
- double mc_t {0}
- double fd_n {0}
- double fd_s {0}
- double fd_t {0}
- double md_n {0}
- double md_s {0}
- double md_t {0}
- int cnt_model_id {-1}

7.5.1 Member Data Documentation

7.5.1.1 branch_1

```
double netdem::BondEntryData::branch_1[3] {1, 0, 0}
```

7.5.1.2 branch_2

```
double netdem::BondEntryData::branch_2[3] {1, 0, 0}
```

7.5.1.3 cnt_model_id

```
int netdem::BondEntryData::cnt_model_id {-1}
```

7.5.1.4 dir_n

```
double netdem::BondEntryData::dir_n[3] {1, 0, 0}
```

7.5.1.5 dir_n_ini

```
double netdem::BondEntryData::dir_n_ini[3] {1, 0, 0}
```

7.5.1.6 dir_s

```
double netdem::BondEntryData::dir_s[3] {0, 1, 0}
```

7.5.1.7 dir_s_ini

```
double netdem::BondEntryData::dir_s_ini[3] {0, 1, 0}
```

7.5.1.8 dir_t

```
double netdem::BondEntryData::dir_t[3] {0, 0, 1}
```

7.5.1.9 dir_t_ini

```
double netdem::BondEntryData::dir_t_ini[3] {0, 0, 1}
```

7.5.1.10 fc_n

```
double netdem::BondEntryData::fc_n {0}
```

7.5.1.11 fc_s

```
double netdem::BondEntryData::fc_s {0}
```

7.5.1.12 fc_t

```
double netdem::BondEntryData::fc_t {0}
```

7.5.1.13 fd_n

```
double netdem::BondEntryData::fd_n {0}
```

7.5.1.14 fd_s

```
double netdem::BondEntryData::fd_s {0}
```

7.5.1.15 fd_t

```
double netdem::BondEntryData::fd_t {0}
```

7.5.1.16 mc_n

```
double netdem::BondEntryData::mc_n {0}
```

7.5.1.17 mc_s

```
double netdem::BondEntryData::mc_s {0}
```

7.5.1.18 mc_t

```
double netdem::BondEntryData::mc_t {0}
```

7.5.1.19 md_n

```
double netdem::BondEntryData::md_n {0}
```

7.5.1.20 md_s

```
double netdem::BondEntryData::md_s {0}
```

7.5.1.21 md_t

```
double netdem::BondEntryData::md_t {0}
```

7.5.1.22 pos

```
double netdem::BondEntryData::pos[3] {0, 0, 0}
```

7.5.1.23 pos_1_ini

```
double netdem::BondEntryData::pos_1_ini[3] {0, 0, 0}
```

7.5.1.24 pos_2_ini

```
double netdem::BondEntryData::pos_2_ini[3] {0, 0, 0}
```

7.5.1.25 pos_ini

```
double netdem::BondEntryData::pos_ini[3] {0, 0, 0}
```

7.5.1.26 quat_1_ini

```
double netdem::BondEntryData::quat_1_ini[4] {1, 0, 0, 0}
```

7.5.1.27 quat_2_ini

```
double netdem::BondEntryData::quat_2_ini[4] {1, 0, 0, 0}
```

7.5.1.28 radius

```
double netdem::BondEntryData::radius {0}
```

The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/bond_entry_data.hpp

7.6 netdem::BondEntryParser Class Reference

```
#include <bond_entry_parser.hpp>
```

Static Public Member Functions

- static void ClassToStruct (const BondEntry *const entry_class, BondEntryData *const entry_struct)
- static void StructToClass (BondEntry *const entry_class, const BondEntryData *const entry_struct, const MiniMap< int, ContactModel * > &contact_model_map)
- static void DefineMPIDataType (MPI_Datatype *const datatype)

7.6.1 Member Function Documentation

7.6.1.1 ClassToStruct()

```
void BondEntryParser::ClassToStruct (
            const BondEntry *const entry_class,
            BondEntryData *const entry_struct )  [static]
```

7.6.1.2 DefineMPIDataType()

```
void BondEntryParser::DefineMPIDataType (
            MPI_Datatype *const datatype )  [static]
```

7.6.1.3 StructToClass()

```
void BondEntryParser::StructToClass (
            BondEntry *const entry_class,
            const BondEntryData *const entry_struct,
            const MiniMap< int, ContactModel * > & contact_model_map )  [static]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/bond_entry_parser.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/bond_entry_parser.cpp

7.7 netdem::BondGeometries Class Reference

```
#include <bond_geometries.hpp>
```

Public Attributes

- Vec3d pos {0, 0, 0}
- Vec3d dir_n {1, 0, 0}
- Vec3d dir_s {0, 1, 0}
- Vec3d dir_t {0, 0, 1}
- Vec3d branch_1 {1, 0, 0}
- Vec3d branch_2 {1, 0, 0}
- Vec3d pos_ini {0, 0, 0}
- Vec3d dir_n_ini {1, 0, 0}
- Vec3d dir_s_ini {0, 1, 0}
- Vec3d dir_t_ini {0, 0, 1}
- Vec3d pos_1_ini {0, 0, 0}
- Vec3d pos_2_ini {0, 0, 0}
- Vec4d quat_1_ini {1, 0, 0, 0}
- Vec4d quat_2_ini {1, 0, 0, 0}
- double radius {0}
- double len_n {0}
- double len_s {0}
- double len_t {0}
- double theta_n {0}
- double theta_s {0}
- double theta_t {0}
- bool active {false}

7.7.1 Member Data Documentation

7.7.1.1 active

```
bool netdem::BondGeometries::active {false}
```

7.7.1.2 branch_1

```
Vec3d netdem::BondGeometries::branch_1 {1, 0, 0}
```

7.7.1.3 branch_2

```
Vec3d netdem::BondGeometries::branch_2 {1, 0, 0}
```

7.7.1.4 dir_n

```
Vec3d netdem::BondGeometries::dir_n {1, 0, 0}
```

7.7.1.5 dir_n_ini

```
Vec3d netdem::BondGeometries::dir_n_ini {1, 0, 0}
```

7.7.1.6 dir_s

```
Vec3d netdem::BondGeometries::dir_s {0, 1, 0}
```

7.7.1.7 dir_s_ini

```
Vec3d netdem::BondGeometries::dir_s_ini {0, 1, 0}
```

7.7.1.8 dir_t

```
Vec3d netdem::BondGeometries::dir_t {0, 0, 1}
```

7.7.1.9 dir_t_ini

```
Vec3d netdem::BondGeometries::dir_t_ini {0, 0, 1}
```

7.7.1.10 len_n

```
double netdem::BondGeometries::len_n {0}
```

7.7.1.11 len_s

```
double netdem::BondGeometries::len_s {0}
```

7.7.1.12 len_t

```
double netdem::BondGeometries::len_t {0}
```

7.7.1.13 pos

```
Vec3d netdem::BondGeometries::pos {0, 0, 0}
```

7.7.1.14 pos_1_ini

```
Vec3d netdem::BondGeometries::pos_1_ini {0, 0, 0}
```

7.7.1.15 pos_2_ini

```
Vec3d netdem::BondGeometries::pos_2_ini {0, 0, 0}
```

7.7.1.16 pos_ini

```
Vec3d netdem::BondGeometries::pos_ini {0, 0, 0}
```

7.7.1.17 quat_1_ini

```
Vec4d netdem::BondGeometries::quat_1_ini {1, 0, 0, 0}
```

7.7.1.18 quat_2_ini

```
Vec4d netdem::BondGeometries::quat_2_ini {1, 0, 0, 0}
```

7.7.1.19 radius

```
double netdem::BondGeometries::radius {0}
```

7.7.1.20 theta_n

```
double netdem::BondGeometries::theta_n {0}
```

7.7.1.21 theta_s

```
double netdem::BondGeometries::theta_s {0}
```

7.7.1.22 theta_t

```
double netdem::BondGeometries::theta_t {0}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_geometries.hpp

7.8 netdem::BondSolverPP Class Reference

```
#include <bond_solver_pp.hpp>
```

Public Member Functions

- BondSolverPP ()
- BondSolverPP (Particle *const p1, Particle *const p2)
- void Init (Particle *const p1, Particle *const p2)
- void ResolveInit (ContactPP *const cnt, double timestep)
- void ResolveUpdate (ContactPP *const cnt, double timestep)
- void ResolveInit (BondGeometries *const cnt_geoms, Vec3d const &bond_pos, Vec3d const &bond_dir_n, double bound_radius)
- void ResolveUpdate (BondGeometries *const cnt_geoms, double timestep)

Public Attributes

- Particle * particle_1 {nullptr}
- Particle * particle_2 {nullptr}

7.8.1 Constructor & Destructor Documentation

7.8.1.1 BondSolverPP() [1/2]

```
BondSolverPP::BondSolverPP ( )
```

7.8.1.2 BondSolverPP() [2/2]

```
BondSolverPP::BondSolverPP (
            Particle *const p1,
            Particle *const p2 )
```

7.8.2 Member Function Documentation

7.8.2.1 Init()

```
void BondSolverPP::Init (
            Particle *const p1,
            Particle *const p2 )
```

7.8.2.2 ResolveInit() [1/2]

```
void BondSolverPP::ResolveInit (
            BondGeometries *const cnt_geoms,
            Vec3d const & bond_pos,
            Vec3d const & bond_dir_n,
            double bound_radius )
```

7.8.2.3 ResolveInit() [2/2]

```
void BondSolverPP::ResolveInit (
            ContactPP *const cnt,
            double timestep )
```

7.8.2.4 ResolveUpdate() [1/2]

```
void BondSolverPP::ResolveUpdate (
            BondGeometries *const cnt_geoms,
            double timestep )
```

7.8.2.5 ResolveUpdate() [2/2]

```
void BondSolverPP::ResolveUpdate (
            ContactPP *const cnt,
            double timestep )
```

7.8.3 Member Data Documentation

7.8.3.1 particle_1

```
Particle* netdem::BondSolverPP::particle_1 {nullptr}
```

7.8.3.2 particle_2

```
Particle * netdem::BondSolverPP::particle_2 {nullptr}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_solver_pp.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_solver_pp.cpp

7.9 netdem::BondSolverPW Class Reference

```
#include <bond_solver_pw.hpp>
```

Public Member Functions

- BondSolverPW ()
- BondSolverPW (Particle *const p, Wall *const w)
- void Init (Particle *const p, Wall *const w)
- void ResolveInit (ContactPW *const cnt, double timestep)
- void ResolveUpdate (ContactPW *const cnt, double timestep)
- void ResolveInit (BondGeometries *const cnt_geoms, Vec3d const &bond_pos, Vec3d const &bond_dir_n, double bound_radius)
- void ResolveUpdate (BondGeometries *const cnt_geoms, double timestep)

Public Attributes

- Particle * particle {nullptr}
- Wall * wall {nullptr}

7.9.1 Constructor & Destructor Documentation

7.9.1.1 BondSolverPW() [1/2]

```
BondSolverPW::BondSolverPW ( )
```

7.9.1.2 BondSolverPW() [2/2]

```
BondSolverPW::BondSolverPW (
            Particle *const p,
            Wall *const w )
```

7.9.2 Member Function Documentation

7.9.2.1 Init()

```
void BondSolverPW::Init (
            Particle *const p,
            Wall *const w )
```

7.9.2.2 ResolveInit() [1/2]

```
void BondSolverPW::ResolveInit (
            BondGeometries *const cnt_geoms,
            Vec3d const & bond_pos,
            Vec3d const & bond_dir_n,
            double bound_radius )
```

7.9.2.3 ResolveInit() [2/2]

```
void BondSolverPW::ResolveInit (
            ContactPW *const cnt,
            double timestep )
```

7.9.2.4 ResolveUpdate() [1/2]

```
void BondSolverPW::ResolveUpdate (
            BondGeometries *const cnt_geoms,
            double timestep )
```

7.9.2.5 ResolveUpdate() [2/2]

```
void BondSolverPW::ResolveUpdate (
            ContactPW *const cnt,
            double timestep )
```

7.9.3 Member Data Documentation

7.9.3.1 particle

```
Particle* netdem::BondSolverPW::particle {nullptr}
```

7.9.3.2 wall

```
Wall* netdem::BondSolverPW::wall {nullptr}
```

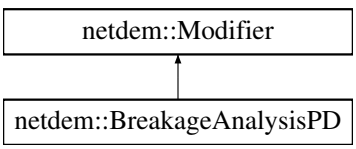
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_solver_pw.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_solver_pw.cpp

7.10 netdem::BreakageAnalysisPD Class Reference

```
#include <breakage_analysis_pd.hpp>
```

Inheritance diagram for netdem::BreakageAnalysisPD:



Public Member Functions

- BreakageAnalysisPD ()
- void SetRootPath (std::string const &root_path)
- void SetFrequency (bool save_by_cycles, double interval)
- void SetParticlesFromScene ()
- void SetParticles (const VecXT< int > &id_list)
- void SetParticles (const std::initializer_list< int > &id_list)
- Modifier * Clone () const override
- void Init (Simulation *sim) override
- void Execute () override
- void Update ()

Public Attributes

- VecXT< int > particle_id_list
- VecXT< Particle * > particle_list
- bool use_particles_in_scene {false}
- PeriDigmDEMCoupler pd_dem_coupler

Private Member Functions

- bool CheckIfToExecute ()

Private Attributes

- std::string root_path {"tmp/out/"}
- bool excute_by_cycles {true}
- int cycle_interval {0}
- int cycle_previous {0}
- double time_interval {0}
- double time_previous {0}

7.10.1 Constructor & Destructor Documentation

7.10.1.1 BreakageAnalysisPD()

```
netdem::BreakageAnalysisPD::BreakageAnalysisPD ( )
```

7.10.2 Member Function Documentation

7.10.2.1 CheckIfToExecute()

```
bool netdem::BreakageAnalysisPD::CheckIfToExecute ( )  [private]
```

7.10.2.2 Clone()

```
Modifier * netdem::BreakageAnalysisPD::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.10.2.3 Execute()

```
void netdem::BreakageAnalysisPD::Execute ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.10.2.4 Init()

```
void netdem::BreakageAnalysisPD::Init (
            Simulation * sim )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.10.2.5 SetFrequency()

```
void netdem::BreakageAnalysisPD::SetFrequency (
            bool save_by_cycles,
            double interval )
```

7.10.2.6 SetParticles() [1/2]

```
void netdem::BreakageAnalysisPD::SetParticles (
            const std::initializer_list< int > & id_list )
```

7.10.2.7 SetParticles() [2/2]

```
void netdem::BreakageAnalysisPD::SetParticles (
            const VecXT< int > & id_list )
```

7.10.2.8 SetParticlesFromScene()

```
void netdem::BreakageAnalysisPD::SetParticlesFromScene ( )
```

7.10.2.9 SetRootPath()

```
void netdem::BreakageAnalysisPD::SetRootPath (
            std::string const & root_path )
```

7.10.2.10 Update()

```
void netdem::BreakageAnalysisPD::Update ( )  [virtual]
```

Reimplemented from netdem::Modifier.

7.10.3 Member Data Documentation

7.10.3.1 cycle_interval

```
int netdem::BreakageAnalysisPD::cycle_interval {0}  [private]
```

7.10.3.2 cycle_previous

```
int netdem::BreakageAnalysisPD::cycle_previous {0}  [private]
```

7.10.3.3 excute_by_cycles

```
bool netdem::BreakageAnalysisPD::excute_by_cycles {true}  [private]
```

7.10.3.4 particle_id_list

```
VecXT<int> netdem::BreakageAnalysisPD::particle_id_list
```

7.10.3.5 particle_list

```
VecXT<Particle *> netdem::BreakageAnalysisPD::particle_list
```

7.10.3.6 pd_dem_coupler

```
PeriDigmDEMCoupler netdem::BreakageAnalysisPD::pd_dem_coupler
```

7.10.3.7 root_path

```
std::string netdem::BreakageAnalysisPD::root_path {"tmp/out/"}  [private]
```

7.10.3.8 time_interval

```
double netdem::BreakageAnalysisPD::time_interval {0}  [private]
```

7.10.3.9 time_previous

```
double netdem::BreakageAnalysisPD::time_previous {0}  [private]
```

7.10.3.10 use_particles_in_scene

```
bool netdem::BreakageAnalysisPD::use_particles_in_scene {false}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/breakage_analysis_pd.hp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/breakage_analysis_pd.cp

7.11 netdem::Cell Class Reference

`#include <cell.hpp>`

Public Member Functions

- Cell ()
- Cell (Vec3d const &bmin, Vec3d const &bmax)
- bool IsJudgeCell (Particle const &p, Particle const &q)
- bool IsJudgeCell (Particle const &p, Wall const &w)
- void ClearLinkedLists ()
- STLModel GetSTLModel ()

  for visualization purpose
- ~Cell ()
- void Print ()

Public Attributes

- Vec3d bound_min {-0.5, -0.5, -0.5}
- Vec3d bound_max {0.5, 0.5, 0.5}
- VecXT< std::pair< Particle *, int > > linked_particle_list

  this list is maintained by UpdateLinkedCells in particle
- VecXT< std::pair< Wall *, int > > linked_wall_list

  this list is maintained by UpdateLinkedCells in wall

7.11.1 Detailed Description

cell is used for broad-phase contact detection. The particles in a sub-domain will first be sorted into cells, so that only the particles that fall in the same cell could be potentially contact with each other.

- bound_min, bound_max: coordinates of the lower and upper boundary of the rectangle cell.
- particle_list: particles that fall in this cell.
- wall_list: walls that fall in this cell.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Cell() [1/2]

`Cell::Cell ( )`

7.11.2.2 Cell() [2/2]

```
Cell::Cell (
            Vec3d const & bmin,
            Vec3d const & bmax )
```

7.11.2.3 ~Cell()

```
Cell::~Cell ( )
```

7.11.3 Member Function Documentation

7.11.3.1 ClearLinkedLists()

```
void Cell::ClearLinkedLists ( )
```

7.11.3.2 GetSTLModel()

```
STLModel Cell::GetSTLModel ( )
```

for visualization purpose

7.11.3.3 IsJudgeCell() [1/2]

```
bool Cell::IsJudgeCell (
            Particle const & p,
            Particle const & q )
```

7.11.3.4 IsJudgeCell() [2/2]

```
bool Cell::IsJudgeCell (
            Particle const & p,
            Wall const & w )
```

7.11.3.5 Print()

```
void Cell::Print ( )
```

7.11.4 Member Data Documentation

7.11.4.1 bound_max

```
Vec3d netdem::Cell::bound_max {0.5, 0.5, 0.5}
```

7.11.4.2 bound_min

```
Vec3d netdem::Cell::bound_min {-0.5, -0.5, -0.5}
```

7.11.4.3 linked_particle_list

```
VecXT<std::pair<Particle *, int> > netdem::Cell::linked_particle_list
```

this list is maintained by UpdateLinkedCells in particle

7.11.4.4 linked_wall_list

```
VecXT<std::pair<Wall *, int> > netdem::Cell::linked_wall_list
```

this list is maintained by UpdateLinkedCells in wall

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/cell.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/cell.cpp

7.12 netdem::CellManager Class Reference

```
#include <cell_manager.hpp>
```

Public Member Functions

- CellManager ()
- void Init ()
- void SetBound (double bmin_x, double bmin_y, double bmin_z, double bmax_x, double bmax_y, double bmax_z)
- void SetSpacing (double s_x, double s_y, double s_z)
- std::tuple< Vec3i, Vec3i > GetOverlappedCells (Vec3d const &bmin, Vec3d const &bmax)
- STLModel GetSTLModel ()

  for visualization purpose

Public Attributes

- Vec3d bound_min {-0.5, -0.5, -0.5}
- Vec3d bound_max {0.5, 0.5, 0.5}
- Vec3d spacing {1.0, 1.0, 1.0}
- Vec3i cell_size {3, 3, 3}
- VecXT< VecXT< VecXT< Cell > > > cell_list

7.12.1 Constructor & Destructor Documentation

7.12.1.1 CellManager()

```
CellManager::CellManager ( )
```

7.12.2 Member Function Documentation

7.12.2.1 GetOverlappedCells()

```
tuple< Vec3i, Vec3i > CellManager::GetOverlappedCells (
            Vec3d const & bmin,
            Vec3d const & bmax )
```

7.12.2.2 GetSTLModel()

```
STLModel CellManager::GetSTLModel ( )
```

for visualization purpose

7.12.2.3 Init()

```
void CellManager::Init ( )
```

7.12.2.4 SetBound()

```
void CellManager::SetBound (
            double bmin_x,
            double bmin_y,
            double bmin_z,
            double bmax_x,
            double bmax_y,
            double bmax_z )
```

7.12.2.5 SetSpacing()

```
void CellManager::SetSpacing (
            double s_x,
            double s_y,
            double s_z )
```

7.12.3 Member Data Documentation

7.12.3.1 bound_max

```
Vec3d netdem::CellManager::bound_max {0.5, 0.5, 0.5}
```

7.12.3.2 bound_min

```
Vec3d netdem::CellManager::bound_min {-0.5, -0.5, -0.5}
```

7.12.3.3 cell_list

```
VecXT<VecXT<VecXT<Cell> > > netdem::CellManager::cell_list
```

7.12.3.4 cell_size

```
Vec3i netdem::CellManager::cell_size {3, 3, 3}
```

7.12.3.5 spacing

```
Vec3d netdem::CellManager::spacing {1.0, 1.0, 1.0}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/cell_manager.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/cell_manager.cpp

7.13 netdem::CollisionEntry Class Reference

```
#include <collision_entry.hpp>
```

Public Member Functions

- void UpdateForces (ContactPP ∗const cnt, double dt)
- void UpdateForces (ContactPW ∗const cnt, double dt)
- void UpdateLocalForces (ContactPP ∗const cnt, double dt)
- void UpdateLocalForces (ContactPW ∗const cnt, double dt)
- void UpdateGlobalForces ()

Public Attributes

- CollisionGeometries cnt_geoms
- ContactForces cnt_forces
- ContactModel ∗ cnt_model {nullptr}

7.13.1 Member Function Documentation

7.13.1.1 UpdateForces() [1/2]

```
void CollisionEntry::UpdateForces (
            ContactPP *const cnt,
            double dt )
```

7.13.1.2 UpdateForces() [2/2]

```
void CollisionEntry::UpdateForces (
            ContactPW *const cnt,
            double dt )
```

7.13.1.3 UpdateGlobalForces()

```
void CollisionEntry::UpdateGlobalForces ( )
```

7.13.1.4 UpdateLocalForces() [1/2]

```
void CollisionEntry::UpdateLocalForces (
            ContactPP *const cnt,
            double dt )
```

7.13.1.5 UpdateLocalForces() [2/2]

```
void CollisionEntry::UpdateLocalForces (
            ContactPW *const cnt,
            double dt )
```

7.13.2 Member Data Documentation

7.13.2.1 cnt_forces

```
ContactForces netdem::CollisionEntry::cnt_forces
```

7.13.2.2 cnt_geoms

```
CollisionGeometries netdem::CollisionEntry::cnt_geoms
```

7.13.2.3 cnt_model

```
ContactModel* netdem::CollisionEntry::cnt_model {nullptr}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/collision_entry.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/collision_entry.cpp

7.14 netdem::CollisionEntryData Struct Reference

```
#include <collision_entry_data.hpp>
```

Public Attributes

- double pos [3] {0, 0, 0}
- double dir_n [3] {1, 0, 0}
- double dir_s [3] {0, 1, 0}
- double dir_t [3] {0, 0, 1}
- double branch_1 [3] {1, 0, 0}
- double branch_2 [3] {-1, 0, 0}
- int node_id {0}
- double fc_n {0}
- double fc_s {0}
- double fc_t {0}
- double mc_n {0}
- double mc_s {0}
- double mc_t {0}
- double fd_n {0}
- double fd_s {0}
- double fd_t {0}
- double md_n {0}
- double md_s {0}
- double md_t {0}
- int cnt_model_id {-1}

7.14.1 Member Data Documentation

7.14.1.1 branch_1

```
double netdem::CollisionEntryData::branch_1[3] {1, 0, 0}
```

7.14.1.2 branch_2

```
double netdem::CollisionEntryData::branch_2[3] {-1, 0, 0}
```

7.14.1.3 cnt_model_id

```
int netdem::CollisionEntryData::cnt_model_id {-1}
```

7.14.1.4 dir_n

```
double netdem::CollisionEntryData::dir_n[3] {1, 0, 0}
```

7.14.1.5 dir_s

```
double netdem::CollisionEntryData::dir_s[3] {0, 1, 0}
```

7.14.1.6 dir_t

```
double netdem::CollisionEntryData::dir_t[3] {0, 0, 1}
```

7.14.1.7 fc_n

```
double netdem::CollisionEntryData::fc_n {0}
```

7.14.1.8 fc_s

```
double netdem::CollisionEntryData::fc_s {0}
```

7.14.1.9 fc_t

```
double netdem::CollisionEntryData::fc_t {0}
```

7.14.1.10 fd_n

```
double netdem::CollisionEntryData::fd_n {0}
```

7.14.1.11 fd_s

```
double netdem::CollisionEntryData::fd_s {0}
```

7.14.1.12 fd_t

```
double netdem::CollisionEntryData::fd_t {0}
```

7.14.1.13 mc_n

```
double netdem::CollisionEntryData::mc_n {0}
```

7.14.1.14 mc_s

```
double netdem::CollisionEntryData::mc_s {0}
```

7.14.1.15 mc_t

```
double netdem::CollisionEntryData::mc_t {0}
```

7.14.1.16 md_n

```
double netdem::CollisionEntryData::md_n {0}
```

7.14.1.17 md_s

```
double netdem::CollisionEntryData::md_s {0}
```

7.14.1.18 md_t

```
double netdem::CollisionEntryData::md_t {0}
```

7.14.1.19 node_id

```
int netdem::CollisionEntryData::node_id {0}
```

7.14.1.20 pos

```
double netdem::CollisionEntryData::pos[3] {0, 0, 0}
```

The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/collision_entry_data.hpp

7.15 netdem::CollisionEntryParser Class Reference

```
#include <collision_entry_parser.hpp>
```

Static Public Member Functions

- static void ClassToStruct (const CollisionEntry *const entry_class, CollisionEntryData *const entry_struct)
- static void StructToClass (CollisionEntry *const entry_class, const CollisionEntryData *const entry_struct, const MiniMap< int, ContactModel * > &contact_model_map)
- static void DefineMPIDataType (MPI_Datatype *const datatype)

7.15.1 Member Function Documentation

7.15.1.1 ClassToStruct()

```
void CollisionEntryParser::ClassToStruct (
            const CollisionEntry *const entry_class,
            CollisionEntryData *const entry_struct )  [static]
```

7.15.1.2 DefineMPIDataType()

```
void CollisionEntryParser::DefineMPIDataType (
            MPI_Datatype *const datatype )  [static]
```

7.15.1.3 StructToClass()

```
void CollisionEntryParser::StructToClass (
            CollisionEntry *const entry_class,
            const CollisionEntryData *const entry_struct,
            const MiniMap< int, ContactModel * > & contact_model_map )  [static]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/collision_entry_parser.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/collision_entry_parser.cpp

7.16 netdem::CollisionGeometries Class Reference

`#include <collision_geometries.hpp>`

Public Attributes

- Vec3d pos {0, 0, 0}
- Vec3d dir_n {1, 0, 0}
- Vec3d dir_s {0, 1, 0}
- Vec3d dir_t {0, 0, 1}
- Vec3d branch_1 {1, 0, 0}
- Vec3d branch_2 {1, 0, 0}
- double len_n {0}
- double dlen_n {0}
- double dlen_s {0}
- double dlen_t {0}
- double dtheta_n {0}
- double dtheta_s {0}
- double dtheta_t {0}
- double radius_1 {1}
- double radius_2 {1}
- bool active {true}
- int node_id {0}
- double node_dist {0}
- double vol {0}
- double sn {0}

7.16.1 Detailed Description

- pos: coordinates of the contact position (it could be the centroid of the contact region). The contact position is introduced, such that the contact forces and contact moments are assumed to concentrated at the contact position.
- dir_n, dir_s, dir_t: directions of the contact normal force, and tangential forces, respectively. Note that the tangential directions need to be characterized with two vectors in the 3D framework.
- branch_1, branch_2: vectors that start from the particle centroid and point to the contact position.
- len_n normal overlapping depth. It is used to calculate the contact normal force. For example, in the linear spring contact model, the contact normal force is assumed to be linearly proportional to the normal overlapping depth.
- dlen_s, dlen_t: relative tangential displacements/deformations. The relative tangential displacements/deformation increase as the stangential/shear forces increase.
- radius_1, radius_2: curvature of the particle surface at the contact region. The curvatures are used in Hertz-based contact models.

7.16.2 Member Data Documentation

7.16.2.1 active

```
bool netdem::CollisionGeometries::active {true}
```

7.16.2.2 branch_1

```
Vec3d netdem::CollisionGeometries::branch_1 {1, 0, 0}
```

7.16.2.3 branch_2

```
Vec3d netdem::CollisionGeometries::branch_2 {1, 0, 0}
```

7.16.2.4 dir_n

```
Vec3d netdem::CollisionGeometries::dir_n {1, 0, 0}
```

7.16.2.5 dir_s

```
Vec3d netdem::CollisionGeometries::dir_s {0, 1, 0}
```

7.16.2.6 dir_t

```
Vec3d netdem::CollisionGeometries::dir_t {0, 0, 1}
```

7.16.2.7 dlen_n

```
double netdem::CollisionGeometries::dlen_n {0}
```

7.16.2.8 dlen_s

```
double netdem::CollisionGeometries::dlen_s {0}
```

7.16.2.9 dlen_t

```
double netdem::CollisionGeometries::dlen_t {0}
```

7.16.2.10 dtheta_n

```
double netdem::CollisionGeometries::dtheta_n {0}
```

7.16.2.11 dtheta_s

```
double netdem::CollisionGeometries::dtheta_s {0}
```

7.16.2.12 dtheta_t

```
double netdem::CollisionGeometries::dtheta_t {0}
```

7.16.2.13 len_n

```
double netdem::CollisionGeometries::len_n {0}
```

7.16.2.14 node_dist

```
double netdem::CollisionGeometries::node_dist {0}
```

7.16.2.15 node_id

```
int netdem::CollisionGeometries::node_id {0}
```

7.16.2.16 pos

```
Vec3d netdem::CollisionGeometries::pos {0, 0, 0}
```

7.16.2.17 radius_1

```
double netdem::CollisionGeometries::radius_1 {1}
```

7.16.2.18 radius_2

```
double netdem::CollisionGeometries::radius_2 {1}
```

7.16.2.19 sn

```
double netdem::CollisionGeometries::sn {0}
```

7.16.2.20 vol

```
double netdem::CollisionGeometries::vol {0}
```

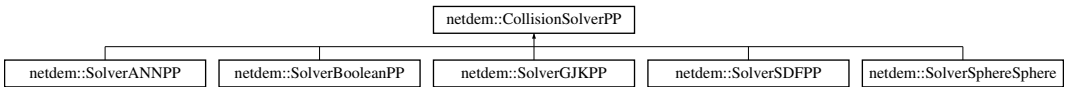
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/collision_geometries.hpp

7.17 netdem::CollisionSolverPP Class Reference

```
#include <collision_solver_pp.hpp>
```

Inheritance diagram for netdem::CollisionSolverPP:



Public Member Functions

- CollisionSolverPP ()
- CollisionSolverPP (Particle ∗const p1, Particle ∗const p2)
- virtual CollisionSolverPP ∗ Clone () const =0
- virtual void Init (Particle ∗const p1, Particle ∗const p2)
- virtual bool Detect ()=0
- virtual bool Detect (ContactPP ∗const cnt)
- virtual void ResolveInit (ContactPP ∗const cnt, double timestep)=0
- virtual void ResolveUpdate (ContactPP ∗const cnt, double timestep)=0
- virtual ∼CollisionSolverPP ()

Public Attributes

- Particle ∗ particle_1 {nullptr}
- Particle ∗ particle_2 {nullptr}

Protected Member Functions

- void InitBasicGeoms (CollisionGeometries ∗const cnt_geoms, double timestep)
- void UpdateBasicGeoms (CollisionGeometries ∗const cnt_geoms, double timestep, Vec3d const &dir_n_old)

7.17.1 Detailed Description

interface for particle and particle contact solver.

- particle_1, particle_2: pointers of particle 1 and 2, respectively.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 CollisionSolverPP() [1/2]

```
netdem::CollisionSolverPP::CollisionSolverPP ( )  [inline]
```

7.17.2.2 CollisionSolverPP() [2/2]

```
netdem::CollisionSolverPP::CollisionSolverPP (
            Particle *const p1,
            Particle *const p2 )  [inline]
```

7.17.2.3 ~CollisionSolverPP()

```
virtual netdem::CollisionSolverPP::~CollisionSolverPP ( )  [inline], [virtual]
```

7.17.3 Member Function Documentation

7.17.3.1 Clone()

```
virtual CollisionSolverPP * netdem::CollisionSolverPP::Clone ( ) const  [pure virtual]
```

Implemented in netdem::SolverBooleanPP, netdem::SolverGJKPP, netdem::SolverSDFPP, netdem::SolverSphereSphere, and netdem::SolverANNPP.

7.17.3.2 Detect() [1/2]

```
virtual bool netdem::CollisionSolverPP::Detect ( )  [pure virtual]
```

Implemented in netdem::SolverBooleanPP, netdem::SolverGJKPP, netdem::SolverSDFPP, netdem::SolverSphereSphere, and netdem::SolverANNPP.

7.17.3.3 Detect() [2/2]

```
virtual bool netdem::CollisionSolverPP::Detect (
            ContactPP *const cnt )  [inline], [virtual]
```

7.17.3.4 Init()

```
virtual void netdem::CollisionSolverPP::Init (
            Particle *const p1,
            Particle *const p2 )  [inline], [virtual]
```

Reimplemented in netdem::SolverBooleanPP, netdem::SolverGJKPP, netdem::SolverSDFPP, netdem::SolverSphereSphere, and netdem::SolverANNPP.

7.17.3.5 InitBasicGeoms()

```
void netdem::CollisionSolverPP::InitBasicGeoms (
            CollisionGeometries *const cnt_geoms,
            double timestep )  [inline], [protected]
```

7.17.3.6 ResolveInit()

```
virtual void netdem::CollisionSolverPP::ResolveInit (
            ContactPP *const cnt,
            double timestep )  [pure virtual]
```

Implemented in netdem::SolverBooleanPP, netdem::SolverGJKPP, netdem::SolverSDFPP, netdem::SolverSphereSphere, and netdem::SolverANNPP.

7.17.3.7 ResolveUpdate()

```
virtual void netdem::CollisionSolverPP::ResolveUpdate (
            ContactPP *const cnt,
            double timestep )  [pure virtual]
```

Implemented in netdem::SolverBooleanPP, netdem::SolverGJKPP, netdem::SolverSDFPP, netdem::SolverSphereSphere, and netdem::SolverANNPP.

7.17.3.8 UpdateBasicGeoms()

```
void netdem::CollisionSolverPP::UpdateBasicGeoms (
            CollisionGeometries *const cnt_geoms,
            double timestep,
            Vec3d const & dir_n_old )  [inline], [protected]
```

7.17.4 Member Data Documentation

7.17.4.1 particle_1

```
Particle* netdem::CollisionSolverPP::particle_1 {nullptr}
```

7.17.4.2 particle_2

```
Particle * netdem::CollisionSolverPP::particle_2 {nullptr}
```

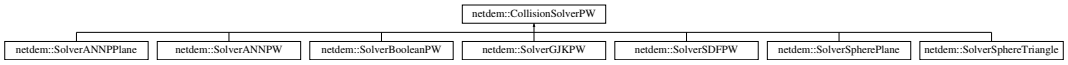
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/collision_solver_pp.hpp

7.18 netdem::CollisionSolverPW Class Reference

```
#include <collision_solver_pw.hpp>
```

Inheritance diagram for netdem::CollisionSolverPW:



Public Member Functions

- CollisionSolverPW ()
- CollisionSolverPW (Particle *const p, Wall *const w)
- virtual CollisionSolverPW * Clone () const =0
- virtual void Init (Particle *const p, Wall *const w)
- virtual bool Detect ()=0
- virtual bool Detect (ContactPW *const cnt)
- virtual void ResolveInit (ContactPW *const cnt, double timestep)=0
- virtual void ResolveUpdate (ContactPW *const cnt, double timestep)=0
- virtual ~CollisionSolverPW ()

Public Attributes

- Particle * particle {nullptr}
- Wall * wall {nullptr}

Protected Member Functions

- void InitBasicGeoms (CollisionGeometries *const cnt_geoms, double timestep)
- void UpdateBasicGeoms (CollisionGeometries *const cnt_geoms, double timestep, Vec3d const &dir_n_old)

7.18.1 Detailed Description

interface for particle and wall contact solver.

- particle, wall: pointers of particle and wall, respectively.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 CollisionSolverPW() [1/2]

```
netdem::CollisionSolverPW::CollisionSolverPW ( )  [inline]
```

7.18.2.2 CollisionSolverPW() [2/2]

```
netdem::CollisionSolverPW::CollisionSolverPW (
            Particle *const p,
            Wall *const w )  [inline]
```

7.18.2.3 ~CollisionSolverPW()

```
virtual netdem::CollisionSolverPW::~CollisionSolverPW ( )  [inline], [virtual]
```

7.18.3 Member Function Documentation

7.18.3.1 Clone()

```
virtual CollisionSolverPW * netdem::CollisionSolverPW::Clone ( ) const  [pure virtual]
```

Implemented in netdem::SolverBooleanPW, netdem::SolverGJKPW, netdem::SolverSDFPW, netdem::SolverSpherePlane, netdem::SolverSphereTriangle, netdem::SolverANNPPlane, and netdem::SolverANNPW.

7.18.3.2 Detect() [1/2]

```
virtual bool netdem::CollisionSolverPW::Detect ( )  [pure virtual]
```

Implemented in netdem::SolverBooleanPW, netdem::SolverGJKPW, netdem::SolverSDFPW, netdem::SolverSpherePlane, netdem::SolverSphereTriangle, netdem::SolverANNPPlane, and netdem::SolverANNPW.

7.18.3.3 Detect() [2/2]

```
virtual bool netdem::CollisionSolverPW::Detect (
            ContactPW *const cnt )  [inline], [virtual]
```

7.18.3.4 Init()

```
virtual void netdem::CollisionSolverPW::Init (
            Particle *const p,
            Wall *const w )  [inline], [virtual]
```

Reimplemented in netdem::SolverBooleanPW, netdem::SolverGJKPW, netdem::SolverSDFPW, netdem::SolverSpherePlane, netdem::SolverSphereTriangle, netdem::SolverANNPPlane, and netdem::SolverANNPW.

7.18.3.5 InitBasicGeoms()

```
void netdem::CollisionSolverPW::InitBasicGeoms (
            CollisionGeometries *const cnt_geoms,
            double timestep )  [inline], [protected]
```

7.18.3.6 ResolveInit()

```
virtual void netdem::CollisionSolverPW::ResolveInit (
            ContactPW *const cnt,
            double timestep )  [pure virtual]
```

Implemented in netdem::SolverBooleanPW, netdem::SolverGJKPW, netdem::SolverSDFPW, netdem::SolverSpherePlane, netdem::SolverSphereTriangle, netdem::SolverANNPPlane, and netdem::SolverANNPW.

7.18.3.7 ResolveUpdate()

```
virtual void netdem::CollisionSolverPW::ResolveUpdate (
            ContactPW *const cnt,
            double timestep )  [pure virtual]
```

Implemented in netdem::SolverBooleanPW, netdem::SolverGJKPW, netdem::SolverSDFPW, netdem::SolverSpherePlane, netdem::SolverSphereTriangle, netdem::SolverANNPPlane, and netdem::SolverANNPW.

7.18.3.8 UpdateBasicGeoms()

```
void netdem::CollisionSolverPW::UpdateBasicGeoms (
            CollisionGeometries *const cnt_geoms,
            double timestep,
            Vec3d const & dir_n_old )  [inline], [protected]
```

7.18.4 Member Data Documentation

7.18.4.1 particle

```
Particle* netdem::CollisionSolverPW::particle {nullptr}
```

7.18.4.2 wall

```
Wall* netdem::CollisionSolverPW::wall {nullptr}
```

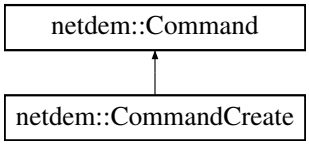
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/collision_solver_pw.hpp

7.19 netdem::Command Class Reference

```
#include <command.hpp>
```

Inheritance diagram for netdem::Command:



Public Member Functions

- Command (nlohmann::json const &info, Simulation *sim)
- virtual void Execute ()=0
- virtual ~Command ()

Public Attributes

- nlohmann::json info
- Simulation * sim

7.19.1 Detailed Description

an interface that all commands would inheritage. A command modifies a simulation according to the provided json information.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Command()

```
netdem::Command::Command (
            nlohmann::json const & info,
            Simulation * sim )  [inline]
```

7.19.2.2 ~Command()

```
virtual netdem::Command::~Command ( )  [inline], [virtual]
```

7.19.3 Member Function Documentation

7.19.3.1 Execute()

```
virtual void netdem::Command::Execute ( )  [pure virtual]
```

Implemented in netdem::CommandCreate.

7.19.4 Member Data Documentation

7.19.4.1 info

```
nlohmann::json netdem::Command::info
```

7.19.4.2 sim

```
Simulation* netdem::Command::sim
```

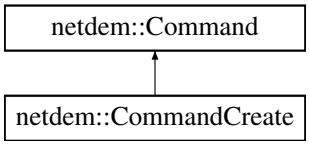
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/input/command.hpp

7.20 netdem::CommandCreate Class Reference

```
#include <command_create.hpp>
```

Inheritance diagram for netdem::CommandCreate:



Public Member Functions

- CommandCreate (nlohmann::json const &info, Simulation *sim)
- void Execute () override

Additional Inherited Members

7.20.1 Detailed Description

With the input parameters defined in json, create and insert a sphere shape into a simulation.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 CommandCreate()

```
netdem::CommandCreate::CommandCreate (
            nlohmann::json const & info,
            Simulation * sim )
```

7.20.3 Member Function Documentation

7.20.3.1 Execute()

```
void netdem::CommandCreate::Execute ( )  [override], [virtual]
```

Implements netdem::Command.

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/input/command_create.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/input/command_create.cpp

7.21 netdem::ContactForces Class Reference

```
#include <contact_forces.hpp>
```

Public Member Functions

- void Clear ()

Public Attributes

- double fc_n {0}
- double fc_s {0}
- double fc_t {0}
- double mc_n {0}
- double mc_s {0}
- double mc_t {0}
- double fd_n {0}
- double fd_s {0}
- double fd_t {0}
- double md_n {0}
- double md_s {0}
- double md_t {0}
- Vec3d force {0, 0, 0}
- Vec3d moment {0, 0, 0}
- Vec3d force_n {0, 0, 0}
- Vec3d force_t {0, 0, 0}
- Vec3d moment_n {0, 0, 0}
- Vec3d moment_t {0, 0, 0}

7.21.1 Detailed Description

- fc_n, fc_s, fc_t: contact forces in the contact normal and tangential directions, respectively.
- mc_n, mc_s, mc_t: contact moments in the contact normal and tangential directions, respectively.
- force, moment: contact forces and moments converted to the global coordinate system.

7.21.2 Member Function Documentation

7.21.2.1 Clear()

```
void netdem::ContactForces::Clear ( )  [inline]
```

7.21.3 Member Data Documentation

7.21.3.1 fc_n

```
double netdem::ContactForces::fc_n {0}
```

7.21.3.2 fc_s

```
double netdem::ContactForces::fc_s {0}
```

7.21.3.3 fc_t

```
double netdem::ContactForces::fc_t {0}
```

7.21.3.4 fd_n

```
double netdem::ContactForces::fd_n {0}
```

7.21.3.5 fd_s

```
double netdem::ContactForces::fd_s {0}
```

7.21.3.6 fd_t

```
double netdem::ContactForces::fd_t {0}
```

7.21.3.7 force

```
Vec3d netdem::ContactForces::force {0, 0, 0}
```

7.21.3.8 force_n

```
Vec3d netdem::ContactForces::force_n {0, 0, 0}
```

7.21.3.9 force_t

```
Vec3d netdem::ContactForces::force_t {0, 0, 0}
```

7.21.3.10 mc_n

```
double netdem::ContactForces::mc_n {0}
```

7.21.3.11 mc_s

```
double netdem::ContactForces::mc_s {0}
```

7.21.3.12 mc_t

```
double netdem::ContactForces::mc_t {0}
```

7.21.3.13 md_n

```
double netdem::ContactForces::md_n {0}
```

7.21.3.14 md_s

```
double netdem::ContactForces::md_s {0}
```

7.21.3.15 md_t

```
double netdem::ContactForces::md_t {0}
```

7.21.3.16 moment

```
Vec3d netdem::ContactForces::moment {0, 0, 0}
```

7.21.3.17 moment_n

```
Vec3d netdem::ContactForces::moment_n {0, 0, 0}
```

7.21.3.18 moment_t

```
Vec3d netdem::ContactForces::moment_t {0, 0, 0}
```

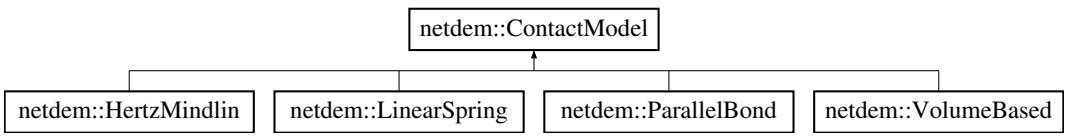
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/contact_forces.hpp

7.22 netdem::ContactModel Class Reference

```
#include <contact_model.hpp>
```

Inheritance diagram for netdem::ContactModel:



Public Types

- enum Type {
  none , linear_spring , hertz_mindlin , volume_based ,
  parallel_bond }

Public Member Functions

- virtual nlohmann::json PackJson ()
- virtual void InitFromJson (nlohmann::json const &js)
- virtual void SetProperty (nlohmann::json const &js)
- virtual ContactModel * Clone () const
- virtual void EvaluateForceMoment (ContactForces *const cnt_forces, CollisionGeometries &cnt_geoms, ContactPP *const cnt, double dt) const
- virtual void EvaluateForceMoment (ContactForces *const cnt_forces, BondGeometries &cnt_geoms, ContactPP *const cnt, double dt) const
- virtual void EvaluateForceMoment (ContactForces *const cnt_forces, CollisionGeometries &cnt_geoms, ContactPW *const cnt, double dt) const
- virtual void EvaluateForceMoment (ContactForces *const cnt_forces, BondGeometries &cnt_geoms, ContactPW *const cnt, double dt) const
- virtual void Print () const
- virtual ~ContactModel ()

Public Attributes

- int id {0}
- std::string label {"default"}
- int model_type {0}
- std::string model_name {"contact_model"}

7.22.1 Detailed Description

interface of contact model. A concrete contact model should have and override the EvaluateForceMoment method.

7.22.2 Member Enumeration Documentation

7.22.2.1 Type

`enum netdem::ContactModel::Type`

Enumerator

none	
linear_spring	
hertz_mindlin	
volume_based	
parallel_bond	

7.22.3 Constructor & Destructor Documentation

7.22.3.1 ~ContactModel()

```
virtual netdem::ContactModel::~ContactModel ( )  [inline], [virtual]
```

7.22.4 Member Function Documentation

7.22.4.1 Clone()

```
virtual ContactModel * netdem::ContactModel::Clone ( ) const  [inline], [virtual]
```

Reimplemented in netdem::HertzMindlin, netdem::LinearSpring, netdem::ParallelBond, and netdem::VolumeBased.

7.22.4.2 EvaluateForceMoment() [1/4]

```
virtual void netdem::ContactModel::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            BondGeometries & cnt_geoms,
            ContactPP *const cnt,
            double dt ) const  [inline], [virtual]
```

Reimplemented in netdem::ParallelBond.

7.22.4.3 EvaluateForceMoment() [2/4]

```
virtual void netdem::ContactModel::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            BondGeometries & cnt_geoms,
            ContactPW *const cnt,
            double dt ) const  [inline], [virtual]
```

Reimplemented in netdem::ParallelBond.

7.22.4.4 EvaluateForceMoment() [3/4]

```
virtual void netdem::ContactModel::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            CollisionGeometries & cnt_geoms,
            ContactPP *const cnt,
            double dt ) const  [inline], [virtual]
```

Reimplemented in netdem::HertzMindlin, netdem::LinearSpring, and netdem::VolumeBased.

7.22.4.5 EvaluateForceMoment() [4/4]

```
virtual void netdem::ContactModel::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            CollisionGeometries & cnt_geoms,
            ContactPW *const cnt,
            double dt ) const  [inline], [virtual]
```

Reimplemented in netdem::HertzMindlin, netdem::LinearSpring, and netdem::VolumeBased.

7.22.4.6 InitFromJson()

```
virtual void netdem::ContactModel::InitFromJson (
            nlohmann::json const & js )  [inline], [virtual]
```

Reimplemented in netdem::HertzMindlin, netdem::LinearSpring, netdem::ParallelBond, and netdem::VolumeBased.

7.22.4.7 PackJson()

```
virtual nlohmann::json netdem::ContactModel::PackJson ( )  [inline], [virtual]
```

Reimplemented in netdem::HertzMindlin, netdem::LinearSpring, netdem::ParallelBond, and netdem::VolumeBased.

7.22.4.8 Print()

```
virtual void netdem::ContactModel::Print ( ) const  [inline], [virtual]
```

Reimplemented in netdem::HertzMindlin, netdem::LinearSpring, netdem::ParallelBond, and netdem::VolumeBased.

7.22.4.9 SetProperty()

```
virtual void netdem::ContactModel::SetProperty (
            nlohmann::json const & js )  [inline], [virtual]
```

Reimplemented in netdem::HertzMindlin, netdem::LinearSpring, netdem::ParallelBond, and netdem::VolumeBased.

7.22.5 Member Data Documentation

7.22.5.1 id

```
int netdem::ContactModel::id {0}
```

7.22.5.2 label

```
std::string netdem::ContactModel::label {"default"}
```

7.22.5.3 model_name

```
std::string netdem::ContactModel::model_name {"contact_model"}
```

7.22.5.4 model_type

```
int netdem::ContactModel::model_type {0}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/contact_model.hpp

7.23 netdem::ContactModelFactory Class Reference

```
#include <contact_model_factory.hpp>
```

Static Public Member Functions

- static ContactModel * NewContactModel (std::string const &model_name, nlohmann::json const &js)

Static Public Attributes

- static std::unordered_map< std::string, ContactModel::Type > model_map

7.23.1 Member Function Documentation

7.23.1.1 NewContactModel()

```
ContactModel * ContactModelFactory::NewContactModel (
            std::string const & model_name,
            nlohmann::json const & js )  [static]
```

7.23.2 Member Data Documentation

7.23.2.1 model_map

```
unordered_map< string, ContactModel::Type > ContactModelFactory::model_map  [static]
```

Initial value:

```
= {
    {"linear_spring", ContactModel::Type::linear_spring},
    {"hertz_mindlin", ContactModel::Type::hertz_mindlin},
    {"volume_based", ContactModel::Type::volume_based},
    {"parallel_bond", ContactModel::Type::parallel_bond}}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/contact_model_factory.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/contact_model_factory.cpp

7.24 netdem::ContactPP Class Reference

```
#include <contact_pp.hpp>
```

Public Member Functions

- ContactPP ()
- ContactPP (Particle *const p1, Particle *const p2)
- void Init (Particle *const p1, Particle *const p2)
- void SetBondModel (ContactModel *const cnt_model)
- void SetCollisionModel (ContactModel *const cnt_model)
- void EvaluateForceMoment (double dt)
- void ApplyToParticle ()
- void ApplyToParticle1 ()
- void ApplyToParticle2 ()
- bool IsActive ()
- void Clear ()
- void Print ()

Public Attributes

- Particle * particle_1 {nullptr}
- Particle * particle_2 {nullptr}
- ContactModel * bond_model {nullptr}
- ContactModel * collision_model {nullptr}
- VecXT< BondEntry > bond_entries
- VecXT< CollisionEntry > collision_entries
- bool active {true}
- MiniMap< std::string, double > dynamic_properties

7.24.1 Detailed Description

to update contact between particles:

- particle_1, particle_2: pointers of particle 1 and 2, respectively.
- contact_model: contact model for this contact.
- collision_geometries: contact geometries, which will be used by the contact model to calculate contact forces.
- updated: a flag indicates that whether this contact has been updated in one DEM cycle. At the begining of a DEM cycle, this flag is set to false. If two particles (or a particle and a wall) are still in contact, the existing contact object will be udpated, and this flag will be set to true. If contact is not updated, it means that this contact is not activated (i.e., this contact vanishes) any more, and thus will be deleted.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 ContactPP() [1/2]

```
ContactPP::ContactPP ( )
```

7.24.2.2 ContactPP() [2/2]

```
ContactPP::ContactPP (
            Particle *const p1,
            Particle *const p2 )
```

7.24.3 Member Function Documentation

7.24.3.1 ApplyToParticle()

```
void ContactPP::ApplyToParticle ( )
```

7.24.3.2 ApplyToParticle1()

```
void ContactPP::ApplyToParticle1 ( )
```

7.24.3.3 ApplyToParticle2()

```
void ContactPP::ApplyToParticle2 ( )
```

7.24.3.4 Clear()

```
void ContactPP::Clear ( )
```

7.24.3.5 EvaluateForceMoment()

```
void ContactPP::EvaluateForceMoment (
            double dt )
```

7.24.3.6 Init()

```
void ContactPP::Init (
            Particle *const p1,
            Particle *const p2 )
```

7.24.3.7 IsActive()

```
bool ContactPP::IsActive ( )
```

7.24.3.8 Print()

```
void ContactPP::Print ( )
```

7.24.3.9 SetBondModel()

```
void ContactPP::SetBondModel (
            ContactModel *const cnt_model )
```

7.24.3.10 SetCollisionModel()

```
void ContactPP::SetCollisionModel (
            ContactModel *const cnt_model )
```

7.24.4 Member Data Documentation

7.24.4.1 active

```
bool netdem::ContactPP::active {true}
```

7.24.4.2 bond_entries

```
VecXT<BondEntry> netdem::ContactPP::bond_entries
```

7.24.4.3 bond_model

```
ContactModel* netdem::ContactPP::bond_model {nullptr}
```

7.24.4.4 collision_entries

```
VecXT<CollisionEntry> netdem::ContactPP::collision_entries
```

7.24.4.5 collision_model

```
ContactModel * netdem::ContactPP::collision_model {nullptr}
```

7.24.4.6 dynamic_properties

```
MiniMap<std::string, double> netdem::ContactPP::dynamic_properties
```

7.24.4.7 particle_1

```
Particle* netdem::ContactPP::particle_1 {nullptr}
```

7.24.4.8 particle_2

```
Particle * netdem::ContactPP::particle_2 {nullptr}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/contact_pp.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/contact_pp.cpp

7.25 netdem::ContactPPData Struct Reference

```
#include <contact_pp_data.hpp>
```

Public Attributes

- int particle_1_id {0}
- int particle_2_id {0}
- int bond_model_id {-1}
- int collision_model_id {-1}
- int num_bond_entries {0}
- int num_collision_entries {0}

7.25.1 Detailed Description

defines the particle-particle contact data for MPI.

7.25.2 Member Data Documentation

7.25.2.1 bond_model_id

```
int netdem::ContactPPData::bond_model_id {-1}
```

7.25.2.2 collision_model_id

```
int netdem::ContactPPData::collision_model_id {-1}
```

7.25.2.3 num_bond_entries

```
int netdem::ContactPPData::num_bond_entries {0}
```

7.25.2.4 num_collision_entries

```
int netdem::ContactPPData::num_collision_entries {0}
```

7.25.2.5 particle_1_id

```
int netdem::ContactPPData::particle_1_id {0}
```

7.25.2.6 particle_2_id

```
int netdem::ContactPPData::particle_2_id {0}
```

The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pp_data.hpp

7.26 netdem::ContactPPParser Class Reference

```
#include <contact_pp_parser.hpp>
```

Static Public Member Functions

- static void ClassToStruct (const ContactPP *const cnt_class, ContactPPData *const cnt_struct)
- static void StructToClass (ContactPP *const cnt_class, const ContactPPData *const cnt_struct, const BondEntryData *const bond_entries_data, const CollisionEntryData *const collision_entries_data, const MiniMap< int, ContactModel * > &contact_model_map)
- static void DefineMPIDataType (MPI_Datatype *const datatype)

7.26.1 Detailed Description

convert particle class from/to particle data struct

7.26.2 Member Function Documentation

7.26.2.1 ClassToStruct()

```
void ContactPPParser::ClassToStruct (
            const ContactPP *const cnt_class,
            ContactPPData *const cnt_struct )  [static]
```

7.26.2.2 DefineMPIDataType()

```
void ContactPPParser::DefineMPIDataType (
            MPI_Datatype *const datatype )  [static]
```

7.26.2.3 StructToClass()

```
void ContactPPParser::StructToClass (
            ContactPP *const cnt_class,
            const ContactPPData *const cnt_struct,
            const BondEntryData *const bond_entries_data,
            const CollisionEntryData *const collision_entries_data,
            const MiniMap< int, ContactModel * > & contact_model_map )  [static]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pp_parser.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pp_parser.cpp

7.27 netdem::ContactPW Class Reference

`#include <contact_pw.hpp>`

Public Member Functions

- ContactPW ()
- ContactPW (Particle *const p, Wall *const w)
- void Init (Particle *const p, Wall *const w)
- void SetBondModel (ContactModel *const cnt_model)
- void SetCollisionModel (ContactModel *const cnt_model)
- void EvaluateForceMoment (double dt)
- void ApplyToParticle ()
- void ApplyToWall ()
- bool IsActive ()
- void Clear ()
- void Print ()

Public Attributes

- Particle * particle {nullptr}
- Wall * wall {nullptr}
- ContactModel * bond_model {nullptr}
- ContactModel * collision_model {nullptr}
- VecXT< BondEntry > bond_entries
- VecXT< CollisionEntry > collision_entries
- bool active {true}
- MiniMap< std::string, double > dynamic_properties

7.27.1 Detailed Description

to update contact between particle and wall:

- particle, wall: pointers of particle and wall, respectively.
- contact_model: contact model for this contact.
- collision_geometries: contact geometries, which will be used by the contact model to calculate contact forces.
- updated: a flag indicates that whether this contact has been updated in one DEM cycle. At the begining of a DEM cycle, this flag is set to false. If two particles (or a particle and a wall) are still in contact, the existing contact object will be udpated, and this flag will be set to true. If contact is not updated, it means that this contact is not activated (i.e., this contact vanishes) any more, and thus will be deleted.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 ContactPW() [1/2]

```
ContactPW::ContactPW ( )
```

7.27.2.2 ContactPW() [2/2]

```
ContactPW::ContactPW (
            Particle *const p,
            Wall *const w )
```

7.27.3 Member Function Documentation

7.27.3.1 ApplyToParticle()

```
void ContactPW::ApplyToParticle ( )
```

7.27.3.2 ApplyToWall()

```
void ContactPW::ApplyToWall ( )
```

7.27.3.3 Clear()

```
void ContactPW::Clear ( )
```

7.27.3.4 EvaluateForceMoment()

```
void ContactPW::EvaluateForceMoment (
            double dt )
```

7.27.3.5 Init()

```
void ContactPW::Init (
            Particle *const p,
            Wall *const w )
```

7.27.3.6 IsActive()

```
bool ContactPW::IsActive ( )
```

7.27.3.7 Print()

```
void ContactPW::Print ( )
```

7.27.3.8 SetBondModel()

```
void ContactPW::SetBondModel (
            ContactModel *const cnt_model )
```

7.27.3.9 SetCollisionModel()

```
void ContactPW::SetCollisionModel (
            ContactModel *const cnt_model )
```

7.27.4 Member Data Documentation

7.27.4.1 active

```
bool netdem::ContactPW::active {true}
```

7.27.4.2 bond_entries

```
VecXT<BondEntry> netdem::ContactPW::bond_entries
```

7.27.4.3 bond_model

```
ContactModel* netdem::ContactPW::bond_model {nullptr}
```

7.27.4.4 collision_entries

```
VecXT<CollisionEntry> netdem::ContactPW::collision_entries
```

7.27.4.5 collision_model

```
ContactModel * netdem::ContactPW::collision_model {nullptr}
```

7.27.4.6 dynamic_properties

```
MiniMap<std::string, double> netdem::ContactPW::dynamic_properties
```

7.27.4.7 particle

```
Particle* netdem::ContactPW::particle {nullptr}
```

7.27.4.8 wall

```
Wall* netdem::ContactPW::wall {nullptr}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/contact_pw.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/contact_pw.cpp

7.28 netdem::ContactPWData Struct Reference

```
#include <contact_pw_data.hpp>
```

Public Attributes

- int particle_id {0}
- int wall_id {0}
- int bond_model_id {-1}
- int collision_model_id {-1}
- int num_bond_entries {0}
- int num_collision_entries {0}

7.28.1 Detailed Description

defines the particle-wall contact data for MPI.

7.28.2 Member Data Documentation

7.28.2.1 bond_model_id

```
int netdem::ContactPWData::bond_model_id {-1}
```

7.28.2.2 collision_model_id

```
int netdem::ContactPWData::collision_model_id {-1}
```

7.28.2.3 num_bond_entries

```
int netdem::ContactPWData::num_bond_entries {0}
```

7.28.2.4 num_collision_entries

```
int netdem::ContactPWData::num_collision_entries {0}
```

7.28.2.5 particle_id

```
int netdem::ContactPWData::particle_id {0}
```

7.28.2.6 wall_id

```
int netdem::ContactPWData::wall_id {0}
```

The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pw_data.hpp

7.29 netdem::ContactPWParser Class Reference

`#include <contact_pw_parser.hpp>`

Static Public Member Functions

- static void ClassToStruct (const ContactPW *const cnt_class, ContactPWData *const cnt_struct)
- static void StructToClass (ContactPW *const cnt_class, const ContactPWData *const cnt_struct, const BondEntryData *const bond_entries_data, const CollisionEntryData *const collision_entries_data, const MiniMap< int, ContactModel * > &contact_model_map)
- static void DefineMPIDataType (MPI_Datatype *const datatype)

7.29.1 Detailed Description

convert particle class from/to particle data struct

7.29.2 Member Function Documentation

7.29.2.1 ClassToStruct()

```
void ContactPWParser::ClassToStruct (
            const ContactPW *const cnt_class,
            ContactPWData *const cnt_struct )  [static]
```

7.29.2.2 DefineMPIDataType()

```
void ContactPWParser::DefineMPIDataType (
            MPI_Datatype *const datatype )  [static]
```

7.29.2.3 StructToClass()

```
void ContactPWParser::StructToClass (
            ContactPW *const cnt_class,
            const ContactPWData *const cnt_struct,
            const BondEntryData *const bond_entries_data,
            const CollisionEntryData *const collision_entries_data,
            const MiniMap< int, ContactModel * > & contact_model_map )  [static]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pw_parser.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pw_parser.cpp

7.30 netdem::ContactSolverFactory Class Reference

`#include <contact_solver_factory.hpp>`

Public Member Functions

- ContactSolverFactory ()
- ContactSolverFactory (ContactSolverFactory &tmp_factory)
- ContactSolverFactory (ContactSolverFactory &&tmp_factory)
- ContactSolverFactory & operator= (ContactSolverFactory &tmp_factory)
- ContactSolverFactory & operator= (ContactSolverFactory &&tmp_factory)
- BondSolverPP * GetBondSolver (Particle *const p1, Particle *const p2)
- BondSolverPW * GetBondSolver (Particle *const p, Wall *const w)
- CollisionSolverPP * GetCollisionSolver (Particle *const p1, Particle *const p2)
- CollisionSolverPW * GetCollisionSolver (Particle *const p, Wall *const w)
- int InsertSolver (CollisionSolverPP *const cnt_solver)
- int InsertSolver (CollisionSolverPW *const cnt_solver)
- void CustomizeSolverPP (int shape_id1, int shape_id2, int solver_id)
- void CustomizeSolverPW (int shape_id1, int shape_id2, int solver_id)
- ~ContactSolverFactory ()

Public Attributes

- ContactSolverSettings settings
- BondSolverPP bond_solver_pp
- BondSolverPW bond_solver_pw
- VecNT< VecNT< int, Shape::Type::num_shapes >, Shape::Type::num_shapes > solver_id_pp_list
- VecNT< VecNT< int, Shape::Type::num_shapes >, Shape::Type::num_shapes > solver_id_pw_list
- std::unordered_map< std::pair< int, int >, int, pair_hash > solver_id_customized
- VecXT< CollisionSolverPP * > solver_pp_pool
- VecXT< CollisionSolverPW * > solver_pw_pool

Private Member Functions

- CollisionSolverPP * NewCollisionSolver (Particle *const p1, Particle *const p2)
- CollisionSolverPW * NewCollisionSolver (Particle *const p, Wall *const w)

7.30.1 Detailed Description

ContactSolverFactory behaviors as a factory to get solvers for contact detection and resolution.

- solver_pp_pool: stores the solvers for different pairs of shapes. When contact needs to be checked for two particles, solvers is gotten from the solver_pp_pool based on the shapes of the two particles.
- solver_pw_pool: similar to the solver_pp_pool, whereas it is for particle and wall contacts.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 ContactSolverFactory() [1/3]

```
ContactSolverFactory::ContactSolverFactory ( )
```

7.30.2.2 ContactSolverFactory() [2/3]

```
ContactSolverFactory::ContactSolverFactory (
            ContactSolverFactory & tmp_factory )
```

7.30.2.3 ContactSolverFactory() [3/3]

```
ContactSolverFactory::ContactSolverFactory (
            ContactSolverFactory && tmp_factory )
```

7.30.2.4 ~ContactSolverFactory()

```
ContactSolverFactory::~ContactSolverFactory ( )
```

7.30.3 Member Function Documentation

7.30.3.1 CustomizeSolverPP()

```
void ContactSolverFactory::CustomizeSolverPP (
            int shape_id1,
            int shape_id2,
            int solver_id )
```

7.30.3.2 CustomizeSolverPW()

```
void ContactSolverFactory::CustomizeSolverPW (
            int shape_id1,
            int shape_id2,
            int solver_id )
```

7.30.3.3 GetBondSolver() [1/2]

```
BondSolverPW * ContactSolverFactory::GetBondSolver (
            Particle *const p,
            Wall *const w )
```

7.30.3.4 GetBondSolver() [2/2]

```
BondSolverPP * ContactSolverFactory::GetBondSolver (
            Particle *const p1,
            Particle *const p2 )
```

7.30.3.5 GetCollisionSolver() [1/2]

```
CollisionSolverPW * ContactSolverFactory::GetCollisionSolver (
            Particle *const p,
            Wall *const w )
```

7.30.3.6 GetCollisionSolver() [2/2]

```
CollisionSolverPP * ContactSolverFactory::GetCollisionSolver (
            Particle *const p1,
            Particle *const p2 )
```

7.30.3.7 InsertSolver() [1/2]

```
int ContactSolverFactory::InsertSolver (
            CollisionSolverPP *const cnt_solver )
```

7.30.3.8 InsertSolver() [2/2]

```
int ContactSolverFactory::InsertSolver (
            CollisionSolverPW *const cnt_solver )
```

7.30.3.9 NewCollisionSolver() [1/2]

```
CollisionSolverPW * ContactSolverFactory::NewCollisionSolver (
            Particle *const p,
            Wall *const w )  [private]
```

7.30.3.10 NewCollisionSolver() [2/2]

```
CollisionSolverPP * ContactSolverFactory::NewCollisionSolver (
            Particle *const p1,
            Particle *const p2 )  [private]
```

7.30.3.11 operator=() [1/2]

```
ContactSolverFactory & ContactSolverFactory::operator= (
            ContactSolverFactory && tmp_factory )
```

7.30.3.12 operator=() [2/2]

```
ContactSolverFactory & ContactSolverFactory::operator= (
            ContactSolverFactory & tmp_factory )
```

7.30.4 Member Data Documentation

7.30.4.1 bond_solver_pp

```
BondSolverPP netdem::ContactSolverFactory::bond_solver_pp
```

7.30.4.2 bond_solver_pw

```
BondSolverPW netdem::ContactSolverFactory::bond_solver_pw
```

7.30.4.3 settings

```
ContactSolverSettings netdem::ContactSolverFactory::settings
```

7.30.4.4 solver_id_customized

```
std::unordered_map<std::pair<int, int>, int, pair_hash> netdem::ContactSolverFactory::solver↩
_id_customized
```

7.30.4.5 solver_id_pp_list

```
VecNT<VecNT<int, Shape::Type::num_shapes>, Shape::Type::num_shapes> netdem::ContactSolver↩
Factory::solver_id_pp_list
```

lookup table for finding the id of a solver (solvers are stored in solver_pp_pool)

7.30.4.6 solver_id_pw_list

```
VecNT<VecNT<int, Shape::Type::num_shapes>, Shape::Type::num_shapes> netdem::ContactSolver↩
Factory::solver_id_pw_list
```

7.30.4.7 solver_pp_pool

```
VecXT<CollisionSolverPP *> netdem::ContactSolverFactory::solver_pp_pool
```

7.30.4.8 solver_pw_pool

```
VecXT<CollisionSolverPW *> netdem::ContactSolverFactory::solver_pw_pool
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/contact_solver_factory.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/contact_solver_factory.cpp

7.31 netdem::ContactSolverSettings Class Reference

```
#include <contact_solver_factory.hpp>
```

Public Types

- enum SolverType { gjk , sdf , automatic }

Public Attributes

- SolverType solver_type {SolverType::automatic}
- bool gjk_use_erosion {false}
- double gjk_erosion_ratio_initial {0.01}
- double gjk_erosion_ratio_increment {0.01}
- int sdf_potential_type {0}
- bool sdf_solve_two_sides {false}

7.31.1 Member Enumeration Documentation

7.31.1.1 SolverType

```
enum netdem::ContactSolverSettings::SolverType
```

Enumerator

gjk	
sdf	
automatic	

7.31.2 Member Data Documentation

7.31.2.1 gjk_erosion_ratio_increment

```
double netdem::ContactSolverSettings::gjk_erosion_ratio_increment {0.01}
```

7.31.2.2 gjk_erosion_ratio_initial

```
double netdem::ContactSolverSettings::gjk_erosion_ratio_initial {0.01}
```

7.31.2.3 gjk_use_erosion

```
bool netdem::ContactSolverSettings::gjk_use_erosion {false}
```

7.31.2.4 sdf_potential_type

```
int netdem::ContactSolverSettings::sdf_potential_type {0}
```

7.31.2.5 sdf_solve_two_sides

```
bool netdem::ContactSolverSettings::sdf_solve_two_sides {false}
```

7.31.2.6 solver_type

```
SolverType netdem::ContactSolverSettings::solver_type {SolverType::automatic}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/contact_solver_factory.hpp

7.32 netdem::Cork Class Reference

```
#include <cork_wrapper.hpp>
```

Static Public Member Functions

- static void MeshIntersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- static void MeshUnion (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- static void MeshDifference (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- static void MeshXor (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- static void MeshIntersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- static void MeshUnion (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- static void MeshDifference (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- static void MeshXor (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- static void MeshIntersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, double dist_pc_to_plane, Vec3d const &dir_n, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)

7.32.1 Member Function Documentation

7.32.1.1 MeshDifference() [1/2]

```
static void netdem::Cork::MeshDifference (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab )  [static]
```

7.32.1.2 MeshDifference() [2/2]

```
static void netdem::Cork::MeshDifference (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab,
            VecXT< int > *const jab )  [static]
```

7.32.1.3 MeshIntersect() [1/3]

```
static void netdem::Cork::MeshIntersect (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab )  [static]
```

7.32.1.4 MeshIntersect() [2/3]

```
static void netdem::Cork::MeshIntersect (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab,
            VecXT< int > *const jab )  [static]
```

7.32.1.5 MeshIntersect() [3/3]

```
static void netdem::Cork::MeshIntersect (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            double dist_pc_to_plane,
            Vec3d const & dir_n,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab,
            VecXT< int > *const jab )  [static]
```

7.32.1.6 MeshUnion() [1/2]

```
static void netdem::Cork::MeshUnion (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab )  [static]
```

7.32.1.7 MeshUnion() [2/2]

```
static void netdem::Cork::MeshUnion (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab,
            VecXT< int > *const jab )  [static]
```

7.32.1.8 MeshXor() [1/2]

```
static void netdem::Cork::MeshXor (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab )  [static]
```

7.32.1.9 MeshXor() [2/2]

```
static void netdem::Cork::MeshXor (
            const VecXT< Vec3d > & va,
            const VecXT< Vec3i > & fa,
            const VecXT< Vec3d > & vb,
            const VecXT< Vec3i > & fb,
            VecXT< Vec3d > *const vab,
            VecXT< Vec3i > *const fab,
            VecXT< int > *const jab )  [static]
```

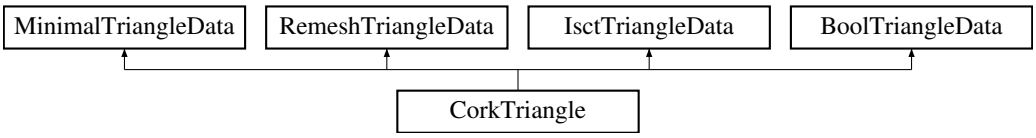
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/cork_wrapper.hpp

7.33 CorkTriangle Struct Reference

```
#include <cork_decls.hpp>
```

Inheritance diagram for CorkTriangle:



Public Member Functions

- void merge (const CorkTriangle &, const CorkTriangle &)
- void move (const CorkTriangle &)
- void subdivide (SubdivideTriInput< CorkVertex, CorkTriangle > input)

Static Public Member Functions

- static void split (CorkTriangle &, CorkTriangle &, const CorkTriangle &)

7.33.1 Member Function Documentation

7.33.1.1 merge()

```
void CorkTriangle::merge (
            const CorkTriangle & ,
            const CorkTriangle &  )  [inline]
```

7.33.1.2 move()

```
void CorkTriangle::move (
            const CorkTriangle &  )  [inline]
```

7.33.1.3 split()

```
static void CorkTriangle::split (
            CorkTriangle & ,
            CorkTriangle & ,
            const CorkTriangle &  )  [inline], [static]
```

7.33.1.4 subdivide()

```
void CorkTriangle::subdivide (
            SubdivideTriInput< CorkVertex, CorkTriangle > input )  [inline]
```

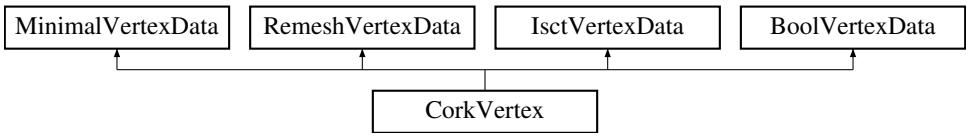
The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/cork_decls.hpp

7.34 CorkVertex Struct Reference

`#include <cork_decls.hpp>`

Inheritance diagram for CorkVertex:



Public Member Functions

- void merge (const CorkVertex &v0, const CorkVertex &v1)
- void interpolate (const CorkVertex &v0, const CorkVertex &v1)
- void isct (IsctVertEdgeTriInput< CorkVertex, CorkTriangle > input)
- void isct (IsctVertTriTriTriInput< CorkVertex, CorkTriangle > input)
- void isctInterpolate (const CorkVertex &v0, const CorkVertex &v1)

7.34.1 Member Function Documentation

7.34.1.1 interpolate()

```
void CorkVertex::interpolate (
            const CorkVertex & v0,
            const CorkVertex & v1 )  [inline]
```

7.34.1.2 isct() [1/2]

```
void CorkVertex::isct (
            IsctVertEdgeTriInput< CorkVertex, CorkTriangle > input )  [inline]
```

7.34.1.3 isct() [2/2]

```
void CorkVertex::isct (
            IsctVertTriTriTriInput< CorkVertex, CorkTriangle > input )  [inline]
```

7.34.1.4 isctInterpolate()

```
void CorkVertex::isctInterpolate (
            const CorkVertex & v0,
            const CorkVertex & v1 )  [inline]
```

7.34.1.5 merge()

```
void CorkVertex::merge (
            const CorkVertex & v0,
            const CorkVertex & v1 )  [inline]
```

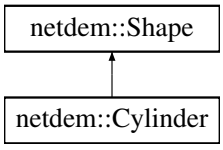
The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/cork_decls.hpp

7.35 netdem::Cylinder Class Reference

`#include <shape_cylinder.hpp>`

Inheritance diagram for netdem::Cylinder:



Public Member Functions

- Cylinder ()
- Cylinder (double r, double h)
- Shape * Clone () const override
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void Init ()
- void SetSize (double d) override
- void UpdateNodes () override
- void UpdateShapeProperties () override
- STLModel GetSTLModel (int num_facets=400) override
- Vec3d SupportPoint (Vec3d const &dir) override
- VecXT< Vec3d > SupportPoints (Vec3d const &dir) override
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- double CalculateRho (Vec3d const &dir)
- void Print () override

Public Attributes

- double radius {0.5}
- double height {1.0}

Additional Inherited Members

7.35.1 Constructor & Destructor Documentation

7.35.1.1 Cylinder() [1/2]

```
Cylinder::Cylinder ( )
```

7.35.1.2 Cylinder() [2/2]

```
Cylinder::Cylinder (
            double r,
            double h )
```

7.35.2 Member Function Documentation

7.35.2.1 CalculateRho()

```
double Cylinder::CalculateRho (
            Vec3d const & dir )
```

7.35.2.2 Clone()

```
Shape * Cylinder::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.3 GetSTLModel()

```
STLModel Cylinder::GetSTLModel (
            int num_facets = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.4 Init()

```
void Cylinder::Init ( )
```

7.35.2.5 InitFromJson()

```
void Cylinder::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.6 PackJson()

```
nlohmann::json Cylinder::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.7 Print()

```
void Cylinder::Print ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.8 SetSize()

```
void Cylinder::SetSize (
            double d )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.9 SignedDistance()

```
double Cylinder::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.10 SupportPoint()

```
Vec3d Cylinder::SupportPoint (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.11 SupportPoints()

```
VecXT< Vec3d > Cylinder::SupportPoints (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.12 SurfacePoint()

```
Vec3d Cylinder::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.35.2.13 UpdateNodes()

```
void Cylinder::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.2.14 UpdateShapeProperties()

```
void Cylinder::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.35.3 Member Data Documentation

7.35.3.1 height

```
double netdem::Cylinder::height {1.0}
```

7.35.3.2 radius

```
double netdem::Cylinder::radius {0.5}
```

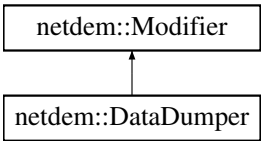
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_cylinder.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_cylinder.cpp

7.36 netdem::DataDumper Class Reference

```
#include <data_dumper.hpp>
```

Inheritance diagram for netdem::DataDumper:



Public Member Functions

- DataDumper ()
- void Init (Simulation *sim) override
- void SetRootPath (std::string const &root_path)
- void SetDataType (std::string const &data_type)
- void SetFrequency (bool save_by_cycles, double interval)
- void SaveParticleInfoAsVTK ()
- void SaveParticleInfoAsDump ()
- void SaveParticleInfoAsVTKWithProxy ()
- void SaveWallInfoAsVTK ()
- void SaveWallInfoAsDump ()
- void SaveCollisionInfoAsVTK ()
- void SaveCollisionInfoAsDump ()
- void SaveBondInfoAsVTK ()
- void SaveBondInfoAsDump ()
- void SaveShapeInfoAsSTL ()
- void SaveShapeInfoAsVTK ()
- void SaveShapeInfoAsJson (bool all_in_one=false)
- Modifier * Clone () const override
- void Execute () override

Public Attributes

- bool dump_particle_info {true}
- bool dump_wall_info {false}
- bool dump_contact_info {false}
- bool dump_shape_info {false}

Private Member Functions

- void GetCollisionContacts (VecXT< ContactPP * > *const cnt_pp_list, VecXT< ContactPW * > *const cnt_pw_list)
- void GetBondContacts (VecXT< ContactPP * > *const cnt_pp_list, VecXT< ContactPW * > *const cnt_↩ pw_list)
- std::string GetParticleInfoFilename ()
- std::string GetWallInfoFilename ()
- std::string GetCollisionInfoFilename ()
- std::string GetBondInfoFilename ()
- std::string GetShapeInfoFilename ()
- bool CheckIfToSave ()

Private Attributes

- std::string root_path {"tmp/out/"}
- std::string data_type {"vtk"}
- bool save_by_cycles {true}
- int cycle_interval {0}
- int cycle_previous {0}
- double time_interval {0}
- double time_previous {0}

7.36.1 Detailed Description

To dump particle data into vtk files. This is a post-modifier, which will be excuted at the end of a DEM cycle.

- file_dir: the directory of the output file
- file_basename: the name prefix of the outpuf file
- data_type: currently, vtk or dump (which is used in LAMMPS) format is implemented
- save_by_cycles: data is saved in every [cycle_interval] cycles if this flag is true. Otherwise, data is saved in every [time_interval] mechanical time (i.e., time of the simulated world).
- cycle_previous: the id of the cycle in which the data is saved. If the difference between cycle_previsous and current cycle id is greater than cycle_interval. Data would be saved.
- time_previous: function similarly to cycle_previous.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 DataDumper()

```
netdem::DataDumper::DataDumper ( )
```

7.36.3 Member Function Documentation

7.36.3.1 CheckIfToSave()

```
bool netdem::DataDumper::CheckIfToSave ( )  [inline], [private]
```

7.36.3.2 Clone()

```
Modifier * netdem::DataDumper::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.36.3.3 Execute()

```
void netdem::DataDumper::Execute ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.36.3.4 GetBondContacts()

```
void netdem::DataDumper::GetBondContacts (
            VecXT< ContactPP * > *const cnt_pp_list,
            VecXT< ContactPW * > *const cnt_pw_list )  [private]
```

7.36.3.5 GetBondInfoFilename()

```
string netdem::DataDumper::GetBondInfoFilename ( )  [inline], [private]
```

7.36.3.6 GetCollisionContacts()

```
void netdem::DataDumper::GetCollisionContacts (
            VecXT< ContactPP * > *const cnt_pp_list,
            VecXT< ContactPW * > *const cnt_pw_list )  [private]
```

7.36.3.7 GetCollisionInfoFilename()

```
string netdem::DataDumper::GetCollisionInfoFilename ( )  [inline], [private]
```

7.36.3.8 GetParticleInfoFilename()

```
string netdem::DataDumper::GetParticleInfoFilename ( )  [inline], [private]
```

7.36.3.9 GetShapeInfoFilename()

```
string netdem::DataDumper::GetShapeInfoFilename ( )  [inline], [private]
```

7.36.3.10 GetWallInfoFilename()

```
string netdem::DataDumper::GetWallInfoFilename ( )  [inline], [private]
```

7.36.3.11 Init()

```
void netdem::DataDumper::Init (
            Simulation * sim )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.36.3.12 SaveBondInfoAsDump()

```
void netdem::DataDumper::SaveBondInfoAsDump ( )
```

7.36.3.13 SaveBondInfoAsVTK()

```
void netdem::DataDumper::SaveBondInfoAsVTK ( )
```

7.36.3.14 SaveCollisionInfoAsDump()

```
void netdem::DataDumper::SaveCollisionInfoAsDump ( )
```

7.36.3.15 SaveCollisionInfoAsVTK()

```
void netdem::DataDumper::SaveCollisionInfoAsVTK ( )
```

7.36.3.16 SaveParticleInfoAsDump()

```
void netdem::DataDumper::SaveParticleInfoAsDump ( )
```

7.36.3.17 SaveParticleInfoAsVTK()

```
void netdem::DataDumper::SaveParticleInfoAsVTK ( )
```

7.36.3.18 SaveParticleInfoAsVTKWithProxy()

```
void netdem::DataDumper::SaveParticleInfoAsVTKWithProxy ( )
```

7.36.3.19 SaveShapeInfoAsJson()

```
void netdem::DataDumper::SaveShapeInfoAsJson (
            bool all_in_one = false )
```

7.36.3.20 SaveShapeInfoAsSTL()

```
void netdem::DataDumper::SaveShapeInfoAsSTL ( )
```

7.36.3.21 SaveShapeInfoAsVTK()

```
void netdem::DataDumper::SaveShapeInfoAsVTK ( )
```

7.36.3.22 SaveWallInfoAsDump()

```
void netdem::DataDumper::SaveWallInfoAsDump ( )
```

7.36.3.23 SaveWallInfoAsVTK()

```
void netdem::DataDumper::SaveWallInfoAsVTK ( )
```

7.36.3.24 SetDataType()

```
void netdem::DataDumper::SetDataType (
            std::string const & data_type )
```

7.36.3.25 SetFrequency()

```
void netdem::DataDumper::SetFrequency (
            bool save_by_cycles,
            double interval )
```

7.36.3.26 SetRootPath()

```
void netdem::DataDumper::SetRootPath (
            std::string const & root_path )
```

7.36.4 Member Data Documentation

7.36.4.1 cycle_interval

```
int netdem::DataDumper::cycle_interval {0}  [private]
```

7.36.4.2 cycle_previous

```
int netdem::DataDumper::cycle_previous {0}  [private]
```

7.36.4.3 data_type

```
std::string netdem::DataDumper::data_type {"vtk"}  [private]
```

7.36.4.4 dump_contact_info

```
bool netdem::DataDumper::dump_contact_info {false}
```

7.36.4.5 dump_particle_info

```
bool netdem::DataDumper::dump_particle_info {true}
```

7.36.4.6 dump_shape_info

```
bool netdem::DataDumper::dump_shape_info {false}
```

7.36.4.7 dump_wall_info

```
bool netdem::DataDumper::dump_wall_info {false}
```

7.36.4.8 root_path

```
std::string netdem::DataDumper::root_path {"tmp/out/"}  [private]
```

7.36.4.9 save_by_cycles

```
bool netdem::DataDumper::save_by_cycles {true}  [private]
```

7.36.4.10 time_interval

```
double netdem::DataDumper::time_interval {0}  [private]
```

7.36.4.11 time_previous

```
double netdem::DataDumper::time_previous {0}  [private]
```

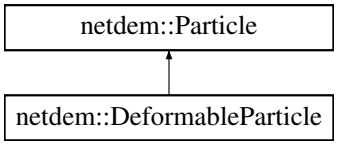
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/data_dumper.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/data_dumper.cpp

7.37 netdem::DeformableParticle Class Reference

```
#include <deformable_particle.hpp>
```

Inheritance diagram for netdem::DeformableParticle:



Public Member Functions

- DeformableParticle ()
- Particle * Clone () const override
- void SetShape (Shape *s) override
- void SetDensity (double dens) override
- void SetPosition (double x, double y, double z) override
- void SetRodrigues (double angle, double axis_x, double axis_y, double axis_z) override
- void SetQuaternion (double q_0, double q_1, double q_2, double q_3) override
- void SetVelocity (double v_x, double v_y, double v_z) override
- Vec3d GetVelocity (Vec3d const &pos) override
- void AddForce (int node_id, Vec3d const &f)
- void ClearForce () override
- void ApplyContactForce (ContactPP const *cnt) override
- void ApplyContactForce (ContactPW const *cnt) override
- void UpdateMotion (double dt) override
- void UpdateShape ()
- void UpdateBound () override
- void SaveSurfaceAsVTK (std::string const &filename)
- void SaveAsVTK (std::string const &filename) override
- ~DeformableParticle () override

Public Attributes

- TriMesh * trimesh {nullptr}
- TetMesh tetmesh
- FEMSimulator fem_simulator
- int mesh_res {20}

Private Member Functions

- void InitFEMSimulator ()

7.37.1 Constructor & Destructor Documentation

7.37.1.1 DeformableParticle()

```
netdem::DeformableParticle::DeformableParticle ( )
```

7.37.1.2 ~DeformableParticle()

```
netdem::DeformableParticle::~DeformableParticle ( )  [override]
```

7.37.2 Member Function Documentation

7.37.2.1 AddForce()

```
void netdem::DeformableParticle::AddForce (
            int node_id,
            Vec3d const & f )
```

7.37.2.2 ApplyContactForce() [1/2]

```
void netdem::DeformableParticle::ApplyContactForce (
            ContactPP const * cnt )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.3 ApplyContactForce() [2/2]

```
void netdem::DeformableParticle::ApplyContactForce (
            ContactPW const * cnt )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.4 ClearForce()

```
void netdem::DeformableParticle::ClearForce ( )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.5 Clone()

```
Particle * netdem::DeformableParticle::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.6 GetVelocity()

```
Vec3d netdem::DeformableParticle::GetVelocity (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.7 InitFEMSimulator()

```
void netdem::DeformableParticle::InitFEMSimulator ( )  [private]
```

7.37.2.8 SaveAsVTK()

```
void netdem::DeformableParticle::SaveAsVTK (
            std::string const & filename )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.9 SaveSurfaceAsVTK()

```
void netdem::DeformableParticle::SaveSurfaceAsVTK (
            std::string const & filename )
```

7.37.2.10 SetDensity()

```
void netdem::DeformableParticle::SetDensity (
            double dens )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.11 SetPosition()

```
void netdem::DeformableParticle::SetPosition (
            double x,
            double y,
            double z )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.12 SetQuaternion()

```
void netdem::DeformableParticle::SetQuaternion (
            double q_0,
            double q_1,
            double q_2,
            double q_3 )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.13 SetRodrigues()

```
void netdem::DeformableParticle::SetRodrigues (
            double angle,
            double axis_x,
            double axis_y,
            double axis_z )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.14 SetShape()

```
void netdem::DeformableParticle::SetShape (
            Shape * s )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.15 SetVelocity()

```
void netdem::DeformableParticle::SetVelocity (
            double v_x,
            double v_y,
            double v_z )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.16 UpdateBound()

```
void netdem::DeformableParticle::UpdateBound ( )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.17 UpdateMotion()

```
void netdem::DeformableParticle::UpdateMotion (
            double dt )  [override], [virtual]
```

Reimplemented from netdem::Particle.

7.37.2.18 UpdateShape()

```
void netdem::DeformableParticle::UpdateShape ( )
```

7.37.3 Member Data Documentation

7.37.3.1 fem_simulator

```
FEMSimulator netdem::DeformableParticle::fem_simulator
```

7.37.3.2 mesh_res

```
int netdem::DeformableParticle::mesh_res {20}
```

7.37.3.3 tetmesh

```
TetMesh netdem::DeformableParticle::tetmesh
```

7.37.3.4 trimesh

```
TriMesh* netdem::DeformableParticle::trimesh {nullptr}
```

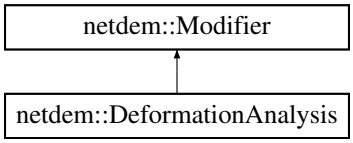
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/deformable_particle.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/deformable_particle.cpp

7.38 netdem::DeformationAnalysis Class Reference

`#include <deformation_analysis.hpp>`

Inheritance diagram for netdem::DeformationAnalysis:



Classes

- class Settings

Public Member Functions

- DeformationAnalysis ()
- void SetParticlesFromScene ()
- void SetParticles (const VecXT< int > &id_list)
- void SetParticles (const std::initializer_list< int > &id_list)
- Modifier * Clone () const override
- void Init (Simulation *sim) override
- void Execute () override
- void Update ()

Public Attributes

- Settings settings
- VecXT< int > particle_id_list
- std::unordered_map< Particle *, std::pair< bool, DeformableParticle > > particle_map

Private Member Functions

- void SolveDeformation ()
- void SaveFEMAsVTK ()
- void EvaluateBCForce (DeformableParticle *const p_deformable_ptr, ContactPP *const cnt)
- void EvaluateBCForce (DeformableParticle *const p_deformable_ptr, ContactPW *const cnt)
- void SetSettings (FEMSimulator *const fem_sim)
- std::string GetFEMResultFileName (Particle *const p_ptr)
- bool CheckIfToExecute ()

Private Attributes

- bool use_particles_in_scene {false}
- int solve_cycle_previous {0}
- double solve_time_previous {0}
- int save_cycle_previous {0}
- double save_time_previous {0}

7.38.1 Constructor & Destructor Documentation

7.38.1.1 DeformationAnalysis()

```
netdem::DeformationAnalysis::DeformationAnalysis ( )
```

7.38.2 Member Function Documentation

7.38.2.1 CheckIfToExecute()

```
bool netdem::DeformationAnalysis::CheckIfToExecute ( )  [private]
```

7.38.2.2 Clone()

```
Modifier * netdem::DeformationAnalysis::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.38.2.3 EvaluateBCForce() [1/2]

```
void netdem::DeformationAnalysis::EvaluateBCForce (
            DeformableParticle *const p_deformable_ptr,
            ContactPP *const cnt )  [private]
```

7.38.2.4 EvaluateBCForce() [2/2]

```
void netdem::DeformationAnalysis::EvaluateBCForce (
            DeformableParticle *const p_deformable_ptr,
            ContactPW *const cnt )  [private]
```

7.38.2.5 Execute()

```
void netdem::DeformationAnalysis::Execute ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.38.2.6 GetFEMResultFileName()

```
std::string netdem::DeformationAnalysis::GetFEMResultFileName (
            Particle *const p_ptr )  [private]
```

7.38.2.7 Init()

```
void netdem::DeformationAnalysis::Init (
            Simulation * sim )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.38.2.8 SaveFEMAsVTK()

```
void netdem::DeformationAnalysis::SaveFEMAsVTK ( )  [private]
```

7.38.2.9 SetParticles() [1/2]

```
void netdem::DeformationAnalysis::SetParticles (
            const std::initializer_list< int > & id_list )
```

7.38.2.10 SetParticles() [2/2]

```
void netdem::DeformationAnalysis::SetParticles (
            const VecXT< int > & id_list )
```

7.38.2.11 SetParticlesFromScene()

```
void netdem::DeformationAnalysis::SetParticlesFromScene ( )
```

7.38.2.12 SetSettings()

```
void netdem::DeformationAnalysis::SetSettings (
            FEMSimulator *const fem_sim )  [private]
```

7.38.2.13 SolveDeformation()

```
void netdem::DeformationAnalysis::SolveDeformation ( )  [private]
```

7.38.2.14 Update()

```
void netdem::DeformationAnalysis::Update ( )  [virtual]
```

Reimplemented from netdem::Modifier.

7.38.3 Member Data Documentation

7.38.3.1 particle_id_list

```
VecXT<int> netdem::DeformationAnalysis::particle_id_list
```

7.38.3.2 particle_map

```
std::unordered_map<Particle *, std::pair<bool, DeformableParticle> > netdem::Deformation↩
Analysis::particle_map
```

7.38.3.3 save_cycle_previous

```
int netdem::DeformationAnalysis::save_cycle_previous {0}  [private]
```

7.38.3.4 save_time_previous

```
double netdem::DeformationAnalysis::save_time_previous {0}  [private]
```

7.38.3.5 settings

```
Settings netdem::DeformationAnalysis::settings
```

7.38.3.6 solve_cycle_previous

```
int netdem::DeformationAnalysis::solve_cycle_previous {0}  [private]
```

7.38.3.7 solve_time_previous

```
double netdem::DeformationAnalysis::solve_time_previous {0}  [private]
```

7.38.3.8 use_particles_in_scene

```
bool netdem::DeformationAnalysis::use_particles_in_scene {false}  [private]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/deformation_analysis.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/deformation_analysis.cpp

7.39 netdem::DEMFragment Class Reference

```
#include <dem_fragment.hpp>
```

Public Member Functions

- void InitLevelSet (double corner_x, double corner_y, double corner_z, double sp, int dim_x, int dim_y, int dim_z)
- void ResolverOverlap (DEMFragment *const frag_q)
- void ReInitSTLModel ()

Public Attributes

- Shape::Type shape_type {Shape::Type::trimesh}
- double sphere_size {1.0}
- Vec3d pos {0, 0, 0}
- STLModel stl_model
- Vec3d vel {0, 0, 0}
- Vec3d spin {0, 0, 0}
- LevelSetFunction level_set

7.39.1 Member Function Documentation

7.39.1.1 InitLevelSet()

```
void DEMFragment::InitLevelSet (
            double corner_x,
            double corner_y,
            double corner_z,
            double sp,
            int dim_x,
            int dim_y,
            int dim_z )
```

7.39.1.2 ReInitSTLModel()

```
void DEMFragment::ReInitSTLModel ( )
```

7.39.1.3 ResolverOverlap()

```
void DEMFragment::ResolverOverlap (
            DEMFragment *const frag_q )
```

7.39.2 Member Data Documentation

7.39.2.1 level_set

```
LevelSetFunction netdem::DEMFragment::level_set
```

7.39.2.2 pos

```
Vec3d netdem::DEMFragment::pos {0, 0, 0}
```

7.39.2.3 shape_type

```
Shape::Type netdem::DEMFragment::shape_type {Shape::Type::trimesh}
```

7.39.2.4 sphere_size

```
double netdem::DEMFragment::sphere_size {1.0}
```

7.39.2.5 spin

```
Vec3d netdem::DEMFragment::spin {0, 0, 0}
```

7.39.2.6 stl_model

```
STLModel netdem::DEMFragment::stl_model
```

7.39.2.7 vel

```
Vec3d netdem::DEMFragment::vel {0, 0, 0}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/dem_fragment.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/dem_fragment.cpp

7.40 netdem::DEMObjectPool Class Reference

particles and contacts are frequently added to or removed from the scene. The pool strategy is used to avoid the frequently construction and de-construction of object instances. When a particle or wall needs to be added, an instances will be obtained from the pool. When a particle or wall needs to be removed,it is recycled and stored in the pool. to do: object pool need to be improved

```
#include <dem_object_pool.hpp>
```

Public Member Functions

- DEMObjectPool (const DEMObjectPool &)=delete
- DEMObjectPool & operator= (const DEMObjectPool &)=delete
- Particle * GetParticle ()
- ContactPP * GetContactPP ()
- ContactPW * GetContactPW ()
- ContactPP * Clone (ContactPP const *cnt)
- ContactPW * Clone (ContactPW const *cnt)
- void RecycleParticle (Particle **p)
- void RecycleContactPP (ContactPP **cnt)
- void RecycleContactPW (ContactPW **cnt)
- void RecycleParticle (VecXT< Particle * > *p_list)
- void RecycleContactPP (VecXT< ContactPP * > *cnt_list)
- void RecycleContactPW (VecXT< ContactPW * > *cnt_list)
- void RecycleParticle (VecXT< VecXT< Particle * > > *p_list)
- ~DEMObjectPool ()

Static Public Member Functions

- static DEMObjectPool & GetInstance ()

Private Member Functions

- DEMObjectPool ()

Private Attributes

- VecXT< Particle * > particle_pool
- VecXT< ContactPP * > contact_pp_pool
- VecXT< ContactPW * > contact_pw_pool

7.40.1 Detailed Description

particles and contacts are frequently added to or removed from the scene. The pool strategy is used to avoid the frequently construction and de-construction of object instances. When a particle or wall needs to be added, an instances will be obtained from the pool. When a particle or wall needs to be removed,it is recycled and stored in the pool. to do: object pool need to be improved

7.40.2 Constructor & Destructor Documentation

7.40.2.1 DEMObjectPool() [1/2]

```
netdem::DEMObjectPool::DEMObjectPool (
            const DEMObjectPool &  )  [delete]
```

7.40.2.2 ~DEMObjectPool()

```
netdem::DEMObjectPool::~DEMObjectPool ( )
```

7.40.2.3 DEMObjectPool() [2/2]

```
netdem::DEMObjectPool::DEMObjectPool ( )  [inline], [private]
```

7.40.3 Member Function Documentation

7.40.3.1 Clone() [1/2]

```
ContactPP * netdem::DEMObjectPool::Clone (
            ContactPP const * cnt )
```

7.40.3.2 Clone() [2/2]

```
ContactPW * netdem::DEMObjectPool::Clone (
            ContactPW const * cnt )
```

7.40.3.3 GetContactPP()

```
ContactPP * netdem::DEMObjectPool::GetContactPP ( )
```

7.40.3.4 GetContactPW()

```
ContactPW * netdem::DEMObjectPool::GetContactPW ( )
```

7.40.3.5 GetInstance()

```
static DEMObjectPool & netdem::DEMObjectPool::GetInstance ( )  [inline], [static]
```

7.40.3.6 GetParticle()

```
Particle * netdem::DEMObjectPool::GetParticle ( )
```

7.40.3.7 operator=()

```
DEMObjectPool & netdem::DEMObjectPool::operator= (
            const DEMObjectPool &  )  [delete]
```

7.40.3.8 RecycleContactPP() [1/2]

```
void netdem::DEMObjectPool::RecycleContactPP (
            ContactPP ** cnt )
```

7.40.3.9 RecycleContactPP() [2/2]

```
void netdem::DEMObjectPool::RecycleContactPP (
            VecXT< ContactPP * > * cnt_list )
```

7.40.3.10 RecycleContactPW() [1/2]

```
void netdem::DEMObjectPool::RecycleContactPW (
            ContactPW ** cnt )
```

7.40.3.11 RecycleContactPW() [2/2]

```
void netdem::DEMObjectPool::RecycleContactPW (
            VecXT< ContactPW * > * cnt_list )
```

7.40.3.12 RecycleParticle() [1/3]

```
void netdem::DEMObjectPool::RecycleParticle (
            Particle ** p )
```

7.40.3.13 RecycleParticle() [2/3]

```
void netdem::DEMObjectPool::RecycleParticle (
            VecXT< Particle * > * p_list )
```

7.40.3.14 RecycleParticle() [3/3]

```
void netdem::DEMObjectPool::RecycleParticle (
            VecXT< VecXT< Particle * > > * p_list )
```

7.40.4 Member Data Documentation

7.40.4.1 contact_pp_pool

```
VecXT<ContactPP *> netdem::DEMObjectPool::contact_pp_pool  [private]
```

7.40.4.2 contact_pw_pool

```
VecXT<ContactPW *> netdem::DEMObjectPool::contact_pw_pool  [private]
```

7.40.4.3 particle_pool

```
VecXT<Particle *> netdem::DEMObjectPool::particle_pool  [private]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/dem_object_pool.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/dem_object_pool.cpp

7.41 netdem::DEMProfiler Class Reference

```
#include <dem_profiler.hpp>
```

Public Member Functions

- DEMProfiler ()
- void StartTimer (TimerType t_type)
- void EndTimer (TimerType t_type)
- void Clear ()
- void Print ()

Static Public Member Functions

- static int64t GetTimeMicros ()

Public Attributes

- int64t timer_list [TimerType::num_timers]
- int num_particles {0}
- int num_walls {0}
- int num_neighs {0}
- int num_neigh_builds {0}
- double num_neighs_per_p {0}

Private Member Functions

- int64t GetTotalTime ()

Private Attributes

- int64t t_start [TimerType::num_timers]
- bool timer_started [TimerType::num_timers]

7.41.1 Constructor & Destructor Documentation

7.41.1.1 DEMProfiler()

```
DEMProfiler::DEMProfiler ( )
```

7.41.2 Member Function Documentation

7.41.2.1 Clear()

```
void DEMProfiler::Clear ( )
```

7.41.2.2 EndTimer()

```
void DEMProfiler::EndTimer (
            TimerType t_type )
```

7.41.2.3 GetTimeMicros()

```
int64t DEMProfiler::GetTimeMicros ( )  [static]
```

7.41.2.4 GetTotalTime()

```
int64t DEMProfiler::GetTotalTime ( )  [inline], [private]
```

7.41.2.5 Print()

```
void DEMProfiler::Print ( )
```

7.41.2.6 StartTimer()

```
void DEMProfiler::StartTimer (
            TimerType t_type )
```

7.41.3 Member Data Documentation

7.41.3.1 num_neigh_builds

```
int netdem::DEMProfiler::num_neigh_builds {0}
```

7.41.3.2 num_neighs

```
int netdem::DEMProfiler::num_neighs {0}
```

7.41.3.3 num_neighs_per_p

```
double netdem::DEMProfiler::num_neighs_per_p {0}
```

7.41.3.4 num_particles

```
int netdem::DEMProfiler::num_particles {0}
```

7.41.3.5 num_walls

```
int netdem::DEMProfiler::num_walls {0}
```

7.41.3.6 t_start

```
int64t netdem::DEMProfiler::t_start[TimerType::num_timers]  [private]
```

7.41.3.7 timer_list

```
int64t netdem::DEMProfiler::timer_list[TimerType::num_timers]
```

7.41.3.8 timer_started

```
bool netdem::DEMProfiler::timer_started[TimerType::num_timers]  [private]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/dem_profiler.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/dem_profiler.cpp

7.42 netdem::DEMSolver Class Reference

```
#include <dem_solver.hpp>
```

Public Types

- enum CyclePoint {
  pre , mid_0 , mid_1 , mid_2 ,
  post , num_cycle_points }

  this is used by modifiers.

Public Member Functions

- DEMSolver ()
- void Init (Simulation *sim)
- void UpdatePreModifiers ()

  step 1
- void UpdateLinkedList ()

  step 2
- void UpdateContacts ()

  step 3
- void UpdateParticles ()

  step 4
- void UpdateWalls ()

  step 5
- void UpdatePostModifiers ()

  step 6
- void Cycle (int num_cycles)
- void Solve (double time)

Public Attributes

- double timestep {1.0e-4}
- DEMProfiler dem_profiler

  track the matics of interest
- ContactSolverFactory contact_solver_factory

  stores the solvers for contacts between particles and walls

Private Member Functions

- void UpdateMidModifiers (CyclePoint cyc_point)
- void Cycle ()
- void DryCycle ()
- void SolveContactPP (Particle *const p_ii, NeighPofP *const neigh_tuple, ContactSolverFactory *const solver_factory)
- void SolveContactPW (Particle *const p_ii, NeighWofP *const neigh_tuple, ContactSolverFactory *const solver_factory)

Private Attributes

- Simulation * sim {nullptr}

7.42.1 Member Enumeration Documentation

7.42.1.1 CyclePoint

enum netdem::DEMSolver::CyclePoint

this is used by modifiers.

Enumerator

pre	
mid_0	
mid_1	
mid_2	
post	
num_cycle_points	

7.42.2 Constructor & Destructor Documentation

7.42.2.1 DEMSolver()

```
DEMSolver::DEMSolver ( )
```

7.42.3 Member Function Documentation

7.42.3.1 Cycle() [1/2]

```
void DEMSolver::Cycle ( )  [private]
```

7.42.3.2 Cycle() [2/2]

```
void DEMSolver::Cycle (
            int num_cycles )
```

7.42.3.3 DryCycle()

```
void DEMSolver::DryCycle ( )  [private]
```

7.42.3.4 Init()

```
void DEMSolver::Init (
            Simulation * sim )
```

7.42.3.5 Solve()

```
void DEMSolver::Solve (
            double time )
```

7.42.3.6 SolveContactPP()

```
void DEMSolver::SolveContactPP (
            Particle *const p_ii,
            NeighPofP *const neigh_tuple,
            ContactSolverFactory *const solver_factory )  [private]
```

7.42.3.7 SolveContactPW()

```
void DEMSolver::SolveContactPW (
            Particle *const p_ii,
            NeighWofP *const neigh_tuple,
            ContactSolverFactory *const solver_factory )  [private]
```

7.42.3.8 UpdateContacts()

```
void DEMSolver::UpdateContacts ( )
```

step 3

7.42.3.9 UpdateLinkedList()

```
void DEMSolver::UpdateLinkedList ( )
```

step 2

7.42.3.10 UpdateMidModifiers()

```
void DEMSolver::UpdateMidModifiers (
            CyclePoint cyc_point )  [private]
```

7.42.3.11 UpdateParticles()

```
void DEMSolver::UpdateParticles ( )
```

step 4

7.42.3.12 UpdatePostModifiers()

```
void DEMSolver::UpdatePostModifiers ( )
```

step 6

7.42.3.13 UpdatePreModifiers()

```
void DEMSolver::UpdatePreModifiers ( )
```

step 1

7.42.3.14 UpdateWalls()

```
void DEMSolver::UpdateWalls ( )
```

step 5

7.42.4 Member Data Documentation

7.42.4.1 contact_solver_factory

```
ContactSolverFactory netdem::DEMSolver::contact_solver_factory
```

stores the solvers for contacts between particles and walls

7.42.4.2 dem_profiler

```
DEMProfiler netdem::DEMSolver::dem_profiler
```

track the matics of interest

7.42.4.3 sim

```
Simulation* netdem::DEMSolver::sim {nullptr}  [private]
```

7.42.4.4 timestep

```
double netdem::DEMSolver::timestep {1.0e-4}
```

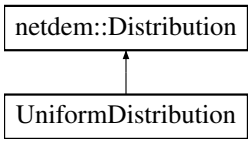
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/dem_solver.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/dem_solver.cpp

7.43 netdem::Distribution Class Reference

```
#include <distribution.hpp>
```

Inheritance diagram for netdem::Distribution:



Public Member Functions

- virtual double Get ()=0
- virtual VecXT< double > Get (int num)=0
- virtual ~Distribution ()

7.43.1 Detailed Description

Interface for distributions.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 ~Distribution()

```
virtual netdem::Distribution::~Distribution ( )  [inline], [virtual]
```

7.43.3 Member Function Documentation

7.43.3.1 Get() [1/2]

```
virtual double netdem::Distribution::Get ( )  [pure virtual]
```

Implemented in UniformDistribution.

7.43.3.2 Get() [2/2]

```
virtual VecXT< double > netdem::Distribution::Get (
            int num )  [pure virtual]
```

Implemented in UniformDistribution.

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/distribution.hpp

7.44 netdem::Domain Class Reference

```
#include <domain.hpp>
```

Public Member Functions

- Domain ()
- Domain (Vec3d const &bmin, Vec3d const &bmax)
- void Init ()
- void SetBound (double bmin_x, double bmin_y, double bmin_z, double bmax_x, double bmax_y, double bmax_z)

  set the domain size
- bool IsJudgeDomain (Particle const &p, Particle const &q)
- bool IsJudgeDomain (Particle const &p, Wall const &w)
- bool IsBelongToDomain (Particle const &p)
- bool IsBelongToDomain (ParticleData const &p)
- bool IsParticleProxyToSend (Particle const &p)
- bool IsParticleProxyToRecv (Particle const &p)
- bool IsParticleProxyToRecv (ParticleData const &p)
- void Print ()

  print domain information to the screen
- void ClearLinkedLists ()
- STLModel GetSTLModel ()

  for visualization purpose
- ~Domain ()

Public Attributes

- int my_rank {0}

  the rank id and the total number of processes of a mpi run.
- int num_procs {0}
- Vec3d bound_min {-0.5, -0.5, -0.5}
- Vec3d bound_max {0.5, 0.5, 0.5}
- CellManager cell_manager

  manages the cells, which are used in the borad-phase contact detection.
- VecXT< std::pair< Particle *, int > > outer_particle_list

7.44.1 Constructor & Destructor Documentation

7.44.1.1 Domain() [1/2]

```
Domain::Domain ( )
```

7.44.1.2 Domain() [2/2]

```
Domain::Domain (
            Vec3d const & bmin,
            Vec3d const & bmax )
```

7.44.1.3 ~Domain()

```
Domain::~Domain ( )
```

7.44.2 Member Function Documentation

7.44.2.1 ClearLinkedLists()

```
void Domain::ClearLinkedLists ( )
```

7.44.2.2 GetSTLModel()

```
STLModel Domain::GetSTLModel ( )
```

for visualization purpose

7.44.2.3 Init()

```
void Domain::Init ( )
```

7.44.2.4 IsBelongToDomain() [1/2]

```
bool Domain::IsBelongToDomain (
            Particle const & p )
```

to check if the particle belongs to this domain. A particle is assumed to belong to this domain if its centroid is inside this domain.

7.44.2.5 IsBelongToDomain() [2/2]

```
bool Domain::IsBelongToDomain (
            ParticleData const & p )
```

7.44.2.6 IsJudgeDomain() [1/2]

```
bool Domain::IsJudgeDomain (
            Particle const & p,
            Particle const & q )
```

to check if the contact needs to be solved in this domain (a contact would be checked in only one domain to avoid duplications

7.44.2.7 IsJudgeDomain() [2/2]

```
bool Domain::IsJudgeDomain (
            Particle const & p,
            Wall const & w )
```

7.44.2.8 IsParticleProxyToRecv() [1/2]

```
bool Domain::IsParticleProxyToRecv (
            Particle const & p )
```

to check if a particle need to be received into this domain as a proxy for contact detection and resolution. Particle is a class for DEM calculation, PartilceData is a struct for MPI to exchange particle information.

7.44.2.9 IsParticleProxyToRecv() [2/2]

```
bool Domain::IsParticleProxyToRecv (
            ParticleData const & p )
```

7.44.2.10 IsParticleProxyToSend()

```
bool Domain::IsParticleProxyToSend (
            Particle const & p )
```

to check if a particle need to be sent to other domains as a proxy for contact detection and resolution.

7.44.2.11 Print()

```
void Domain::Print ( )
```

print domain information to the screen

7.44.2.12 SetBound()

```
void Domain::SetBound (
            double bmin_x,
            double bmin_y,
            double bmin_z,
            double bmax_x,
            double bmax_y,
            double bmax_z )
```

set the domain size

7.44.3 Member Data Documentation

7.44.3.1 bound_max

```
Vec3d netdem::Domain::bound_max {0.5, 0.5, 0.5}
```

7.44.3.2 bound_min

```
Vec3d netdem::Domain::bound_min {-0.5, -0.5, -0.5}
```

the lower and upper bounds of the x, y, and z coordinates of the domain. By default, the domain is set to a unit cube centered at origin.

7.44.3.3 cell_manager

```
CellManager netdem::Domain::cell_manager
```

manages the cells, which are used in the borad-phase contact detection.

7.44.3.4 my_rank

```
int netdem::Domain::my_rank {0}
```

the rank id and the total number of processes of a mpi run.

7.44.3.5 num_procs

```
int netdem::Domain::num_procs {0}
```

7.44.3.6 outer_particle_list

```
VecXT<std::pair<Particle *, int> > netdem::Domain::outer_particle_list
```

manages the particles that overlaps with this domains. Siimlar to the linked-list algorithm for cells in dem_solver, this list is updated by UpdateLinkedDomains of particle, and will be used by mpi_manager: if a particle is linked to a domain, it will be transferred to this domain for contact detection.

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/domain.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/domain.cpp

7.45 netdem::DomainManager Class Reference

`#include <domain_manager.hpp>`

Public Member Functions

- DomainManager ()
- void Init (Simulation *s)

  Initialize the domain and sub-domains.
- void Init ()
- void SetBound (double bmin_x, double bmin_y, double bmin_z, double bmax_x, double bmax_y, double bmax_z)
- void SetDecomposition (int num_div_x, int num_div_y, int num_div_z)
- Domain * GetSelfDomain ()

Public Attributes

- Vec3d bound_min {-0.5, -0.5, -0.5}
- Vec3d bound_max {0.5, 0.5, 0.5}
- Vec3i num_div {1, 1, 1}
- VecXT< Domain > domain_list

  the instances of sub-domains

Private Attributes

- Simulation * sim {nullptr}

7.45.1 Constructor & Destructor Documentation

7.45.1.1 DomainManager()

```
DomainManager::DomainManager ( )
```

7.45.2 Member Function Documentation

7.45.2.1 GetSelfDomain()

```
Domain * DomainManager::GetSelfDomain ( )
```

7.45.2.2 Init() [1/2]

```
void DomainManager::Init ( )
```

7.45.2.3 Init() [2/2]

```
void DomainManager::Init (
            Simulation * s )
```

Initialize the domain and sub-domains.

7.45.2.4 SetBound()

```
void DomainManager::SetBound (
            double bmin_x,
            double bmin_y,
            double bmin_z,
            double bmax_x,
            double bmax_y,
            double bmax_z )
```

7.45.2.5 SetDecomposition()

```
void DomainManager::SetDecomposition (
            int num_div_x,
            int num_div_y,
            int num_div_z )
```

7.45.3 Member Data Documentation

7.45.3.1 bound_max

```
Vec3d netdem::DomainManager::bound_max {0.5, 0.5, 0.5}
```

7.45.3.2 bound_min

```
Vec3d netdem::DomainManager::bound_min {-0.5, -0.5, -0.5}
```

the lower and upper bounds of the x, y, and z coordinates of the whole domain. By default, the domain is set to a unit cube centered at origin.

7.45.3.3 domain_list

```
VecXT<Domain> netdem::DomainManager::domain_list
```

the instances of sub-domains

7.45.3.4 num_div

```
Vec3i netdem::DomainManager::num_div {1, 1, 1}
```

the number of domain devisions in each dimension. Currently, only a priliminary domain devision algorithm is implemented, that the domain is uniformly devided into n_x ∗ n_y ∗ n_z sub-domains

7.45.3.5 sim

```
Simulation* netdem::DomainManager::sim {nullptr}  [private]
```

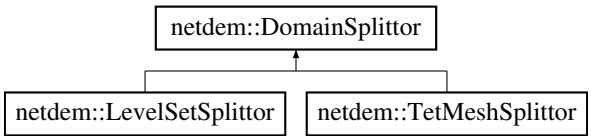
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/domain_manager.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/domain_manager.cpp

7.46 netdem::DomainSplittor Class Reference

```
#include <domain_splittor.hpp>
```

Inheritance diagram for netdem::DomainSplittor:



Public Member Functions

- DomainSplittor ()
- virtual void InitFromSTL (std::string const &stl_file, int res)
- virtual void InitFromSTL (STLModel const &stl_model, int res)=0
- virtual void GetPeriDigmNodes (VecXT< Vec3d > ∗const nodes, VecXT< double > ∗const node_vols)=0
- virtual void MakePorosity (double porosity)=0
- virtual STLModel GetSTLModel ()=0
- virtual STLModel GetSTLModel (const VecXT< int > &indices)=0

7.46.1 Constructor & Destructor Documentation

7.46.1.1 DomainSplittor()

```
netdem::DomainSplittor::DomainSplittor ( )  [inline]
```

7.46.2 Member Function Documentation

7.46.2.1 GetPeriDigmNodes()

```
virtual void netdem::DomainSplittor::GetPeriDigmNodes (
            VecXT< Vec3d > *const nodes,
            VecXT< double > *const node_vols )  [pure virtual]
```

Implemented in netdem::LevelSetSplittor, and netdem::TetMeshSplittor.

7.46.2.2 GetSTLModel() [1/2]

```
virtual STLModel netdem::DomainSplittor::GetSTLModel ( )  [pure virtual]
```

Implemented in netdem::LevelSetSplittor, and netdem::TetMeshSplittor.

7.46.2.3 GetSTLModel() [2/2]

```
virtual STLModel netdem::DomainSplittor::GetSTLModel (
            const VecXT< int > & indices )  [pure virtual]
```

Implemented in netdem::LevelSetSplittor, and netdem::TetMeshSplittor.

7.46.2.4 InitFromSTL() [1/2]

```
virtual void netdem::DomainSplittor::InitFromSTL (
            std::string const & stl_file,
            int res )  [inline], [virtual]
```

7.46.2.5 InitFromSTL() [2/2]

```
virtual void netdem::DomainSplittor::InitFromSTL (
            STLModel const & stl_model,
            int res )  [pure virtual]
```

Implemented in netdem::LevelSetSplittor, and netdem::TetMeshSplittor.

7.46.2.6 MakePorosity()

```
virtual void netdem::DomainSplittor::MakePorosity (
            double porosity )  [pure virtual]
```

Implemented in netdem::LevelSetSplittor, and netdem::TetMeshSplittor.

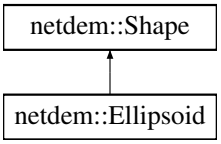
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/domain_splittor.hpp

7.47 netdem::Ellipsoid Class Reference

```
#include <shape_ellipsoid.hpp>
```

Inheritance diagram for netdem::Ellipsoid:



Public Member Functions

- Ellipsoid ()
- Ellipsoid (double a, double b, double c)
- Shape * Clone () const override
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void Init ()
- void UpdateNodes () override
- void UpdateShapeProperties () override
- void SetSize (double d) override
- STLModel GetSTLModel (int num_facets=400) override
- Vec3d SupportPoint (Vec3d const &dir) override
- VecXT< Vec3d > SupportPoints (Vec3d const &dir) override
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- double CalculateRho (Vec3d const &dir)
- void Print () override

Public Attributes

- double axis_a {0.5}
- double axis_b {0.5}
- double axis_c {0.5}

Additional Inherited Members

7.47.1 Constructor & Destructor Documentation

7.47.1.1 Ellipsoid() [1/2]

```
Ellipsoid::Ellipsoid ( )
```

7.47.1.2 Ellipsoid() [2/2]

```
Ellipsoid::Ellipsoid (
            double a,
            double b,
            double c )
```

7.47.2 Member Function Documentation

7.47.2.1 CalculateRho()

```
double Ellipsoid::CalculateRho (
            Vec3d const & dir )
```

7.47.2.2 Clone()

```
Shape * Ellipsoid::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.3 GetSTLModel()

```
STLModel Ellipsoid::GetSTLModel (
            int num_facets = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.4 Init()

```
void Ellipsoid::Init ( )
```

7.47.2.5 InitFromJson()

```
void Ellipsoid::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.6 PackJson()

```
nlohmann::json Ellipsoid::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.7 Print()

```
void Ellipsoid::Print ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.8 SetSize()

```
void Ellipsoid::SetSize (
            double d )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.9 SignedDistance()

```
double Ellipsoid::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.10 SupportPoint()

```
Vec3d Ellipsoid::SupportPoint (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.11 SupportPoints()

```
VecXT< Vec3d > Ellipsoid::SupportPoints (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.12 SurfacePoint()

```
Vec3d Ellipsoid::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.47.2.13 UpdateNodes()

```
void Ellipsoid::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.2.14 UpdateShapeProperties()

```
void Ellipsoid::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.47.3 Member Data Documentation

7.47.3.1 axis_a

```
double netdem::Ellipsoid::axis_a {0.5}
```

7.47.3.2 axis_b

```
double netdem::Ellipsoid::axis_b {0.5}
```

7.47.3.3 axis_c

```
double netdem::Ellipsoid::axis_c {0.5}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_ellipsoid.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_ellipsoid.cpp

7.48 netdem::FEMSimulator Class Reference

```
#include <fem_simulator.hpp>
```

Public Member Functions

- FEMSimulator ()
- void SetMesh (const TetMesh &tetmesh)
- void Init ()
- void SetBCNodalVelocity (int nid, double vx, double vy, double vz, bool use_prescribed_vx, bool use_↩ prescribed_vy, bool use_prescribed_vz)
- void AddBCFacetForce (int bc_fid, double fx, double fy, double fz)
- void SetNodalVels (double v_x, double v_y, double v_z)
- void ClearBoundaryCondition ()
- void Solve (double dt)
- VecXT< Vec3d > GetNodalPositions (VecXT< int > nids)
- VecXT< Vec3d > GetNodalDisps (VecXT< int > nids)
- VecXT< Vec3d > GetNodalVels (VecXT< int > nids)
- void SaveAsVTK (std::string const &file_name)

Public Attributes

- double neo_k {6.94e5}
- double neo_mu {5.21e5}
- double density {500.0}
- double damp_coef {0.7}
- Vec3d gravity_coef {0.0, 0.0, 0.0}
- double timestep {1.0e-4}
- VecXT< Vec3d > nodes
- VecXT< Vec4i > elements
- VecXT< Vec3d > nodes_ref
- VecXT< Vec3i > bound_facets
- VecXT< int > bound_nodes
- VecXT< double > elemental_vol
- VecXT< VecNT< double, 6 > > elemental_stress
- VecXT< double > nodal_vols
- VecXT< Vec3d > nodal_vels
- VecXT< Vec3d > bc_facet_forces
- VecXT< VecNT< double, 6 > > bc_nodal_vels

Protected Member Functions

- void Advance (double dt)
- void InitInitialCondition ()
- double GetElementVolume (Vec3d const &v0, Vec3d const &v1, Vec3d const &v2, Vec3d const &v3)
- Mat3d GetDeformationGradient (Vec3d const &v0_new, Vec3d const &v1_new, Vec3d const &v2_new, Vec3d const &v3_new, Vec3d const &v0_ref, Vec3d const &v1_ref, Vec3d const &v2_ref, Vec3d const &v3_ref)
- Mat3d GetCauchyStress (Mat3d const &def_grad)
- MatNd< 4, 3 > GetInternalForces (Mat3d const &cauchy_stress, Vec3d const &v0, Vec3d const &v1, Vec3d const &v2, Vec3d const &v3)

Protected Attributes

- VecXT< Vec3d > nodal_forces_int
- VecXT< Vec3d > nodal_forces_ext
- VecXT< Vec3d > nodal_vels_ave

7.48.1 Constructor & Destructor Documentation

7.48.1.1 FEMSimulator()

```
FEMSimulator::FEMSimulator ( )
```

7.48.2 Member Function Documentation

7.48.2.1 AddBCFacetForce()

```
void FEMSimulator::AddBCFacetForce (
            int bc_fid,
            double fx,
            double fy,
            double fz )
```

7.48.2.2 Advance()

```
void FEMSimulator::Advance (
            double dt )  [protected]
```

7.48.2.3 ClearBoundaryCondition()

```
void FEMSimulator::ClearBoundaryCondition ( )
```

7.48.2.4 GetCauchyStress()

```
Mat3d FEMSimulator::GetCauchyStress (
            Mat3d const & def_grad )  [protected]
```

7.48.2.5 GetDeformationGradient()

```
Mat3d FEMSimulator::GetDeformationGradient (
            Vec3d const & v0_new,
            Vec3d const & v1_new,
            Vec3d const & v2_new,
            Vec3d const & v3_new,
            Vec3d const & v0_ref,
            Vec3d const & v1_ref,
            Vec3d const & v2_ref,
            Vec3d const & v3_ref )  [protected]
```

7.48.2.6 GetElementVolume()

```
double FEMSimulator::GetElementVolume (
            Vec3d const & v0,
            Vec3d const & v1,
            Vec3d const & v2,
            Vec3d const & v3 )  [protected]
```

7.48.2.7 GetInternalForces()

```
MatNd< 4, 3 > FEMSimulator::GetInternalForces (
            Mat3d const & cauchy_stress,
            Vec3d const & v0,
            Vec3d const & v1,
            Vec3d const & v2,
            Vec3d const & v3 )  [protected]
```

7.48.2.8 GetNodalDisps()

```
VecXT< Vec3d > FEMSimulator::GetNodalDisps (
            VecXT< int > nids )
```

7.48.2.9 GetNodalPositions()

```
VecXT< Vec3d > FEMSimulator::GetNodalPositions (
            VecXT< int > nids )
```

7.48.2.10 GetNodalVels()

```
VecXT< Vec3d > FEMSimulator::GetNodalVels (
            VecXT< int > nids )
```

7.48.2.11 Init()

```
void FEMSimulator::Init ( )
```

7.48.2.12 InitInitialCondition()

```
void FEMSimulator::InitInitialCondition ( )  [protected]
```

7.48.2.13 SaveAsVTK()

```
void FEMSimulator::SaveAsVTK (
            std::string const & file_name )
```

7.48.2.14 SetBCNodalVelocity()

```
void FEMSimulator::SetBCNodalVelocity (
            int nid,
            double vx,
            double vy,
            double vz,
            bool use_prescribed_vx,
            bool use_prescribed_vy,
            bool use_prescribed_vz )
```

7.48.2.15 SetMesh()

```
void FEMSimulator::SetMesh (
            const TetMesh & tetmesh )
```

7.48.2.16 SetNodalVels()

```
void FEMSimulator::SetNodalVels (
            double v_x,
            double v_y,
            double v_z )
```

7.48.2.17 Solve()

```
void FEMSimulator::Solve (
            double dt )
```

7.48.3 Member Data Documentation

7.48.3.1 bc_facet_forces

```
VecXT<Vec3d> netdem::FEMSimulator::bc_facet_forces
```

7.48.3.2 bc_nodal_vels

```
VecXT<VecNT<double, 6> > netdem::FEMSimulator::bc_nodal_vels
```

7.48.3.3 bound_facets

```
VecXT<Vec3i> netdem::FEMSimulator::bound_facets
```

7.48.3.4 bound_nodes

```
VecXT<int> netdem::FEMSimulator::bound_nodes
```

7.48.3.5 damp_coef

```
double netdem::FEMSimulator::damp_coef {0.7}
```

7.48.3.6 density

```
double netdem::FEMSimulator::density {500.0}
```

7.48.3.7 elemental_stress

```
VecXT<VecNT<double, 6> > netdem::FEMSimulator::elemental_stress
```

7.48.3.8 elemental_vol

```
VecXT<double> netdem::FEMSimulator::elemental_vol
```

7.48.3.9 elements

```
VecXT<Vec4i> netdem::FEMSimulator::elements
```

7.48.3.10 gravity_coef

```
Vec3d netdem::FEMSimulator::gravity_coef {0.0, 0.0, 0.0}
```

7.48.3.11 neo_k

```
double netdem::FEMSimulator::neo_k {6.94e5}
```

7.48.3.12 neo_mu

```
double netdem::FEMSimulator::neo_mu {5.21e5}
```

7.48.3.13 nodal_forces_ext

```
VecXT<Vec3d> netdem::FEMSimulator::nodal_forces_ext  [protected]
```

7.48.3.14 nodal_forces_int

```
VecXT<Vec3d> netdem::FEMSimulator::nodal_forces_int  [protected]
```

7.48.3.15 nodal_vels

```
VecXT<Vec3d> netdem::FEMSimulator::nodal_vels
```

7.48.3.16 nodal_vels_ave

```
VecXT<Vec3d> netdem::FEMSimulator::nodal_vels_ave  [protected]
```

7.48.3.17 nodal_vols

```
VecXT<double> netdem::FEMSimulator::nodal_vols
```

7.48.3.18 nodes

```
VecXT<Vec3d> netdem::FEMSimulator::nodes
```

7.48.3.19 nodes_ref

```
VecXT<Vec3d> netdem::FEMSimulator::nodes_ref
```

7.48.3.20 timestep

```
double netdem::FEMSimulator::timestep {1.0e-4}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/fem_simulator.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/fem_simulator.cpp

7.49 netdem::GeneralNet Class Reference

```
#include <general_net.hpp>
```

Public Member Functions

- void ResetModel ()
- void AddLayer (LayerName layer_name,...)
- void Train (const arma::mat &data_x, const arma::mat &data_y)
- arma::mat Predict (const arma::mat &data_x)
- arma::mat Classify (const arma::mat &data_x)
- arma::mat Regress (const arma::mat &data_x)
- void Load (std::string const &filename, std::string const &label)
- void Save (std::string const &filename, std::string const &label)

Public Attributes

- mlpack::ann::FFN model
- double step_size {0.01}
- int batch_size {32}
- double decay_rate_moment {0.9}
- double decay_rate_norm {0.9}
- double gradient_init_param {1e-8}
- int epochs {100}
- double stop_tol {1e-8}

7.49.1 Member Function Documentation

7.49.1.1 AddLayer()

```
void netdem::GeneralNet::AddLayer (
            LayerName layer_name,
             ...  )
```

7.49.1.2 Classify()

```
arma::mat netdem::GeneralNet::Classify (
            const arma::mat & data_x )
```

7.49.1.3 Load()

```
void netdem::GeneralNet::Load (
            std::string const & filename,
            std::string const & label )
```

7.49.1.4 Predict()

```
arma::mat netdem::GeneralNet::Predict (
            const arma::mat & data_x )
```

7.49.1.5 Regress()

```
arma::mat netdem::GeneralNet::Regress (
            const arma::mat & data_x )
```

7.49.1.6 ResetModel()

```
void netdem::GeneralNet::ResetModel ( )
```

7.49.1.7 Save()

```
void netdem::GeneralNet::Save (
            std::string const & filename,
            std::string const & label )
```

7.49.1.8 Train()

```
void netdem::GeneralNet::Train (
            const arma::mat & data_x,
            const arma::mat & data_y )
```

7.49.2 Member Data Documentation

7.49.2.1 batch_size

```
int netdem::GeneralNet::batch_size {32}
```

7.49.2.2 decay_rate_moment

```
double netdem::GeneralNet::decay_rate_moment {0.9}
```

7.49.2.3 decay_rate_norm

```
double netdem::GeneralNet::decay_rate_norm {0.9}
```

7.49.2.4 epochs

```
int netdem::GeneralNet::epochs {100}
```

7.49.2.5 gradient_init_param

```
double netdem::GeneralNet::gradient_init_param {1e-8}
```

7.49.2.6 model

```
mlpack::ann::FFN netdem::GeneralNet::model
```

7.49.2.7 step_size

```
double netdem::GeneralNet::step_size {0.01}
```

7.49.2.8 stop_tol

```
double netdem::GeneralNet::stop_tol {1e-8}
```

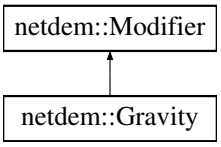
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/general_net.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/general_net.cpp

7.50 netdem::Gravity Class Reference

```
#include <gravity.hpp>
```

Inheritance diagram for netdem::Gravity:



Public Member Functions

- Gravity ()
- Modifier * Clone () const override
- void Init (Simulation *sim) override
- void Execute () override

Additional Inherited Members

7.50.1 Constructor & Destructor Documentation

7.50.1.1 Gravity()

```
netdem::Gravity::Gravity ( )
```

7.50.2 Member Function Documentation

7.50.2.1 Clone()

```
Modifier * netdem::Gravity::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.50.2.2 Execute()

```
void netdem::Gravity::Execute ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.50.2.3 Init()

```
void netdem::Gravity::Init (
            Simulation * sim )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

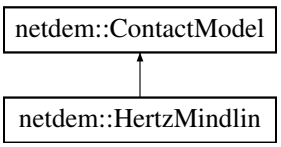
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/gravity.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/gravity.cpp

7.51 netdem::HertzMindlin Class Reference

```
#include <model_hertz_mindlin.hpp>
```

Inheritance diagram for netdem::HertzMindlin:



Public Member Functions

- HertzMindlin ()
- HertzMindlin (double kn, double kt, double beta, double mu)
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void SetProperty (nlohmann::json const &js) override
- ContactModel * Clone () const override
- void EvaluateForceMoment (ContactForces *const cnt_forces, CollisionGeometries &cnt_geoms, ContactPP *const cnt, double dt) const override
- void EvaluateForceMoment (ContactForces *const cnt_forces, CollisionGeometries &cnt_geoms, ContactPW *const cnt, double dt) const override
- void Print () const override

Public Attributes

- double kn {2.0e7}
- double kt {1.0e6}
- double beta {0.7}
- double mu {0.5}

Additional Inherited Members

7.51.1 Constructor & Destructor Documentation

7.51.1.1 HertzMindlin() [1/2]

```
netdem::HertzMindlin::HertzMindlin ( )
```

7.51.1.2 HertzMindlin() [2/2]

```
netdem::HertzMindlin::HertzMindlin (
            double kn,
            double kt,
            double beta,
            double mu )
```

7.51.2 Member Function Documentation

7.51.2.1 Clone()

```
ContactModel * netdem::HertzMindlin::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.51.2.2 EvaluateForceMoment() [1/2]

```
void netdem::HertzMindlin::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            CollisionGeometries & cnt_geoms,
            ContactPP *const cnt,
            double dt ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.51.2.3 EvaluateForceMoment() [2/2]

```
void netdem::HertzMindlin::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            CollisionGeometries & cnt_geoms,
            ContactPW *const cnt,
            double dt ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.51.2.4 InitFromJson()

```
void netdem::HertzMindlin::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.51.2.5 PackJson()

```
nlohmann::json netdem::HertzMindlin::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.51.2.6 Print()

```
void netdem::HertzMindlin::Print ( ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.51.2.7 SetProperty()

```
void netdem::HertzMindlin::SetProperty (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.51.3 Member Data Documentation

7.51.3.1 beta

```
double netdem::HertzMindlin::beta {0.7}
```

7.51.3.2 kn

```
double netdem::HertzMindlin::kn {2.0e7}
```

7.51.3.3 kt

```
double netdem::HertzMindlin::kt {1.0e6}
```

7.51.3.4 mu

```
double netdem::HertzMindlin::mu {0.5}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_hertz_mindlin.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_hertz_mindlin.cpp

7.52 netdem::InputProcessor Class Reference

`#include <input_processor.hpp>`

Public Member Functions

- InputProcessor ()
- void Init (Simulation *s)
- void ProcessJsonFile (std::string const &filename)
- void ProcessJson (nlohmann::json const &js)

Private Attributes

- Simulation * sim {nullptr}

7.52.1 Detailed Description

Create or modify a simulation using json input file.

7.52.2 Constructor & Destructor Documentation

7.52.2.1 InputProcessor()

```
netdem::InputProcessor::InputProcessor ( )
```

7.52.3 Member Function Documentation

7.52.3.1 Init()

```
void netdem::InputProcessor::Init (
            Simulation * s )
```

7.52.3.2 ProcessJson()

```
void netdem::InputProcessor::ProcessJson (
            nlohmann::json const & js )
```

7.52.3.3 ProcessJsonFile()

```
void netdem::InputProcessor::ProcessJsonFile (
            std::string const & filename )
```

7.52.4 Member Data Documentation

7.52.4.1 sim

```
Simulation* netdem::InputProcessor::sim {nullptr}  [private]
```

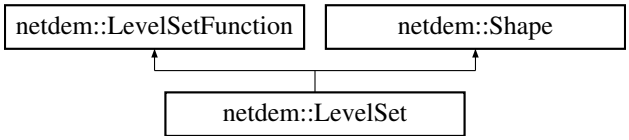
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/input/input_processor.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/input/input_processor.cpp

7.53 netdem::LevelSet Class Reference

```
#include <shape_level_set.hpp>
```

Inheritance diagram for netdem::LevelSet:



Public Member Functions

- LevelSet ()
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void InitFromSTL (std::string const &file, int mesh_res=25)
- void InitFromSTL (STLModel const &stl_model, int mesh_res=25)
- void InitFromDistanceMap (double c_0, double c_1, double c_2, double sp, const VecXT< VecXT< VecXT< double > > > &dist_map)
- void Init ()
- void AlignAxes ()
- void UpdateNodes () override
- void UpdateShapeProperties () override
- void SetSize (double d) override
- Shape * Clone () const override
- STLModel GetSTLModel (int res=400) override
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- void Print () override

Additional Inherited Members

7.53.1 Constructor & Destructor Documentation

7.53.1.1 LevelSet()

```
LevelSet::LevelSet ( )
```

7.53.2 Member Function Documentation

7.53.2.1 AlignAxes()

```
void LevelSet::AlignAxes ( )
```

7.53.2.2 Clone()

```
Shape * LevelSet::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.53.2.3 GetSTLModel()

```
STLModel LevelSet::GetSTLModel (
            int res = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.53.2.4 Init()

```
void LevelSet::Init ( )
```

7.53.2.5 InitFromDistanceMap()

```
void LevelSet::InitFromDistanceMap (
            double c_0,
            double c_1,
            double c_2,
            double sp,
            const VecXT< VecXT< VecXT< double > > > & dist_map )
```

7.53.2.6 InitFromJson()

```
void LevelSet::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.53.2.7 InitFromSTL() [1/2]

```
void netdem::LevelSet::InitFromSTL (
            std::string const & file,
            int mesh_res = 25 )
```

7.53.2.8 InitFromSTL() [2/2]

```
void LevelSet::InitFromSTL (
            STLModel const & stl_model,
            int mesh_res = 25 )
```

7.53.2.9 PackJson()

```
nlohmann::json LevelSet::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.53.2.10 Print()

```
void LevelSet::Print ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.53.2.11 SetSize()

```
void LevelSet::SetSize (
            double d )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.53.2.12 SignedDistance()

```
double LevelSet::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.53.2.13 SurfacePoint()

```
Vec3d LevelSet::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.53.2.14 UpdateNodes()

```
void LevelSet::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.53.2.15 UpdateShapeProperties()

```
void LevelSet::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

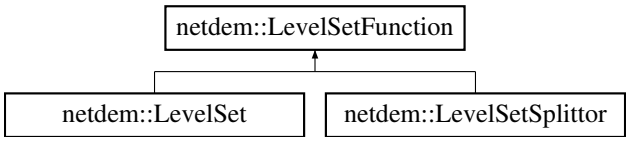
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_level_set.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_level_set.cpp

7.54 netdem::LevelSetFunction Class Reference

`#include <level_set_function.hpp>`

Inheritance diagram for netdem::LevelSetFunction:



Public Member Functions

- LevelSetFunction ()
- void SetCorner (double corner_x, double corner_y, double corner_z)
- void SetSpacing (double sp)
- void SetDimension (double dim_x, double dim_y, double dim_z)
- void InitFromSDFCalculator (const SDFCalculator &sdf_calculator)
- double SignedDistance (Vec3d const &pos)
- Vec3d GradientInterpolate (Vec3d const &pos)
- Vec3d GradientMinus (int i, int j, int k)
- Vec3d GradientPlus (int i, int j, int k)
- void Reinitialization (int iter, double dt)
- void Reinitialization ()

Public Attributes

- Vec3d corner {-0.5, -0.5, -0.5}
- double spacing {0.05}
- Vec3i dim {21, 21, 21}
- VecXT< VecXT< VecXT< double > > > signed_distance_table

7.54.1 Constructor & Destructor Documentation

7.54.1.1 LevelSetFunction()

`LevelSetFunction::LevelSetFunction ( )`

7.54.2 Member Function Documentation

7.54.2.1 GradientInterpolate()

```
Vec3d LevelSetFunction::GradientInterpolate (
            Vec3d const & pos )
```

7.54.2.2 GradientMinus()

```
Vec3d LevelSetFunction::GradientMinus (
            int i,
            int j,
            int k )
```

7.54.2.3 GradientPlus()

```
Vec3d LevelSetFunction::GradientPlus (
            int i,
            int j,
            int k )
```

7.54.2.4 InitFromSDFCalculator()

```
void LevelSetFunction::InitFromSDFCalculator (
            const SDFCalculator & sdf_calculator )
```

7.54.2.5 Reinitialization() [1/2]

```
void LevelSetFunction::Reinitialization ( )
```

7.54.2.6 Reinitialization() [2/2]

```
void LevelSetFunction::Reinitialization (
            int iter,
            double dt )
```

7.54.2.7 SetCorner()

```
void LevelSetFunction::SetCorner (
            double corner_x,
            double corner_y,
            double corner_z )
```

7.54.2.8 SetDimension()

```
void LevelSetFunction::SetDimension (
            double dim_x,
            double dim_y,
            double dim_z )
```

7.54.2.9 SetSpacing()

```
void LevelSetFunction::SetSpacing (
            double sp )
```

7.54.2.10 SignedDistance()

```
double LevelSetFunction::SignedDistance (
            Vec3d const & pos )
```

7.54.3 Member Data Documentation

7.54.3.1 corner

```
Vec3d netdem::LevelSetFunction::corner {-0.5, -0.5, -0.5}
```

7.54.3.2 dim

```
Vec3i netdem::LevelSetFunction::dim {21, 21, 21}
```

7.54.3.3 signed_distance_table

```
VecXT<VecXT<VecXT<double> > > netdem::LevelSetFunction::signed_distance_table
```

7.54.3.4 spacing

```
double netdem::LevelSetFunction::spacing {0.05}
```

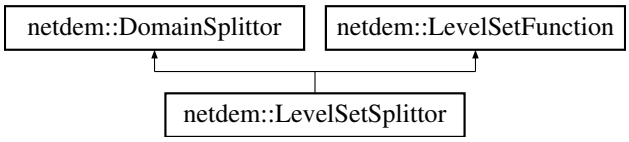
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/level_set_function.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/level_set_function.cpp

7.55 netdem::LevelSetSplittor Class Reference

```
#include <level_set_splittor.hpp>
```

Inheritance diagram for netdem::LevelSetSplittor:



Public Member Functions

- LevelSetSplittor ()
- void InitFromSTL (STLModel const &stl_model, int res) override
- void InitFromDistanceMap (std::string const &file_name)
- void InitFromDistanceMap (double corner_x, double corner_y, double corner_z, double sp, int dim_x, int dim_y, int dim_z, const VecXT< double > &dist_list)
- void GetPeriDigmNodes (VecXT< Vec3d > *const nodes, VecXT< double > *const node_vols) override
- void MakePorosity (double porosity) override
- STLModel GetSTLModel () override
- STLModel GetSTLModel (const VecXT< int > &node_indices) override

Private Attributes

- VecXT< Vec3i > node_grid_indices

Additional Inherited Members

7.55.1 Constructor & Destructor Documentation

7.55.1.1 LevelSetSplittor()

```
LevelSetSplittor::LevelSetSplittor ( )
```

7.55.2 Member Function Documentation

7.55.2.1 GetPeriDigmNodes()

```
void LevelSetSplittor::GetPeriDigmNodes (
            VecXT< Vec3d > *const nodes,
            VecXT< double > *const node_vols )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.55.2.2 GetSTLModel() [1/2]

```
STLModel LevelSetSplittor::GetSTLModel ( )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.55.2.3 GetSTLModel() [2/2]

```
STLModel LevelSetSplittor::GetSTLModel (
            const VecXT< int > & node_indices )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.55.2.4 InitFromDistanceMap() [1/2]

```
void LevelSetSplittor::InitFromDistanceMap (
            double corner_x,
            double corner_y,
            double corner_z,
            double sp,
            int dim_x,
            int dim_y,
            int dim_z,
            const VecXT< double > & dist_list )
```

7.55.2.5 InitFromDistanceMap() [2/2]

```
void LevelSetSplittor::InitFromDistanceMap (
            std::string const & file_name )
```

7.55.2.6 InitFromSTL()

```
void LevelSetSplittor::InitFromSTL (
            STLModel const & stl_model,
            int res )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.55.2.7 MakePorosity()

```
void LevelSetSplittor::MakePorosity (
            double porosity )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.55.3 Member Data Documentation

7.55.3.1 node_grid_indices

```
VecXT<Vec3i> netdem::LevelSetSplittor::node_grid_indices  [private]
```

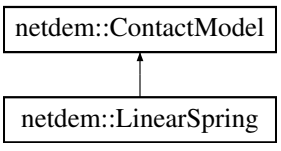
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/level_set_splittor.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/level_set_splittor.cpp

7.56 netdem::LinearSpring Class Reference

```
#include <model_linear_spring.hpp>
```

Inheritance diagram for netdem::LinearSpring:



Public Member Functions

- LinearSpring ()
- LinearSpring (double kn, double kt, double beta, double mu)
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void SetProperty (nlohmann::json const &js) override
- ContactModel * Clone () const override
- void EvaluateForceMoment (ContactForces *const cnt_forces, CollisionGeometries &cnt_geoms, ContactPP *const cnt, double dt) const override
- void EvaluateForceMoment (ContactForces *const cnt_forces, CollisionGeometries &cnt_geoms, ContactPW *const cnt, double dt) const override
- void Print () const override

Public Attributes

- double kn {2e6}
- double kt {1e6}
- double beta {0.7}
- double mu {0.5}
- bool use_viscous_damping {false}

Additional Inherited Members

7.56.1 Detailed Description

- kn, kt: normal and tangential contact stiffness. Generally, in the linear spring contact model, the contact force is the contact stiffness multiplied by the contact deformation.
- mu: friction coefficient, to incoorporate the Columnb's friction law. According to this law, the tangential force should not exceed the normal force multiplied by the friction coefficient.

7.56.2 Constructor & Destructor Documentation

7.56.2.1 LinearSpring() [1/2]

```
netdem::LinearSpring::LinearSpring ( )
```

7.56.2.2 LinearSpring() [2/2]

```
netdem::LinearSpring::LinearSpring (
            double kn,
            double kt,
            double beta,
            double mu )
```

7.56.3 Member Function Documentation

7.56.3.1 Clone()

```
ContactModel * netdem::LinearSpring::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.56.3.2 EvaluateForceMoment() [1/2]

```
void netdem::LinearSpring::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            CollisionGeometries & cnt_geoms,
            ContactPP *const cnt,
            double dt ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.56.3.3 EvaluateForceMoment() [2/2]

```
void netdem::LinearSpring::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            CollisionGeometries & cnt_geoms,
            ContactPW *const cnt,
            double dt ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.56.3.4 InitFromJson()

```
void netdem::LinearSpring::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.56.3.5 PackJson()

```
nlohmann::json netdem::LinearSpring::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.56.3.6 Print()

```
void netdem::LinearSpring::Print ( ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.56.3.7 SetProperty()

```
void netdem::LinearSpring::SetProperty (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.56.4 Member Data Documentation

7.56.4.1 beta

```
double netdem::LinearSpring::beta {0.7}
```

7.56.4.2 kn

```
double netdem::LinearSpring::kn {2e6}
```

7.56.4.3 kt

```
double netdem::LinearSpring::kt {1e6}
```

7.56.4.4 mu

```
double netdem::LinearSpring::mu {0.5}
```

7.56.4.5 use_viscous_damping

```
bool netdem::LinearSpring::use_viscous_damping {false}
```

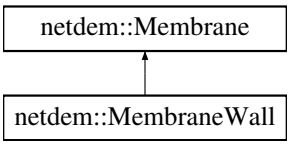
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_linear_spring.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_linear_spring.cpp

7.57 netdem::Membrane Class Reference

```
#include <membrane.hpp>
```

Inheritance diagram for netdem::Membrane:



Public Member Functions

- Membrane (double radius, double height)
- Membrane (double radius, double height, double mesh_size)
- Membrane (double radius, double height, double mesh_size, double center_x, double center_y, double center_z)
- void Remesh (double ele_size)
- void Init ()
- void SetBCNodalVelocity (int nid, double vx, double vy, double vz, bool use_prescribed_vx, bool use_↩ prescribed_vy, bool use_prescribed_vz)
- void Solve (double dt)
- void SaveAsVTK (std::string const &file_name)
- ~Membrane ()

Public Attributes

- double radius {0.25}
- double height {1.0}
- double mesh_size {0.1}
- Vec3d center {0, 0, 0}
- double neo_k {6.94e5}
- double neo_mu {5.21e5}
- double density {500.0}
- double thickness {0.3e-3}
- double damp_coef {0.7}
- double timestep {1.0e-4}
- VecXT< Vec3d > nodes
- VecXT< Vec3i > elements
- VecXT< Vec3d > elemental_stress
- VecXT< double > nodal_vols
- VecXT< Vec3d > nodal_vels
- VecXT< double > bc_facet_pressure
- VecXT< Vec3d > bc_facet_forces
- VecXT< VecNT< double, 6 > > bc_nodal_vels

Protected Member Functions

- void Advance (double dt)
- void InitMesh ()
- void InitInitialCondition ()
- Mat2d GetDeformationGradient (Vec3d const &v0, Vec3d const &v1, Vec3d const &v2)
- Mat2d GetCauchyStress (Mat2d const &def_grad)
- std::tuple< Mat3d, Mat3d > GetGlobalForces (Mat2d const &cauchy_stress, double pressure, Vec3d const &v0, Vec3d const &v1, Vec3d const &v2)

Protected Attributes

- double ref_ele_width {0.1}
- double ref_ele_height {0.1}
- double ref_ele_area {0.05}
- VecXT< Vec3d > nodal_forces_int
- VecXT< Vec3d > nodal_forces_ext

7.57.1 Constructor & Destructor Documentation

7.57.1.1 Membrane() [1/3]

```
Membrane::Membrane (
            double radius,
            double height )
```

7.57.1.2 Membrane() [2/3]

```
Membrane::Membrane (
            double radius,
            double height,
            double mesh_size )
```

7.57.1.3 Membrane() [3/3]

```
Membrane::Membrane (
            double radius,
            double height,
            double mesh_size,
            double center_x,
            double center_y,
            double center_z )
```

7.57.1.4 ~Membrane()

```
Membrane::~Membrane ( )
```

7.57.2 Member Function Documentation

7.57.2.1 Advance()

```
void Membrane::Advance (
            double dt )  [protected]
```

7.57.2.2 GetCauchyStress()

```
Mat2d Membrane::GetCauchyStress (
            Mat2d const & def_grad )  [protected]
```

7.57.2.3 GetDeformationGradient()

```
Mat2d Membrane::GetDeformationGradient (
            Vec3d const & v0,
            Vec3d const & v1,
            Vec3d const & v2 )  [protected]
```

7.57.2.4 GetGlobalForces()

```
tuple< Mat3d, Mat3d > Membrane::GetGlobalForces (
            Mat2d const & cauchy_stress,
            double pressure,
            Vec3d const & v0,
            Vec3d const & v1,
            Vec3d const & v2 )  [protected]
```

7.57.2.5 Init()

```
void Membrane::Init ( )
```

7.57.2.6 InitInitialCondition()

```
void Membrane::InitInitialCondition ( )  [protected]
```

7.57.2.7 InitMesh()

```
void Membrane::InitMesh ( )  [protected]
```

7.57.2.8 Remesh()

```
void Membrane::Remesh (
            double ele_size )
```

7.57.2.9 SaveAsVTK()

```
void Membrane::SaveAsVTK (
            std::string const & file_name )
```

7.57.2.10 SetBCNodalVelocity()

```
void Membrane::SetBCNodalVelocity (
            int nid,
            double vx,
            double vy,
            double vz,
            bool use_prescribed_vx,
            bool use_prescribed_vy,
            bool use_prescribed_vz )
```

7.57.2.11 Solve()

```
void Membrane::Solve (
            double dt )
```

7.57.3 Member Data Documentation

7.57.3.1 bc_facet_forces

```
VecXT<Vec3d> netdem::Membrane::bc_facet_forces
```

7.57.3.2 bc_facet_pressure

```
VecXT<double> netdem::Membrane::bc_facet_pressure
```

7.57.3.3 bc_nodal_vels

```
VecXT<VecNT<double, 6> > netdem::Membrane::bc_nodal_vels
```

7.57.3.4 center

```
Vec3d netdem::Membrane::center {0, 0, 0}
```

7.57.3.5 damp_coef

```
double netdem::Membrane::damp_coef {0.7}
```

7.57.3.6 density

```
double netdem::Membrane::density {500.0}
```

7.57.3.7 elemental_stress

```
VecXT<Vec3d> netdem::Membrane::elemental_stress
```

7.57.3.8 elements

```
VecXT<Vec3i> netdem::Membrane::elements
```

7.57.3.9 height

```
double netdem::Membrane::height {1.0}
```

7.57.3.10 mesh_size

```
double netdem::Membrane::mesh_size {0.1}
```

7.57.3.11 neo_k

```
double netdem::Membrane::neo_k {6.94e5}
```

7.57.3.12 neo_mu

```
double netdem::Membrane::neo_mu {5.21e5}
```

7.57.3.13 nodal_forces_ext

```
VecXT<Vec3d> netdem::Membrane::nodal_forces_ext  [protected]
```

7.57.3.14 nodal_forces_int

```
VecXT<Vec3d> netdem::Membrane::nodal_forces_int  [protected]
```

7.57.3.15 nodal_vels

```
VecXT<Vec3d> netdem::Membrane::nodal_vels
```

7.57.3.16 nodal_vols

```
VecXT<double> netdem::Membrane::nodal_vols
```

7.57.3.17 nodes

```
VecXT<Vec3d> netdem::Membrane::nodes
```

7.57.3.18 radius

```
double netdem::Membrane::radius {0.25}
```

7.57.3.19 ref_ele_area

```
double netdem::Membrane::ref_ele_area {0.05}  [protected]
```

7.57.3.20 ref_ele_height

```
double netdem::Membrane::ref_ele_height {0.1}  [protected]
```

7.57.3.21 ref_ele_width

```
double netdem::Membrane::ref_ele_width {0.1}  [protected]
```

7.57.3.22 thickness

```
double netdem::Membrane::thickness {0.3e-3}
```

7.57.3.23 timestep

```
double netdem::Membrane::timestep {1.0e-4}
```

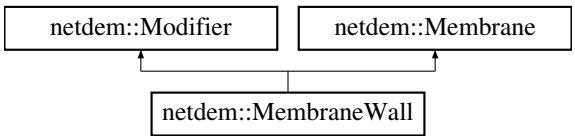
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/membrane.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/membrane.cpp

7.58 netdem::MembraneWall Class Reference

`#include <membrane_wall.hpp>`

Inheritance diagram for netdem::MembraneWall:



Public Member Functions

- MembraneWall ()
- MembraneWall (double radius, double height)
- MembraneWall (double radius, double height, double mesh_size)
- MembraneWall (double radius, double height, double mesh_size, double center_x, double center_y, double center_z)
- void SetRootPath (std::string const &root_path)
- void SetDataType (std::string const &data_type)
- void SetFrequency (bool save_by_cycles, double interval)
- void SetDimensions (double r, double h)
- Modifier * Clone () const override
- void Init (Simulation *sim) override
- void SetPressure (double pressure)
- void Execute () override

Public Attributes

- bool enable_deformation {false}
- bool dump_info {true}
- VecXT< Wall * > wall_list

Private Member Functions

- void UpdateBCForceFromDEM ()
- std::string GetFilename ()
- bool CheckIfToSave ()

Private Attributes

- std::string root_path {"tmp/out/"}
- std::string data_type {"vtk"}
- bool save_by_cycles {true}
- int cycle_interval {0}
- int cycle_previous {0}
- double time_interval {0}
- double time_previous {0}

Additional Inherited Members

7.58.1 Constructor & Destructor Documentation

7.58.1.1 MembraneWall() [1/4]

```
MembraneWall::MembraneWall ( )
```

7.58.1.2 MembraneWall() [2/4]

```
MembraneWall::MembraneWall (
            double radius,
            double height )
```

7.58.1.3 MembraneWall() [3/4]

```
MembraneWall::MembraneWall (
            double radius,
            double height,
            double mesh_size )
```

7.58.1.4 MembraneWall() [4/4]

```
MembraneWall::MembraneWall (
            double radius,
            double height,
            double mesh_size,
            double center_x,
            double center_y,
            double center_z )
```

7.58.2 Member Function Documentation

7.58.2.1 CheckIfToSave()

```
bool MembraneWall::CheckIfToSave ( )  [private]
```

7.58.2.2 Clone()

```
Modifier * MembraneWall::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.58.2.3 Execute()

```
void MembraneWall::Execute ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.58.2.4 GetFilename()

```
string MembraneWall::GetFilename ( )  [private]
```

7.58.2.5 Init()

```
void MembraneWall::Init (
            Simulation * sim )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.58.2.6 SetDataType()

```
void MembraneWall::SetDataType (
            std::string const & data_type )
```

7.58.2.7 SetDimensions()

```
void MembraneWall::SetDimensions (
            double r,
            double h )
```

7.58.2.8 SetFrequency()

```
void MembraneWall::SetFrequency (
            bool save_by_cycles,
            double interval )
```

7.58.2.9 SetPressure()

```
void MembraneWall::SetPressure (
            double pressure )
```

7.58.2.10 SetRootPath()

```
void MembraneWall::SetRootPath (
            std::string const & root_path )
```

7.58.2.11 UpdateBCForceFromDEM()

```
void MembraneWall::UpdateBCForceFromDEM ( )  [private]
```

7.58.3 Member Data Documentation

7.58.3.1 cycle_interval

```
int netdem::MembraneWall::cycle_interval {0}  [private]
```

7.58.3.2 cycle_previous

```
int netdem::MembraneWall::cycle_previous {0}  [private]
```

7.58.3.3 data_type

```
std::string netdem::MembraneWall::data_type {"vtk"}  [private]
```

7.58.3.4 dump_info

```
bool netdem::MembraneWall::dump_info {true}
```

7.58.3.5 enable_deformation

```
bool netdem::MembraneWall::enable_deformation {false}
```

7.58.3.6 root_path

```
std::string netdem::MembraneWall::root_path {"tmp/out/"}  [private]
```

7.58.3.7 save_by_cycles

```
bool netdem::MembraneWall::save_by_cycles {true}  [private]
```

7.58.3.8 time_interval

```
double netdem::MembraneWall::time_interval {0}  [private]
```

7.58.3.9 time_previous

```
double netdem::MembraneWall::time_previous {0}  [private]
```

7.58.3.10 wall_list

```
VecXT<Wall *> netdem::MembraneWall::wall_list
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/membrane_wall.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/membrane_wall.cpp

7.59 netdem::MiniMap< T_key, T_val > Class Template Reference

`#include <mini_map.hpp>`

Public Member Functions

- const my_pair< T_key, T_val > * begin () const
- my_pair< T_key, T_val > * begin ()
- const my_pair< T_key, T_val > * end () const
- my_pair< T_key, T_val > * end ()
- void erase (my_pair< T_key, T_val > *it)
- void erase (const T_key &key)
- const my_pair< T_key, T_val > * find (const T_key &key) const
- my_pair< T_key, T_val > * find (const T_key &key)
- int size () const
- int size ()
- const T_val & operator[ ] (const T_key &key) const
- T_val & operator[ ] (const T_key &key)
- void clear ()

Private Attributes

- VecXT< my_pair< T_key, T_val > > pair_list

7.59.1 Member Function Documentation

7.59.1.1 begin() [1/2]

```
template<typename T_key , typename T_val >
my_pair< T_key, T_val > * netdem::MiniMap< T_key, T_val >::begin ( )  [inline]
```

7.59.1.2 begin() [2/2]

```
template<typename T_key , typename T_val >
const my_pair< T_key, T_val > * netdem::MiniMap< T_key, T_val >::begin ( ) const  [inline]
```

7.59.1.3 clear()

```
template<typename T_key , typename T_val >
void netdem::MiniMap< T_key, T_val >::clear ( )  [inline]
```

7.59.1.4 end() [1/2]

```
template<typename T_key , typename T_val >
my_pair< T_key, T_val > * netdem::MiniMap< T_key, T_val >::end ( )  [inline]
```

7.59.1.5 end() [2/2]

```
template<typename T_key , typename T_val >
const my_pair< T_key, T_val > * netdem::MiniMap< T_key, T_val >::end ( ) const  [inline]
```

7.59.1.6 erase() [1/2]

```
template<typename T_key , typename T_val >
void netdem::MiniMap< T_key, T_val >::erase (
            const T_key & key )  [inline]
```

7.59.1.7 erase() [2/2]

```
template<typename T_key , typename T_val >
void netdem::MiniMap< T_key, T_val >::erase (
            my_pair< T_key, T_val > * it )  [inline]
```

7.59.1.8 find() [1/2]

```
template<typename T_key , typename T_val >
my_pair< T_key, T_val > * netdem::MiniMap< T_key, T_val >::find (
            const T_key & key )  [inline]
```

7.59.1.9 find() [2/2]

```
template<typename T_key , typename T_val >
const my_pair< T_key, T_val > * netdem::MiniMap< T_key, T_val >::find (
            const T_key & key ) const  [inline]
```

7.59.1.10 operator[]() [1/2]

```
template<typename T_key , typename T_val >
T_val & netdem::MiniMap< T_key, T_val >::operator[] (
            const T_key & key )  [inline]
```

7.59.1.11 operator[]() [2/2]

```
template<typename T_key , typename T_val >
const T_val & netdem::MiniMap< T_key, T_val >::operator[] (
            const T_key & key ) const  [inline]
```

7.59.1.12 size() [1/2]

```
template<typename T_key , typename T_val >
int netdem::MiniMap< T_key, T_val >::size ( )  [inline]
```

7.59.1.13 size() [2/2]

```
template<typename T_key , typename T_val >
int netdem::MiniMap< T_key, T_val >::size ( ) const  [inline]
```

7.59.2 Member Data Documentation

7.59.2.1 pair_list

```
template<typename T_key , typename T_val >
VecXT<my_pair<T_key, T_val> > netdem::MiniMap< T_key, T_val >::pair_list  [private]
```

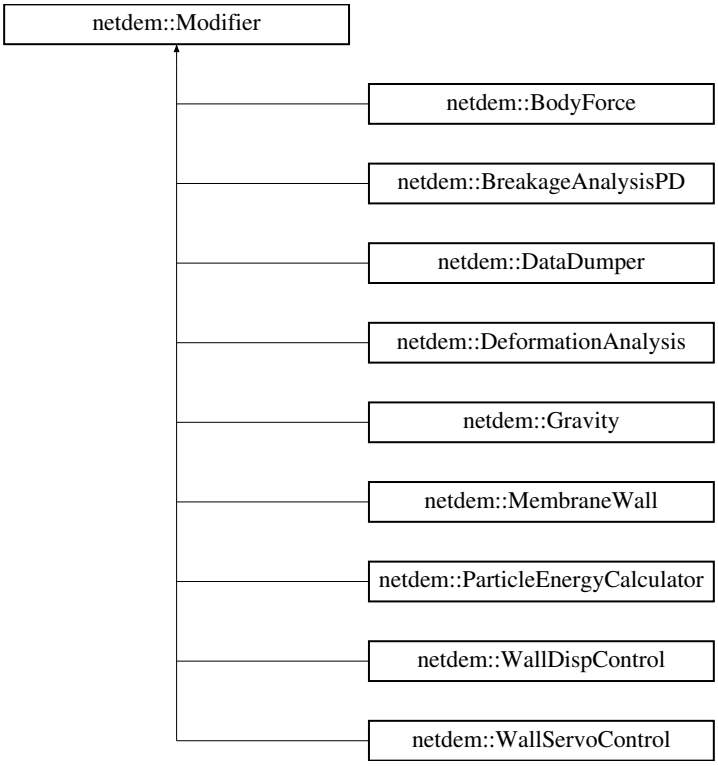
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/mini_map.hpp

7.60 netdem::Modifier Class Reference

`#include <modifier.hpp>`

Inheritance diagram for netdem::Modifier:



Public Member Functions

- Modifier ()
- virtual Modifier * Clone () const
- virtual void Init (Simulation *sim)
- virtual void Enable ()
- virtual void Execute ()
- virtual void Update ()
- virtual ~Modifier ()

Public Attributes

- std::string label {"default"}
- DEMSolver::CyclePoint cycle_point {DEMSolver::CyclePoint::pre}
- Simulation * sim {nullptr}
- Scene * scene {nullptr}
- bool update_with_scene {false}

7.60.1 Detailed Description

The modifier class is an interface to add features to a dem simulation. For example, the effects of gravity, data dumping, scene rendering, etc, are not hard coded in the DEM calculation. They can be incoorporated into a simulation as decorations.

7.60.2 Constructor & Destructor Documentation

7.60.2.1 Modifier()

```
netdem::Modifier::Modifier ( )
```

7.60.2.2 ~Modifier()

```
netdem::Modifier::~Modifier ( )  [virtual]
```

7.60.3 Member Function Documentation

7.60.3.1 Clone()

```
Modifier * netdem::Modifier::Clone ( ) const  [virtual]
```

Reimplemented in netdem::BodyForce, netdem::BreakageAnalysisPD, netdem::DataDumper, netdem::DeformationAnalysis, netdem::Gravity, netdem::MembraneWall, netdem::ParticleEnergyCalculator, netdem::WallDispControl, and netdem::WallServoControl.

7.60.3.2 Enable()

```
void netdem::Modifier::Enable ( )  [virtual]
```

7.60.3.3 Execute()

```
void netdem::Modifier::Execute ( )  [virtual]
```

Reimplemented in netdem::BodyForce, netdem::BreakageAnalysisPD, netdem::DataDumper, netdem::DeformationAnalysis, netdem::Gravity, netdem::MembraneWall, netdem::ParticleEnergyCalculator, netdem::WallDispControl, and netdem::WallServoControl.

7.60.3.4 Init()

```
void netdem::Modifier::Init (
            Simulation * sim )  [virtual]
```

Reimplemented in netdem::BreakageAnalysisPD, netdem::DataDumper, netdem::DeformationAnalysis, netdem::Gravity, and netdem::MembraneWall.

7.60.3.5 Update()

```
void netdem::Modifier::Update ( )  [virtual]
```

Reimplemented in netdem::BreakageAnalysisPD, netdem::DeformationAnalysis, netdem::BodyForce, netdem::ParticleEnergyCalcula netdem::WallDispControl, and netdem::WallServoControl.

7.60.4 Member Data Documentation

7.60.4.1 cycle_point

```
DEMSolver::CyclePoint netdem::Modifier::cycle_point {DEMSolver::CyclePoint::pre}
```

7.60.4.2 label

```
std::string netdem::Modifier::label {"default"}
```

7.60.4.3 scene

```
Scene* netdem::Modifier::scene {nullptr}
```

7.60.4.4 sim

```
Simulation* netdem::Modifier::sim {nullptr}
```

7.60.4.5 update_with_scene

```
bool netdem::Modifier::update_with_scene {false}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/modifier.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/modifier.cpp

7.61 netdem::ModifierManager Class Reference

```
#include <modifier_manager.hpp>
```

Public Member Functions

- ModifierManager ()
- void Init (Simulation ∗s)
- Modifier ∗ Insert (Modifier ∗e)

  insert and remove modifiers
- void RemoveModifier (std::string const &label)
- void Enable (std::string const &label)

  activate and deactivate modifiers
- void Disable (std::string const &label)
- void Enable (Modifier ∗const e)
- void Disable (Modifier ∗const e)
- Modifier ∗ FindModifier (std::string const &label)

  find modifier
- bool FindModifier (Modifier ∗const e)
- void UpdateModifiers ()

  update the modifiers in the scene_state_subscribers when scene changes
- void ExecuteModifiers (DEMSolver::CyclePoint cycle_point)

  excute during DEM cycling
- ∼ModifierManager ()

Public Attributes

- std::unordered_map< std::string, Modifier ∗ > modifier_lib
- VecXT< std::unordered_set< Modifier ∗ > > modifier_list
- std::unordered_set< Modifier ∗ > scene_state_subscribers

Private Attributes

- Simulation ∗ sim {nullptr}

7.61.1 Detailed Description

Manages the modifiers in a simulation. Note that an modifier would take effects only if it is activated (i.e., it is inserted into pre_modifier_list or post_modifier_list).

- modifier_lib: stores the list of defined modifiers.
- modifier_list: stores the list of activated modifiers.
- scene_state_subscribers: stores the list of modifiers, whose properties need to update if the scene changes. For example, a body force modifier is defined for a certain particle. If this particle get out of the domain (i.e., the scene does not contain this particle any more), this body force modifier needs to be updated so that no more particles is in the particle_list.

7.61.2 Constructor & Destructor Documentation

7.61.2.1 ModifierManager()

```
netdem::ModifierManager::ModifierManager ( )
```

7.61.2.2 ∼ModifierManager()

```
netdem::ModifierManager::∼ModifierManager ( )
```

7.61.3 Member Function Documentation

7.61.3.1 Disable() [1/2]

```
void netdem::ModifierManager::Disable (
            Modifier *const e )
```

7.61.3.2 Disable() [2/2]

```
void netdem::ModifierManager::Disable (
            std::string const & label )
```

7.61.3.3 Enable() [1/2]

```
void netdem::ModifierManager::Enable (
            Modifier *const e )
```

7.61.3.4 Enable() [2/2]

```
void netdem::ModifierManager::Enable (
            std::string const & label )
```

activate and deactivate modifiers

7.61.3.5 ExecuteModifiers()

```
void netdem::ModifierManager::ExecuteModifiers (
            DEMSolver::CyclePoint cycle_point )
```

excute during DEM cycling

7.61.3.6 FindModifier() [1/2]

```
bool netdem::ModifierManager::FindModifier (
            Modifier *const e )
```

7.61.3.7 FindModifier() [2/2]

```
Modifier * netdem::ModifierManager::FindModifier (
            std::string const & label )
```

find modifier

7.61.3.8 Init()

```
void netdem::ModifierManager::Init (
            Simulation * s )
```

7.61.3.9 Insert()

```
Modifier * netdem::ModifierManager::Insert (
            Modifier * e )
```

insert and remove modifiers

7.61.3.10 RemoveModifier()

```
void netdem::ModifierManager::RemoveModifier (
            std::string const & label )
```

7.61.3.11 UpdateModifiers()

```
void netdem::ModifierManager::UpdateModifiers ( )
```

update the modifiers in the scene_state_subscribers when scene changes

7.61.4 Member Data Documentation

7.61.4.1 modifier_lib

```
std::unordered_map<std::string, Modifier *> netdem::ModifierManager::modifier_lib
```

7.61.4.2 modifier_list

```
VecXT<std::unordered_set<Modifier *> > netdem::ModifierManager::modifier_list
```

7.61.4.3 scene_state_subscribers

```
std::unordered_set<Modifier *> netdem::ModifierManager::scene_state_subscribers
```

7.61.4.4 sim

```
Simulation* netdem::ModifierManager::sim {nullptr}  [private]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/modifier_manager.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/modifier_manager.cpp

7.62 netdem::MPIDataDefine Class Reference

```
#include <mpi_data_def.hpp>
```

Public Member Functions

- void Init ()

Public Attributes

- MPI_Datatype particle_datatype
- MPI_Datatype bond_entry_datatype
- MPI_Datatype collision_entry_datatype
- MPI_Datatype contact_pp_datatype
- MPI_Datatype contact_pw_datatype

7.62.1 Member Function Documentation

7.62.1.1 Init()

```
void netdem::MPIDataDefine::Init ( )  [inline]
```

7.62.2 Member Data Documentation

7.62.2.1 bond_entry_datatype

```
MPI_Datatype netdem::MPIDataDefine::bond_entry_datatype
```

7.62.2.2 collision_entry_datatype

```
MPI_Datatype netdem::MPIDataDefine::collision_entry_datatype
```

7.62.2.3 contact_pp_datatype

```
MPI_Datatype netdem::MPIDataDefine::contact_pp_datatype
```

7.62.2.4 contact_pw_datatype

```
MPI_Datatype netdem::MPIDataDefine::contact_pw_datatype
```

7.62.2.5 particle_datatype

```
MPI_Datatype netdem::MPIDataDefine::particle_datatype
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/mpi_data_def.hpp

7.63 netdem::MPIManager Class Reference

```
#include <mpi_manager.hpp>
```

Public Member Functions

- MPIManager ()
- void Init (Simulation *sim)
- void CommitMPIDataType ()
- void BuildContactRef ()
- void CleanUpParticleProxy ()
- void ExchangeDataTransfer ()
- void ExchangeDataProxy ()
- void ExchangeDataBack ()
- void CleanUpParticleGhost ()
- void ClearContactRef ()
- void GatherDataProxy ()
- void GatherDataBack ()
- void GatherDataTransfer ()
- void SendDataProxy ()
- void SendDataBack ()

- void SendDataTransfer ()
- void RecvDataProxy ()
- void RecvDataBack ()
- void RecvDataTransfer ()
- void MergeParticleProxy (int source_rank)
- void MergeContactPPProxy (int source_rank)
- void MergeContactPWProxy (int source_rank)
- void MergeContactPPBack (int source_rank)
- void MergeContactPWBack (int source_rank)
- void MergeShapeTransfer (int source_rank)
- void MergeParticleTransfer (int source_rank)
- void MergeContactPPTransfer (int source_rank)
- void MergeContactPWTransfer (int source_rank)

Public Attributes

- MPIDataDefine mpi_data_def

  defines the data structure of the classes to transfer
- int my_rank

  self rank and total number of processors
- int num_procs
- VecXT< Shape * > shape_transfer_out_list
- VecXT< VecXT< Particle * > > particle_proxy_out_list

  particles that overlaps with other sub-domains
- VecXT< VecXT< ContactPP * > > contact_pp_proxy_out_list

  particle-particle contacts of the particle proxies
- VecXT< VecXT< BondEntry * > > bond_entry_pp_proxy_out_list
- VecXT< VecXT< CollisionEntry * > > collision_entry_pp_proxy_out_list
- VecXT< VecXT< ContactPW * > > contact_pw_proxy_out_list

  particle-wall contacts of the particle proxies
- VecXT< VecXT< BondEntry * > > bond_entry_pw_proxy_out_list
- VecXT< VecXT< CollisionEntry * > > collision_entry_pw_proxy_out_list
- VecXT< VecXT< Particle * > > particle_proxy_in_list

  particles transfered from other sub-domains as proxies
- VecXT< VecXT< ContactPP * > > contact_pp_back_out_list

  particle-particle contacts that need to send back to its original domain
- VecXT< VecXT< BondEntry * > > bond_entry_pp_back_out_list
- VecXT< VecXT< CollisionEntry * > > collision_entry_pp_back_out_list
- VecXT< VecXT< ContactPW * > > contact_pw_back_out_list

  particle-wall contacts that need to send back to its original domain
- VecXT< VecXT< BondEntry * > > bond_entry_pw_back_out_list
- VecXT< VecXT< CollisionEntry * > > collision_entry_pw_back_out_list
- VecXT< VecXT< Particle * > > particle_transfer_out_list

  particles that transfer out of the domain
- VecXT< VecXT< ContactPP * > > contact_pp_transfer_out_list

  particle-particle contacts of the particles to be transferred
- VecXT< VecXT< BondEntry * > > bond_entry_pp_transfer_out_list
- VecXT< VecXT< CollisionEntry * > > collision_entry_pp_transfer_out_list
- VecXT< VecXT< ContactPW * > > contact_pw_transfer_out_list

  particle-wall contacts of the particles to be transferred
- VecXT< VecXT< BondEntry * > > bond_entry_pw_transfer_out_list
- VecXT< VecXT< CollisionEntry * > > collision_entry_pw_transfer_out_list

Private Member Functions

- std::list< int > GetRankList ()
- void RemoveParticle (int id, VecXT< Particle * > *p_list)
- void ClearBuffer ()

Private Attributes

- Simulation * sim {nullptr}
- VecXT< ParticleData * > particle_data_list_send
- VecXT< ContactPPData * > contact_pp_data_list_send
- VecXT< BondEntryData * > bond_entry_pp_data_list_send
- VecXT< CollisionEntryData * > collision_entry_pp_data_list_send
- VecXT< ContactPWData * > contact_pw_data_list_send
- VecXT< BondEntryData * > bond_entry_pw_data_list_send
- VecXT< CollisionEntryData * > collision_entry_pw_data_list_send
- VecXT< int > particle_num_list_send
- VecXT< int > contact_pp_num_list_send
- VecXT< int > bond_entry_pp_num_list_send
- VecXT< int > collision_entry_pp_num_list_send
- VecXT< int > contact_pw_num_list_send
- VecXT< int > bond_entry_pw_num_list_send
- VecXT< int > collision_entry_pw_num_list_send
- VecXT< MPI_Request > particle_req_list_send
- VecXT< MPI_Request > contact_pp_req_list_send
- VecXT< MPI_Request > bond_entry_pp_req_list_send
- VecXT< MPI_Request > collision_entry_pp_req_list_send
- VecXT< MPI_Request > contact_pw_req_list_send
- VecXT< MPI_Request > bond_entry_pw_req_list_send
- VecXT< MPI_Request > collision_entry_pw_req_list_send
- std::string * shape_data_send {nullptr}
- VecXT< MPI_Request > shape_req_list_send
- VecXT< std::string > shape_data_list_recv
- VecXT< MPI_Request > shape_req_list_recv
- VecXT< bool > shape_probed_list
- VecXT< ParticleData * > particle_data_list_recv
- VecXT< ContactPPData * > contact_pp_data_list_recv
- VecXT< BondEntryData * > bond_entry_pp_data_list_recv
- VecXT< CollisionEntryData * > collision_entry_pp_data_list_recv
- VecXT< ContactPWData * > contact_pw_data_list_recv
- VecXT< BondEntryData * > bond_entry_pw_data_list_recv
- VecXT< CollisionEntryData * > collision_entry_pw_data_list_recv
- VecXT< int > particle_num_list_recv
- VecXT< int > contact_pp_num_list_recv
- VecXT< int > bond_entry_pp_num_list_recv
- VecXT< int > collision_entry_pp_num_list_recv
- VecXT< int > contact_pw_num_list_recv
- VecXT< int > bond_entry_pw_num_list_recv
- VecXT< int > collision_entry_pw_num_list_recv
- VecXT< MPI_Request > particle_req_list_recv
- VecXT< MPI_Request > contact_pp_req_list_recv
- VecXT< MPI_Request > bond_entry_pp_req_list_recv
- VecXT< MPI_Request > collision_entry_pp_req_list_recv
- VecXT< MPI_Request > contact_pw_req_list_recv

- VecXT< MPI_Request > bond_entry_pw_req_list_recv
- VecXT< MPI_Request > collision_entry_pw_req_list_recv
- VecXT< bool > particle_probed_list
- VecXT< bool > contact_pp_probed_list
- VecXT< bool > bond_entry_pp_probed_list
- VecXT< bool > collision_entry_pp_probed_list
- VecXT< bool > contact_pw_probed_list
- VecXT< bool > bond_entry_pw_probed_list
- VecXT< bool > collision_entry_pw_probed_list

7.63.1 Detailed Description

Manage the data exchange in a DEM simulation. Basicly, a DEM cycle would involve three rounds of data exchange.

- round one: exchange particles as proxies and their corresponding contacts. A partical will be sent as a proxy as its surface exceeds the domain boundary (such that it could potentially contact with particles in other domains). If a particle needs to be sent to other domains as a proxy, its contacts need also to be sent.
- round two: exchange the contacts of particle proxies back to its original domain, so that the contact forces could be applied onto the original particles.
- round three: excange particles and their corresponding contacts if particle get out of the domain (i.e., the particle turn to belong to another domain after updating its positions at the end of a DEM cycle). Terminologies:
- particle proxy or contact proxy: particle belongs to one domain, whereas if it overlaps with other domain, it needs to seed out as a proxy for contact detection and resolution. Thus, proxy means a real object but not belongs to one domain.
- particle ghost: if a particle is gone (e.g., removed or migirated to another domain), it will left a ghost copy of itself (to prevent the nullptr particle in the contacts). Thus, ghost means a virtual object.
- Note: proxy and ghost may excange, which has facilitated the programing and computational efficiency.

7.63.2 Constructor & Destructor Documentation

7.63.2.1 MPIManager()

```
netdem::MPIManager::MPIManager ( )
```

7.63.3 Member Function Documentation

7.63.3.1 BuildContactRef()

```
void netdem::MPIManager::BuildContactRef ( )
```

7.63.3.2 CleanUpParticleGhost()

```
void netdem::MPIManager::CleanUpParticleGhost ( )
```

7.63.3.3 CleanUpParticleProxy()

```
void netdem::MPIManager::CleanUpParticleProxy ( )
```

7.63.3.4 ClearBuffer()

```
void netdem::MPIManager::ClearBuffer ( )  [private]
```

7.63.3.5 ClearContactRef()

```
void netdem::MPIManager::ClearContactRef ( )
```

7.63.3.6 CommitMPIDataType()

```
void netdem::MPIManager::CommitMPIDataType ( )
```

7.63.3.7 ExchangeDataBack()

```
void netdem::MPIManager::ExchangeDataBack ( )
```

7.63.3.8 ExchangeDataProxy()

```
void netdem::MPIManager::ExchangeDataProxy ( )
```

7.63.3.9 ExchangeDataTransfer()

```
void netdem::MPIManager::ExchangeDataTransfer ( )
```

7.63.3.10 GatherDataBack()

```
void netdem::MPIManager::GatherDataBack ( )
```

7.63.3.11 GatherDataProxy()

```
void netdem::MPIManager::GatherDataProxy ( )
```

7.63.3.12 GatherDataTransfer()

```
void netdem::MPIManager::GatherDataTransfer ( )
```

7.63.3.13 GetRankList()

```
list< int > netdem::MPIManager::GetRankList ( )  [private]
```

7.63.3.14 Init()

```
void netdem::MPIManager::Init (
            Simulation * sim )
```

7.63.3.15 MergeContactPPBack()

```
void netdem::MPIManager::MergeContactPPBack (
            int source_rank )
```

7.63.3.16 MergeContactPPProxy()

```
void netdem::MPIManager::MergeContactPPProxy (
            int source_rank )
```

7.63.3.17 MergeContactPPTransfer()

```
void netdem::MPIManager::MergeContactPPTransfer (
            int source_rank )
```

7.63.3.18 MergeContactPWBack()

```
void netdem::MPIManager::MergeContactPWBack (
            int source_rank )
```

7.63.3.19 MergeContactPWProxy()

```
void netdem::MPIManager::MergeContactPWProxy (
            int source_rank )
```

7.63.3.20 MergeContactPWTransfer()

```
void netdem::MPIManager::MergeContactPWTransfer (
            int source_rank )
```

7.63.3.21 MergeParticleProxy()

```
void netdem::MPIManager::MergeParticleProxy (
            int source_rank )
```

7.63.3.22 MergeParticleTransfer()

```
void netdem::MPIManager::MergeParticleTransfer (
            int source_rank )
```

7.63.3.23 MergeShapeTransfer()

```
void netdem::MPIManager::MergeShapeTransfer (
            int source_rank )
```

7.63.3.24 RecvDataBack()

```
void netdem::MPIManager::RecvDataBack ( )
```

7.63.3.25 RecvDataProxy()

```
void netdem::MPIManager::RecvDataProxy ( )
```

7.63.3.26 RecvDataTransfer()

```
void netdem::MPIManager::RecvDataTransfer ( )
```

7.63.3.27 RemoveParticle()

```
void netdem::MPIManager::RemoveParticle (
            int id,
            VecXT< Particle * > * p_list )  [private]
```

7.63.3.28 SendDataBack()

```
void netdem::MPIManager::SendDataBack ( )
```

7.63.3.29 SendDataProxy()

```
void netdem::MPIManager::SendDataProxy ( )
```

7.63.3.30 SendDataTransfer()

```
void netdem::MPIManager::SendDataTransfer ( )
```

7.63.4 Member Data Documentation

7.63.4.1 bond_entry_pp_back_out_list

VecXT<VecXT<BondEntry *> > netdem::MPIManager::bond_entry_pp_back_out_list

7.63.4.2 bond_entry_pp_data_list_recv

VecXT<BondEntryData *> netdem::MPIManager::bond_entry_pp_data_list_recv [private]

7.63.4.3 bond_entry_pp_data_list_send

VecXT<BondEntryData *> netdem::MPIManager::bond_entry_pp_data_list_send [private]

7.63.4.4 bond_entry_pp_num_list_recv

VecXT<int> netdem::MPIManager::bond_entry_pp_num_list_recv [private]

7.63.4.5 bond_entry_pp_num_list_send

VecXT<int> netdem::MPIManager::bond_entry_pp_num_list_send [private]

7.63.4.6 bond_entry_pp_probed_list

VecXT<bool> netdem::MPIManager::bond_entry_pp_probed_list [private]

7.63.4.7 bond_entry_pp_proxy_out_list

VecXT<VecXT<BondEntry *> > netdem::MPIManager::bond_entry_pp_proxy_out_list

7.63.4.8 bond_entry_pp_req_list_recv

VecXT<MPI_Request> netdem::MPIManager::bond_entry_pp_req_list_recv [private]

7.63.4.9 bond_entry_pp_req_list_send

VecXT<MPI_Request> netdem::MPIManager::bond_entry_pp_req_list_send [private]

7.63.4.10 bond_entry_pp_transfer_out_list

VecXT<VecXT<BondEntry *> > netdem::MPIManager::bond_entry_pp_transfer_out_list

7.63.4.11 bond_entry_pw_back_out_list

VecXT<VecXT<BondEntry *> > netdem::MPIManager::bond_entry_pw_back_out_list

7.63.4.12 bond_entry_pw_data_list_recv

VecXT<BondEntryData *> netdem::MPIManager::bond_entry_pw_data_list_recv [private]

7.63.4.13 bond_entry_pw_data_list_send

VecXT<BondEntryData *> netdem::MPIManager::bond_entry_pw_data_list_send [private]

7.63.4.14 bond_entry_pw_num_list_recv

VecXT<int> netdem::MPIManager::bond_entry_pw_num_list_recv [private]

7.63.4.15 bond_entry_pw_num_list_send

VecXT<int> netdem::MPIManager::bond_entry_pw_num_list_send [private]

7.63.4.16 bond_entry_pw_probed_list

VecXT<bool> netdem::MPIManager::bond_entry_pw_probed_list [private]

7.63.4.17 bond_entry_pw_proxy_out_list

VecXT<VecXT<BondEntry *> > netdem::MPIManager::bond_entry_pw_proxy_out_list

7.63.4.18 bond_entry_pw_req_list_recv

VecXT<MPI_Request> netdem::MPIManager::bond_entry_pw_req_list_recv [private]

7.63.4.19 bond_entry_pw_req_list_send

VecXT<MPI_Request> netdem::MPIManager::bond_entry_pw_req_list_send [private]

7.63.4.20 bond_entry_pw_transfer_out_list

VecXT<VecXT<BondEntry *> > netdem::MPIManager::bond_entry_pw_transfer_out_list

7.63.4.21 collision_entry_pp_back_out_list

VecXT<VecXT<CollisionEntry *> > netdem::MPIManager::collision_entry_pp_back_out_list

7.63.4.22 collision_entry_pp_data_list_recv

VecXT<CollisionEntryData *> netdem::MPIManager::collision_entry_pp_data_list_recv [private]

7.63.4.23 collision_entry_pp_data_list_send

VecXT<CollisionEntryData *> netdem::MPIManager::collision_entry_pp_data_list_send [private]

7.63.4.24 collision_entry_pp_num_list_recv

VecXT<int> netdem::MPIManager::collision_entry_pp_num_list_recv [private]

7.63.4.25 collision_entry_pp_num_list_send

VecXT<int> netdem::MPIManager::collision_entry_pp_num_list_send [private]

7.63.4.26 collision_entry_pp_probed_list

VecXT<bool> netdem::MPIManager::collision_entry_pp_probed_list [private]

7.63.4.27 collision_entry_pp_proxy_out_list

VecXT<VecXT<CollisionEntry *> > netdem::MPIManager::collision_entry_pp_proxy_out_list

7.63.4.28 collision_entry_pp_req_list_recv

VecXT<MPI_Request> netdem::MPIManager::collision_entry_pp_req_list_recv [private]

7.63.4.29 collision_entry_pp_req_list_send

VecXT<MPI_Request> netdem::MPIManager::collision_entry_pp_req_list_send [private]

7.63.4.30 collision_entry_pp_transfer_out_list

VecXT<VecXT<CollisionEntry *> > netdem::MPIManager::collision_entry_pp_transfer_out_list

7.63.4.31 collision_entry_pw_back_out_list

VecXT<VecXT<CollisionEntry *> > netdem::MPIManager::collision_entry_pw_back_out_list

7.63.4.32 collision_entry_pw_data_list_recv

VecXT<CollisionEntryData *> netdem::MPIManager::collision_entry_pw_data_list_recv [private]

7.63.4.33 collision_entry_pw_data_list_send

VecXT<CollisionEntryData *> netdem::MPIManager::collision_entry_pw_data_list_send [private]

7.63.4.34 collision_entry_pw_num_list_recv

VecXT<int> netdem::MPIManager::collision_entry_pw_num_list_recv [private]

7.63.4.35 collision_entry_pw_num_list_send

VecXT<int> netdem::MPIManager::collision_entry_pw_num_list_send [private]

7.63.4.36 collision_entry_pw_probed_list

VecXT<bool> netdem::MPIManager::collision_entry_pw_probed_list [private]

7.63.4.37 collision_entry_pw_proxy_out_list

VecXT<VecXT<CollisionEntry *> > netdem::MPIManager::collision_entry_pw_proxy_out_list

7.63.4.38 collision_entry_pw_req_list_recv

VecXT<MPI_Request> netdem::MPIManager::collision_entry_pw_req_list_recv [private]

7.63.4.39 collision_entry_pw_req_list_send

VecXT<MPI_Request> netdem::MPIManager::collision_entry_pw_req_list_send [private]

7.63.4.40 collision_entry_pw_transfer_out_list

VecXT<VecXT<CollisionEntry *> > netdem::MPIManager::collision_entry_pw_transfer_out_list

7.63.4.41 contact_pp_back_out_list

VecXT<VecXT<ContactPP *> > netdem::MPIManager::contact_pp_back_out_list

particle-particle contacts that need to send back to its original domain

7.63.4.42 contact_pp_data_list_recv

VecXT<ContactPPData *> netdem::MPIManager::contact_pp_data_list_recv [private]

7.63.4.43 contact_pp_data_list_send

VecXT<ContactPPData *> netdem::MPIManager::contact_pp_data_list_send [private]

7.63.4.44 contact_pp_num_list_recv

VecXT<int> netdem::MPIManager::contact_pp_num_list_recv [private]

7.63.4.45 contact_pp_num_list_send

VecXT<int> netdem::MPIManager::contact_pp_num_list_send [private]

7.63.4.46 contact_pp_probed_list

VecXT<bool> netdem::MPIManager::contact_pp_probed_list [private]

7.63.4.47 contact_pp_proxy_out_list

VecXT<VecXT<ContactPP *> > netdem::MPIManager::contact_pp_proxy_out_list

particle-particle contacts of the particle proxies

7.63.4.48 contact_pp_req_list_recv

```
VecXT<MPI_Request> netdem::MPIManager::contact_pp_req_list_recv  [private]
```

7.63.4.49 contact_pp_req_list_send

```
VecXT<MPI_Request> netdem::MPIManager::contact_pp_req_list_send  [private]
```

7.63.4.50 contact_pp_transfer_out_list

```
VecXT<VecXT<ContactPP *> > netdem::MPIManager::contact_pp_transfer_out_list
```

particle-particle contacts of the particles to be transferred

7.63.4.51 contact_pw_back_out_list

```
VecXT<VecXT<ContactPW *> > netdem::MPIManager::contact_pw_back_out_list
```

particle-wall contacts that need to send back to its original domain

7.63.4.52 contact_pw_data_list_recv

```
VecXT<ContactPWData *> netdem::MPIManager::contact_pw_data_list_recv  [private]
```

7.63.4.53 contact_pw_data_list_send

```
VecXT<ContactPWData *> netdem::MPIManager::contact_pw_data_list_send  [private]
```

7.63.4.54 contact_pw_num_list_recv

```
VecXT<int> netdem::MPIManager::contact_pw_num_list_recv  [private]
```

7.63.4.55 contact_pw_num_list_send

VecXT<int> netdem::MPIManager::contact_pw_num_list_send [private]

7.63.4.56 contact_pw_probed_list

VecXT<bool> netdem::MPIManager::contact_pw_probed_list [private]

7.63.4.57 contact_pw_proxy_out_list

VecXT<VecXT<ContactPW *> > netdem::MPIManager::contact_pw_proxy_out_list

particle-wall contacts of the particle proxies

7.63.4.58 contact_pw_req_list_recv

VecXT<MPI_Request> netdem::MPIManager::contact_pw_req_list_recv [private]

7.63.4.59 contact_pw_req_list_send

VecXT<MPI_Request> netdem::MPIManager::contact_pw_req_list_send [private]

7.63.4.60 contact_pw_transfer_out_list

VecXT<VecXT<ContactPW *> > netdem::MPIManager::contact_pw_transfer_out_list

particle-wall contacts of the particles to be transferred

7.63.4.61 mpi_data_def

MPIDataDefine netdem::MPIManager::mpi_data_def

defines the data structure of the classes to transfer

7.63.4.62 my_rank

```
int netdem::MPIManager::my_rank
```

self rank and total number of processors

7.63.4.63 num_procs

```
int netdem::MPIManager::num_procs
```

7.63.4.64 particle_data_list_recv

```
VecXT<ParticleData *> netdem::MPIManager::particle_data_list_recv  [private]
```

7.63.4.65 particle_data_list_send

```
VecXT<ParticleData *> netdem::MPIManager::particle_data_list_send  [private]
```

7.63.4.66 particle_num_list_recv

```
VecXT<int> netdem::MPIManager::particle_num_list_recv  [private]
```

7.63.4.67 particle_num_list_send

```
VecXT<int> netdem::MPIManager::particle_num_list_send  [private]
```

7.63.4.68 particle_probed_list

```
VecXT<bool> netdem::MPIManager::particle_probed_list  [private]
```

7.63.4.69 particle_proxy_in_list

```
VecXT<VecXT<Particle *> > netdem::MPIManager::particle_proxy_in_list
```

particles transfered from other sub-domains as proxies

7.63.4.70 particle_proxy_out_list

```
VecXT<VecXT<Particle *> > netdem::MPIManager::particle_proxy_out_list
```

particles that overlaps with other sub-domains

7.63.4.71 particle_req_list_recv

```
VecXT<MPI_Request> netdem::MPIManager::particle_req_list_recv  [private]
```

7.63.4.72 particle_req_list_send

```
VecXT<MPI_Request> netdem::MPIManager::particle_req_list_send  [private]
```

7.63.4.73 particle_transfer_out_list

```
VecXT<VecXT<Particle *> > netdem::MPIManager::particle_transfer_out_list
```

particles that transfer out of the domain

7.63.4.74 shape_data_list_recv

```
VecXT<std::string> netdem::MPIManager::shape_data_list_recv  [private]
```

7.63.4.75 shape_data_send

```
std::string* netdem::MPIManager::shape_data_send {nullptr}  [private]
```

7.63.4.76 shape_probed_list

```
VecXT<bool> netdem::MPIManager::shape_probed_list  [private]
```

7.63.4.77 shape_req_list_recv

```
VecXT<MPI_Request> netdem::MPIManager::shape_req_list_recv  [private]
```

7.63.4.78 shape_req_list_send

```
VecXT<MPI_Request> netdem::MPIManager::shape_req_list_send  [private]
```

7.63.4.79 shape_transfer_out_list

```
VecXT<Shape *> netdem::MPIManager::shape_transfer_out_list
```

using json serilization and de-serilization (i.e., parameters --> json string --> parameters).

7.63.4.80 sim

```
Simulation* netdem::MPIManager::sim {nullptr}  [private]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/mpi_manager.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/mpi_manager.cpp

7.64 netdem::my_pair< T_key, T_val > Struct Template Reference

```
#include <mini_map.hpp>
```

Public Member Functions

- my_pair ()
- my_pair (const T_key &key, const T_key &val)

Public Attributes

- T_key first
- T_val second

7.64.1 Constructor & Destructor Documentation

7.64.1.1 my_pair() [1/2]

```
template<typename T_key , typename T_val >
netdem::my_pair< T_key, T_val >::my_pair ( )  [inline]
```

7.64.1.2 my_pair() [2/2]

```
template<typename T_key , typename T_val >
netdem::my_pair< T_key, T_val >::my_pair (
            const T_key & key,
            const T_key & val )  [inline]
```

7.64.2 Member Data Documentation

7.64.2.1 first

```
template<typename T_key , typename T_val >
T_key netdem::my_pair< T_key, T_val >::first
```

7.64.2.2 second

```
template<typename T_key , typename T_val >
T_val netdem::my_pair< T_key, T_val >::second
```

The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/mini_map.hpp

7.65 netdem::NeighPofP Class Reference

```
#include <contact_pp.hpp>
```

Public Member Functions

- NeighPofP ()
- NeighPofP (Particle ∗const p, int id, ContactPP ∗const cnt)

Public Attributes

- Particle ∗ particle {nullptr}
- int lookup_id {-1}
- ContactPP ∗ contact {nullptr}

7.65.1 Detailed Description

- stores pair info particle: pointer of the neighboring particle lookup_id: index of self in the linked_particle_list of particle contact: pointer of the contact, if is in contact with particle

7.65.2 Constructor & Destructor Documentation

7.65.2.1 NeighPofP() [1/2]

```
netdem::NeighPofP::NeighPofP ( )  [inline]
```

7.65.2.2 NeighPofP() [2/2]

```
netdem::NeighPofP::NeighPofP (
            Particle *const p,
            int id,
            ContactPP *const cnt )  [inline]
```

7.65.3 Member Data Documentation

7.65.3.1 contact

```
ContactPP* netdem::NeighPofP::contact {nullptr}
```

7.65.3.2 lookup_id

```
int netdem::NeighPofP::lookup_id {-1}
```

7.65.3.3 particle

```
Particle* netdem::NeighPofP::particle {nullptr}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/contact_pp.hpp

7.66 netdem::NeighPofW Class Reference

```
#include <contact_pw.hpp>
```

Public Member Functions

- NeighPofW ()
- NeighPofW (Particle *const p, int id, ContactPW *const cnt)

Public Attributes

- Particle * particle {nullptr}
- int lookup_id {-1}
- ContactPW * contact {nullptr}

7.66.1 Detailed Description

- stores pair info particle: pointer of the neighboring particle lookup_id: index of self in the linked_wall_list of particle contact: pointer of the contact, if is in contact with particle

7.66.2 Constructor & Destructor Documentation

7.66.2.1 NeighPofW() [1/2]

```
netdem::NeighPofW::NeighPofW ( )  [inline]
```

7.66.2.2 NeighPofW() [2/2]

```
netdem::NeighPofW::NeighPofW (
            Particle *const p,
            int id,
            ContactPW *const cnt )  [inline]
```

7.66.3 Member Data Documentation

7.66.3.1 contact

```
ContactPW* netdem::NeighPofW::contact {nullptr}
```

7.66.3.2 lookup_id

```
int netdem::NeighPofW::lookup_id {-1}
```

7.66.3.3 particle

```
Particle* netdem::NeighPofW::particle {nullptr}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/contact_pw.hpp

7.67 netdem::NeighWofP Class Reference

```
#include <contact_pp.hpp>
```

Public Member Functions

- NeighWofP ()
- NeighWofP (Wall *const w, int id, ContactPW *const cnt)

Public Attributes

- Wall * wall {nullptr}
- int lookup_id {-1}
- ContactPW * contact {nullptr}

7.67.1 Detailed Description

- stores pair info wall: pointer of the neighboring wall lookup_id: index of self in the linked_particle_list of wall contact: pointer of the contact, if is in contact with wall

7.67.2 Constructor & Destructor Documentation

7.67.2.1 NeighWofP() [1/2]

```
netdem::NeighWofP::NeighWofP ( )  [inline]
```

7.67.2.2 NeighWofP() [2/2]

```
netdem::NeighWofP::NeighWofP (
            Wall *const w,
            int id,
            ContactPW *const cnt )  [inline]
```

7.67.3 Member Data Documentation

7.67.3.1 contact

```
ContactPW* netdem::NeighWofP::contact {nullptr}
```

7.67.3.2 lookup_id

```
int netdem::NeighWofP::lookup_id {-1}
```

7.67.3.3 wall

```
Wall* netdem::NeighWofP::wall {nullptr}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/contact_pp.hpp

7.68 netdem::PackGenerator Class Reference

`#include <gen_pack.hpp>`

Static Public Member Functions

- static VecXT< Particle > GetGridPack (double len_x, double len_y, double len_z, double center_x, double center_y, double center_z, int num_x, int num_y, int num_z, const VecXT< Shape * > &shape_list)
- static VecXT< Particle > GetGridPack (double len_x, double len_y, double len_z, double center_x, double center_y, double center_z, int num_x, int num_y, int num_z, Shape *shape)
- static VecXT< BondedSpheres > GetGridPack (double len_x, double len_y, double len_z, double center_x, double center_y, double center_z, int num_x, int num_y, int num_z, const BondedSpheres &bonded_spheres_template)
- static VecXT< BondedVoronois > GetGridPack (double len_x, double len_y, double len_z, double center_x, double center_y, double center_z, int num_x, int num_y, int num_z, const BondedVoronois &bonded_voronois_template)

7.68.1 Detailed Description

generate a grid of particles. Inputs:

- len_x, len_y, len_z: lenghts of the box in x, y, and z directions.
- center_x, center_y, center_z: centroids of the box.
- num_x, num_y, num_z: number of points in each direction.
- shape_list: reference to shape library. Each generated particle would have a shape that is randomly selected from the shape_list. Outputs: list of generated particles.

7.68.2 Member Function Documentation

7.68.2.1 GetGridPack() [1/4]

```
static VecXT< BondedSpheres > netdem::PackGenerator::GetGridPack (
            double len_x,
            double len_y,
            double len_z,
            double center_x,
            double center_y,
            double center_z,
            int num_x,
            int num_y,
            int num_z,
            const BondedSpheres & bonded_spheres_template )  [inline], [static]
```

7.68.2.2 GetGridPack() [2/4]

```
static VecXT< BondedVoronois > netdem::PackGenerator::GetGridPack (
            double len_x,
            double len_y,
            double len_z,
            double center_x,
            double center_y,
            double center_z,
            int num_x,
            int num_y,
            int num_z,
            const BondedVoronois & bonded_voronois_template )  [inline], [static]
```

7.68.2.3 GetGridPack() [3/4]

```
static VecXT< Particle > netdem::PackGenerator::GetGridPack (
            double len_x,
            double len_y,
            double len_z,
            double center_x,
            double center_y,
            double center_z,
            int num_x,
            int num_y,
            int num_z,
            const VecXT< Shape * > & shape_list )  [inline], [static]
```

7.68.2.4 GetGridPack() [4/4]

```
static VecXT< Particle > netdem::PackGenerator::GetGridPack (
            double len_x,
            double len_y,
            double len_z,
            double center_x,
            double center_y,
            double center_z,
            int num_x,
            int num_y,
            int num_z,
            Shape * shape )  [inline], [static]
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/gen_pack.hpp

7.69 netdem::pair_hash Struct Reference

```
#include <utils_macros.hpp>
```

Public Member Functions

- template<class T1 , class T2 >
  int operator() (const std::pair< T1, T2 > &p) const

7.69.1 Member Function Documentation

7.69.1.1 operator()()

```
template<class T1 , class T2 >
int netdem::pair_hash::operator() (
            const std::pair< T1, T2 > & p ) const  [inline]
```

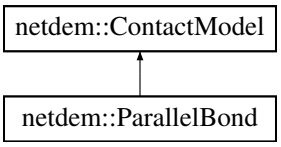
The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/utils_macros.hpp

7.70 netdem::ParallelBond Class Reference

```
#include <model_parallel_bond.hpp>
```

Inheritance diagram for netdem::ParallelBond:



Public Member Functions

- ParallelBond ()
- ParallelBond (double kn, double kt, double sig_n, double sig_t)
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void SetProperty (nlohmann::json const &js) override
- ContactModel * Clone () const override
- void SetRadius (double r)
- void EvaluateForceMoment (ContactForces *const cnt_forces, BondGeometries &cnt_geoms, ContactPP *const cnt, double dt) const override
- void EvaluateForceMoment (ContactForces *const cnt_forces, BondGeometries &cnt_geoms, ContactPW *const cnt, double dt) const override
- void Print () const override

Public Attributes

- double kn {2e6}
- double kt {1e6}
- double max_sig_n {1.0e6}
- double max_sig_t {1.0e6}

Additional Inherited Members

7.70.1 Constructor & Destructor Documentation

7.70.1.1 ParallelBond() [1/2]

```
netdem::ParallelBond::ParallelBond ( )
```

7.70.1.2 ParallelBond() [2/2]

```
netdem::ParallelBond::ParallelBond (
            double kn,
            double kt,
            double sig_n,
            double sig_t )
```

7.70.2 Member Function Documentation

7.70.2.1 Clone()

```
ContactModel * netdem::ParallelBond::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.70.2.2 EvaluateForceMoment() [1/2]

```
void netdem::ParallelBond::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            BondGeometries & cnt_geoms,
            ContactPP *const cnt,
            double dt ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.70.2.3 EvaluateForceMoment() [2/2]

```
void netdem::ParallelBond::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            BondGeometries & cnt_geoms,
            ContactPW *const cnt,
            double dt ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.70.2.4 InitFromJson()

```
void netdem::ParallelBond::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.70.2.5 PackJson()

```
nlohmann::json netdem::ParallelBond::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.70.2.6 Print()

```
void netdem::ParallelBond::Print ( ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.70.2.7 SetProperty()

```
void netdem::ParallelBond::SetProperty (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.70.2.8 SetRadius()

```
void netdem::ParallelBond::SetRadius (
            double r )
```

7.70.3 Member Data Documentation

7.70.3.1 kn

```
double netdem::ParallelBond::kn {2e6}
```

7.70.3.2 kt

```
double netdem::ParallelBond::kt {1e6}
```

7.70.3.3 max_sig_n

```
double netdem::ParallelBond::max_sig_n {1.0e6}
```

7.70.3.4 max_sig_t

```
double netdem::ParallelBond::max_sig_t {1.0e6}
```

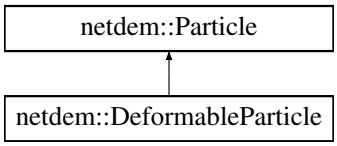
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_parallel_bond.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_parallel_bond.cpp

7.71 netdem::Particle Class Reference

```
#include <particle.hpp>
```

Inheritance diagram for netdem::Particle:



Public Member Functions

- Particle ()
- Particle (Shape *const shape)
- virtual Particle * Clone () const
- virtual void Init ()
- virtual void SetShape (Shape *const shape)
- virtual void SetDensity (double dens)
- virtual void SetForce (double fx, double fy, double fz)
- virtual void SetMoment (double mx, double my, double mz)
- virtual void SetPosition (double pos_x, double pos_y, double pos_z)
- virtual void SetRodrigues (double angle, double axis_x, double axis_y, double axis_z)
- virtual void SetQuaternion (double q_0, double q_1, double q_2, double q_3)
- virtual void SetVelocity (double v_x, double v_y, double v_z)
- virtual void SetSpin (double spin_x, double spin_y, double spin_z)
- virtual Vec3d GetVelocity (Vec3d const &cnt_pos)
- virtual void AddForce (const Vec3d &force)
- virtual void AddMoment (const Vec3d &moment)
- virtual void AddForceAtomic (const Vec3d &f)
- virtual void AddMomentAtomic (const Vec3d &m)
- virtual void ClearForce ()
- virtual void ClearMoment ()
- virtual void ApplyContactForce (ContactPP const *cnt)
- virtual void ApplyContactForce (ContactPW const *cnt)
- virtual void UpdateContactForce ()
- virtual void UpdateMotion (double timestep)
- virtual void UpdateBound ()
- void ClearLinkedCells ()
- void ClearLinkedDomains ()
- void ClearLinkedNeighs ()
- void BuildContactRef ()
- void ClearContactRef ()
- void UpdateLinkedCells (DomainManager *const dm)
- void UpdateLinkedDomains (DomainManager *const dm)
- void UpdateLinkedNeighs (DomainManager *const dm)
- VecXT< ContactPP * > GetContactPPs ()
- VecXT< ContactPW * > GetContactPWs ()
- virtual void UpdateSTLModel ()
- virtual STLModel GetSTLModel (int num_facet=400)
- virtual void SaveAsVTK (std::string const &filename)
- virtual void Print () const
- virtual ~Particle ()
- NeighPofP * MakeLinked (Particle *const q)
- NeighWofP * MakeLinked (Wall *const w)
- NeighPofP * BuildContactRef (Particle *const q, ContactPP *const cnt)
- NeighWofP * BuildContactRef (Wall *const w, ContactPW *const cnt)
- int FindLinked (Particle *const q)
- int FindLinked (Wall *const w)
- int FindContactRef (Particle *const q)
- int FindContactRef (Wall *const w)

Public Attributes

- int id {0}
- Shape * shape {nullptr}
- Vec3d bound_min {0, 0, 0}
- Vec3d bound_max {0, 0, 0}
- double margin {0}
- Vec3d bound_disp {0, 0, 0}
- int material_type {0}
- double density {2650}
- double mass {0.0}
- Vec3d moi_principal {0.0}
- double damp_global {0}
- Vec3d pos {0, 0, 0}
- Vec4d quaternion {1, 0, 0, 0}
- Vec3d vel {0, 0, 0}
- Vec3d spin {0, 0, 0}
- Vec3d vel_m0p5 {0, 0, 0}
- Vec3d spin_principal {0, 0, 0}
- Vec3d force {0, 0, 0}
- Vec3d moment {0, 0, 0}
- MiniMap< std::string, double > dynamic_properties

  customized properties
- bool enable_rotation {true}
- bool enable_bound_aabb {false}
- bool need_update_linked_list {true}
- VecXT< std::pair< Cell *, int > > linked_cell_list
- VecXT< NeighPofP > linked_particle_list
- VecXT< NeighWofP > linked_wall_list
- VecXT< NeighPofP > contact_pp_ref_table
- VecXT< NeighWofP > contact_pw_ref_table
- bool is_on_edge {false}
- bool need_send_out {false}
- VecXT< std::pair< Domain *, int > > linked_domain_list
- bool need_update_stl_model {false}
- STLModel stl_model

7.71.1 Constructor & Destructor Documentation

7.71.1.1 Particle() [1/2]

```
Particle::Particle ( )
```

7.71.1.2 Particle() [2/2]

```
Particle::Particle (
            Shape *const shape )
```

7.71.1.3 ~Particle()

```
Particle::~Particle ( )  [virtual]
```

7.71.2 Member Function Documentation

7.71.2.1 AddForce()

```
void Particle::AddForce (
            const Vec3d & force )  [virtual]
```

7.71.2.2 AddForceAtomic()

```
void Particle::AddForceAtomic (
            const Vec3d & f )  [virtual]
```

7.71.2.3 AddMoment()

```
void Particle::AddMoment (
            const Vec3d & moment )  [virtual]
```

7.71.2.4 AddMomentAtomic()

```
void Particle::AddMomentAtomic (
            const Vec3d & m )  [virtual]
```

7.71.2.5 ApplyContactForce() [1/2]

```
void Particle::ApplyContactForce (
            ContactPP const * cnt )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.6 ApplyContactForce() [2/2]

```
void Particle::ApplyContactForce (
            ContactPW const * cnt )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.7 BuildContactRef() [1/3]

```
void Particle::BuildContactRef ( )
```

7.71.2.8 BuildContactRef() [2/3]

```
NeighPofP * Particle::BuildContactRef (
            Particle *const q,
            ContactPP *const cnt )
```

7.71.2.9 BuildContactRef() [3/3]

```
NeighWofP * Particle::BuildContactRef (
            Wall *const w,
            ContactPW *const cnt )
```

7.71.2.10 ClearContactRef()

```
void Particle::ClearContactRef ( )
```

7.71.2.11 ClearForce()

```
void Particle::ClearForce ( )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.12 ClearLinkedCells()

```
void Particle::ClearLinkedCells ( )
```

7.71.2.13 ClearLinkedDomains()

```
void Particle::ClearLinkedDomains ( )
```

7.71.2.14 ClearLinkedNeighs()

```
void Particle::ClearLinkedNeighs ( )
```

7.71.2.15 ClearMoment()

```
void Particle::ClearMoment ( )  [virtual]
```

7.71.2.16 Clone()

```
Particle * Particle::Clone ( ) const  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.17 FindContactRef() [1/2]

```
int Particle::FindContactRef (
            Particle *const q )
```

7.71.2.18 FindContactRef() [2/2]

```
int Particle::FindContactRef (
            Wall *const w )
```

7.71.2.19 FindLinked() [1/2]

```
int Particle::FindLinked (
            Particle *const q )
```

7.71.2.20 FindLinked() [2/2]

```
int Particle::FindLinked (
            Wall *const w )
```

7.71.2.21 GetContactPPs()

```
VecXT< ContactPP * > Particle::GetContactPPs ( )
```

7.71.2.22 GetContactPWs()

```
VecXT< ContactPW * > Particle::GetContactPWs ( )
```

7.71.2.23 GetSTLModel()

```
STLModel Particle::GetSTLModel (
            int num_facet = 400 )  [virtual]
```

7.71.2.24 GetVelocity()

```
Vec3d Particle::GetVelocity (
            Vec3d const & cnt_pos )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.25 Init()

```
void Particle::Init ( )  [virtual]
```

7.71.2.26 MakeLinked() [1/2]

```
NeighPofP * Particle::MakeLinked (
            Particle *const q )
```

7.71.2.27 MakeLinked() [2/2]

```
NeighWofP * Particle::MakeLinked (
            Wall *const w )
```

7.71.2.28 Print()

```
void Particle::Print ( ) const  [virtual]
```

7.71.2.29 SaveAsVTK()

```
void Particle::SaveAsVTK (
            std::string const & filename )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.30 SetDensity()

```
void Particle::SetDensity (
            double dens )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.31 SetForce()

```
void Particle::SetForce (
            double fx,
            double fy,
            double fz )  [virtual]
```

7.71.2.32 SetMoment()

```
void Particle::SetMoment (
            double mx,
            double my,
            double mz )  [virtual]
```

7.71.2.33 SetPosition()

```
void Particle::SetPosition (
            double pos_x,
            double pos_y,
            double pos_z )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.34 SetQuaternion()

```
void Particle::SetQuaternion (
            double q_0,
            double q_1,
            double q_2,
            double q_3 )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.35 SetRodrigues()

```
void Particle::SetRodrigues (
            double angle,
            double axis_x,
            double axis_y,
            double axis_z )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.36 SetShape()

```
void Particle::SetShape (
            Shape *const shape )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.37 SetSpin()

```
void Particle::SetSpin (
            double spin_x,
            double spin_y,
            double spin_z )  [virtual]
```

7.71.2.38 SetVelocity()

```
void Particle::SetVelocity (
            double v_x,
            double v_y,
            double v_z )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.39 UpdateBound()

```
void Particle::UpdateBound ( )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.40 UpdateContactForce()

```
void Particle::UpdateContactForce ( )  [virtual]
```

7.71.2.41 UpdateLinkedCells()

```
void Particle::UpdateLinkedCells (
            DomainManager *const dm )
```

7.71.2.42 UpdateLinkedDomains()

```
void Particle::UpdateLinkedDomains (
            DomainManager *const dm )
```

7.71.2.43 UpdateLinkedNeighs()

```
void Particle::UpdateLinkedNeighs (
            DomainManager *const dm )
```

7.71.2.44 UpdateMotion()

```
void Particle::UpdateMotion (
            double timestep )  [virtual]
```

Reimplemented in netdem::DeformableParticle.

7.71.2.45 UpdateSTLModel()

```
void Particle::UpdateSTLModel ( )  [virtual]
```

7.71.3 Member Data Documentation

7.71.3.1 bound_disp

```
Vec3d netdem::Particle::bound_disp {0, 0, 0}
```

7.71.3.2 bound_max

```
Vec3d netdem::Particle::bound_max {0, 0, 0}
```

7.71.3.3 bound_min

```
Vec3d netdem::Particle::bound_min {0, 0, 0}
```

7.71.3.4 contact_pp_ref_table

```
VecXT<NeighPofP> netdem::Particle::contact_pp_ref_table
```

7.71.3.5 contact_pw_ref_table

```
VecXT<NeighWofP> netdem::Particle::contact_pw_ref_table
```

7.71.3.6 damp_global

```
double netdem::Particle::damp_global {0}
```

7.71.3.7 density

```
double netdem::Particle::density {2650}
```

7.71.3.8 dynamic_properties

```
MiniMap<std::string, double> netdem::Particle::dynamic_properties
```

customized properties

7.71.3.9 enable_bound_aabb

```
bool netdem::Particle::enable_bound_aabb {false}
```

7.71.3.10 enable_rotation

```
bool netdem::Particle::enable_rotation {true}
```

Sometimes aabb can be expensive, enable_bound_aabb determines whether we would like to use bounding sphere to approximate aabb for effiency.

7.71.3.11 force

```
Vec3d netdem::Particle::force {0, 0, 0}
```

7.71.3.12 id

```
int netdem::Particle::id {0}
```

7.71.3.13 is_on_edge

```
bool netdem::Particle::is_on_edge {false}
```

similar to the linked-list algorithm. linked_domain_list maintains the domains that is overlapped by the particle. It is used by the mpi_manager to transfer data for contact detection and resolution between sub-domains.

7.71.3.14 linked_cell_list

```
VecXT<std::pair<Cell *, int> > netdem::Particle::linked_cell_list
```

7.71.3.15 linked_domain_list

```
VecXT<std::pair<Domain *, int> > netdem::Particle::linked_domain_list
```

7.71.3.16 linked_particle_list

```
VecXT<NeighPofP> netdem::Particle::linked_particle_list
```

7.71.3.17 linked_wall_list

```
VecXT<NeighWofP> netdem::Particle::linked_wall_list
```

7.71.3.18 margin

```
double netdem::Particle::margin {0}
```

7.71.3.19 mass

```
double netdem::Particle::mass {0.0}
```

7.71.3.20 material_type

```
int netdem::Particle::material_type {0}
```

7.71.3.21 moi_principal

```
Vec3d netdem::Particle::moi_principal {0.0}
```

7.71.3.22 moment

```
Vec3d netdem::Particle::moment {0, 0, 0}
```

7.71.3.23 need_send_out

```
bool netdem::Particle::need_send_out {false}
```

7.71.3.24 need_update_linked_list

```
bool netdem::Particle::need_update_linked_list {true}
```

linked-list algorithm for broad-phase contact detection: pair.first represents the linked object, and pair.second represents the id of this object in its linked objects' list

7.71.3.25 need_update_stl_model

```
bool netdem::Particle::need_update_stl_model {false}
```

this is only for trimesh intersection-based contact detection and resolution, disable if not the case for efficiency

7.71.3.26 pos

```
Vec3d netdem::Particle::pos {0, 0, 0}
```

7.71.3.27 quaternion

```
Vec4d netdem::Particle::quaternion {1, 0, 0, 0}
```

7.71.3.28 shape

```
Shape* netdem::Particle::shape {nullptr}
```

7.71.3.29 spin

```
Vec3d netdem::Particle::spin {0, 0, 0}
```

7.71.3.30 spin_principal

```
Vec3d netdem::Particle::spin_principal {0, 0, 0}
```

7.71.3.31 stl_model

```
STLModel netdem::Particle::stl_model
```

7.71.3.32 vel

```
Vec3d netdem::Particle::vel {0, 0, 0}
```

7.71.3.33 vel_m0p5

```
Vec3d netdem::Particle::vel_m0p5 {0, 0, 0}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/particle.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/particle.cpp

7.72 netdem::ParticleData Struct Reference

```
#include <particle_data.hpp>
```

Public Attributes

- int id {0}
- int shape_id {0}
- double bound_min [3] {0, 0, 0}
- double bound_max [3] {0, 0, 0}
- double margin {0}
- double bound_disp [3] {0, 0, 0}
- int material_type {0}
- double density {2650}
- double damp_global {0}
- double pos [3] {0, 0, 0}
- double quaternion [4] {1, 0, 0, 0}
- double vel [3] {0, 0, 0}
- double spin [3] {0, 0, 0}
- double vel_m0p5 [3] {0, 0, 0}
- double spin_principal [3] {0, 0, 0}
- double force [3] {0, 0, 0}
- double moment [3] {0, 0, 0}
- bool enable_rotation {true}
- bool enable_bound_aabb {false}
- bool need_update_linked_list {true}

7.72.1 Detailed Description

Defines the particle data for MPI. The ParticleData struct contains all the properties of the Particle class, expect for the pointers or references. The pointers or references will be re-built based on the particle ids. Note that the particle class stores the pointers or references of a DEM object, so that it can efficiently locate the target object. Pointers or references cannot be excanged through MPI as each process has its own memery space and addressing rules.

For meanings of the properties, please refer to Particle class defined in particle.hpp.

7.72.2 Member Data Documentation

7.72.2.1 bound_disp

```
double netdem::ParticleData::bound_disp[3] {0, 0, 0}
```

7.72.2.2 bound_max

```
double netdem::ParticleData::bound_max[3] {0, 0, 0}
```

7.72.2.3 bound_min

```
double netdem::ParticleData::bound_min[3] {0, 0, 0}
```

7.72.2.4 damp_global

```
double netdem::ParticleData::damp_global {0}
```

7.72.2.5 density

```
double netdem::ParticleData::density {2650}
```

7.72.2.6 enable_bound_aabb

```
bool netdem::ParticleData::enable_bound_aabb {false}
```

7.72.2.7 enable_rotation

```
bool netdem::ParticleData::enable_rotation {true}
```

7.72.2.8 force

```
double netdem::ParticleData::force[3] {0, 0, 0}
```

7.72.2.9 id

```
int netdem::ParticleData::id {0}
```

7.72.2.10 margin

```
double netdem::ParticleData::margin {0}
```

7.72.2.11 material_type

```
int netdem::ParticleData::material_type {0}
```

7.72.2.12 moment

```
double netdem::ParticleData::moment[3] {0, 0, 0}
```

7.72.2.13 need_update_linked_list

```
bool netdem::ParticleData::need_update_linked_list {true}
```

7.72.2.14 pos

```
double netdem::ParticleData::pos[3] {0, 0, 0}
```

7.72.2.15 quaternion

```
double netdem::ParticleData::quaternion[4] {1, 0, 0, 0}
```

7.72.2.16 shape_id

```
int netdem::ParticleData::shape_id {0}
```

7.72.2.17 spin

```
double netdem::ParticleData::spin[3] {0, 0, 0}
```

7.72.2.18 spin_principal

```
double netdem::ParticleData::spin_principal[3] {0, 0, 0}
```

7.72.2.19 vel

```
double netdem::ParticleData::vel[3] {0, 0, 0}
```

7.72.2.20 vel_m0p5

```
double netdem::ParticleData::vel_m0p5[3] {0, 0, 0}
```

The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/particle_data.hpp

7.73 netdem::ParticleEnergy Struct Reference

```
#include <particle_energy_cal.hpp>
```

Public Attributes

- double total {0}
- double kinetic {0}
- double gravitational {0}
- double translational {0}
- double rotational {0}

7.73.1 Member Data Documentation

7.73.1.1 gravitational

```
double netdem::ParticleEnergy::gravitational {0}
```

7.73.1.2 kinetic

```
double netdem::ParticleEnergy::kinetic {0}
```

7.73.1.3 rotational

```
double netdem::ParticleEnergy::rotational {0}
```

7.73.1.4 total

```
double netdem::ParticleEnergy::total {0}
```

7.73.1.5 translational

```
double netdem::ParticleEnergy::translational {0}
```

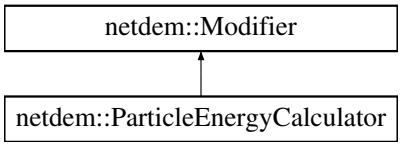
The documentation for this struct was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/particle_energy_cal.hpp

7.74 netdem::ParticleEnergyCalculator Class Reference

```
#include <particle_energy_cal.hpp>
```

Inheritance diagram for netdem::ParticleEnergyCalculator:



Public Member Functions

- ParticleEnergyCalculator ()
- void SetParticlesFromScene ()
- void SetParticles (const VecXT< int > &id_list)
- void SetParticles (int num_ids,...)
- ParticleEnergy GetEnergy ()
- ParticleEnergy GetEnergy (Particle *const p)
- Modifier * Clone () const override
- void Execute () override
- void Execute (const VecXT< Particle * > &p_list)
- void Update () override

Public Attributes

- VecXT< int > particle_id_list
- VecXT< Particle * > particle_list
- VecXT< ParticleEnergy > particle_energy_list
- bool use_particles_in_scene {false}

7.74.1 Detailed Description

To calculate the energy of a particle.

7.74.2 Constructor & Destructor Documentation

7.74.2.1 ParticleEnergyCalculator()

```
netdem::ParticleEnergyCalculator::ParticleEnergyCalculator ( )
```

7.74.3 Member Function Documentation

7.74.3.1 Clone()

```
Modifier * netdem::ParticleEnergyCalculator::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.74.3.2 Execute() [1/2]

```
void netdem::ParticleEnergyCalculator::Execute ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.74.3.3 Execute() [2/2]

```
void netdem::ParticleEnergyCalculator::Execute (
            const VecXT< Particle * > & p_list )
```

7.74.3.4 GetEnergy() [1/2]

```
ParticleEnergy netdem::ParticleEnergyCalculator::GetEnergy ( )
```

7.74.3.5 GetEnergy() [2/2]

```
ParticleEnergy netdem::ParticleEnergyCalculator::GetEnergy (
            Particle *const p )
```

7.74.3.6 SetParticles() [1/2]

```
void netdem::ParticleEnergyCalculator::SetParticles (
            const VecXT< int > & id_list )
```

7.74.3.7 SetParticles() [2/2]

```
void netdem::ParticleEnergyCalculator::SetParticles (
            int num_ids,
             ...  )
```

7.74.3.8 SetParticlesFromScene()

```
void netdem::ParticleEnergyCalculator::SetParticlesFromScene ( )
```

7.74.3.9 Update()

```
void netdem::ParticleEnergyCalculator::Update ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.74.4 Member Data Documentation

7.74.4.1 particle_energy_list

```
VecXT<ParticleEnergy> netdem::ParticleEnergyCalculator::particle_energy_list
```

7.74.4.2 particle_id_list

```
VecXT<int> netdem::ParticleEnergyCalculator::particle_id_list
```

7.74.4.3 particle_list

```
VecXT<Particle *> netdem::ParticleEnergyCalculator::particle_list
```

7.74.4.4 use_particles_in_scene

```
bool netdem::ParticleEnergyCalculator::use_particles_in_scene {false}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/particle_energy_cal.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/particle_energy_cal.cpp

7.75 netdem::ParticleParser Class Reference

```
#include <particle_parser.hpp>
```

Static Public Member Functions

- static void ClassToStruct (const Particle *const p_class, ParticleData *const p_struct)
- static void StructToClass (Particle *const p_class, const ParticleData *const p_struct, const std::unordered_map< int, Shape * > &shape_map)
- static void DefineMPIDataType (MPI_Datatype *const datatype)

7.75.1 Detailed Description

convert particle class from/to particle data struct

7.75.2 Member Function Documentation

7.75.2.1 ClassToStruct()

```
void ParticleParser::ClassToStruct (
            const Particle *const p_class,
            ParticleData *const p_struct )  [static]
```

7.75.2.2 DefineMPIDataType()

```
void ParticleParser::DefineMPIDataType (
            MPI_Datatype *const datatype )  [static]
```

7.75.2.3 StructToClass()

```
void ParticleParser::StructToClass (
            Particle *const p_class,
            const ParticleData *const p_struct,
            const std::unordered_map< int, Shape * > & shape_map )  [static]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/particle_parser.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/particle_parser.cpp

7.76 netdem::ParticleStrengthParameters Class Reference

```
#include <particle_strength_parameters.hpp>
```

Public Member Functions

- double GetEnergyReleaseRate (double size)
- double GetEnergyReleaseRate (double size, double percentile)

Public Attributes

- double ref_size = 1.5e-3
- double ref_energy_release_rate = 60.0
- double weibull_modulus = 3.1
- double weibull_coef_a = -0.76
- double weibull_coef_b = 1.13
- double min_breakable_size = 0.02

7.76.1 Member Function Documentation

7.76.1.1 GetEnergyReleaseRate() [1/2]

```
double netdem::ParticleStrengthParameters::GetEnergyReleaseRate (
            double size )  [inline]
```

7.76.1.2 GetEnergyReleaseRate() [2/2]

```
double netdem::ParticleStrengthParameters::GetEnergyReleaseRate (
            double size,
            double percentile )  [inline]
```

7.76.2 Member Data Documentation

7.76.2.1 min_breakable_size

```
double netdem::ParticleStrengthParameters::min_breakable_size = 0.02
```

7.76.2.2 ref_energy_release_rate

```
double netdem::ParticleStrengthParameters::ref_energy_release_rate = 60.0
```

7.76.2.3 ref_size

```
double netdem::ParticleStrengthParameters::ref_size = 1.5e-3
```

7.76.2.4 weibull_coef_a

```
double netdem::ParticleStrengthParameters::weibull_coef_a = -0.76
```

7.76.2.5 weibull_coef_b

```
double netdem::ParticleStrengthParameters::weibull_coef_b = 1.13
```

7.76.2.6 weibull_modulus

```
double netdem::ParticleStrengthParameters::weibull_modulus = 3.1
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/particle_strength_parame

7.77 netdem::PeriDigmBlock Class Reference

```
#include <peridigm_block.hpp>
```

Public Member Functions

- void WriteInputFile (std::ostream &os, int block_id)

Public Attributes

- VecXT< int > node_indices
- int material_id
- int damage_model_id
- double horizon

7.77.1 Member Function Documentation

7.77.1.1 WriteInputFile()

```
void netdem::PeriDigmBlock::WriteInputFile (
            std::ostream & os,
            int block_id )  [inline]
```

7.77.2 Member Data Documentation

7.77.2.1 damage_model_id

```
int netdem::PeriDigmBlock::damage_model_id
```

7.77.2.2 horizon

```
double netdem::PeriDigmBlock::horizon
```

7.77.2.3 material_id

```
int netdem::PeriDigmBlock::material_id
```

7.77.2.4 node_indices

```
VecXT<int> netdem::PeriDigmBlock::node_indices
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_block.hpp

7.78 netdem::PeriDigmBoundaryCondition Class Reference

```
#include <peridigm_boundary_condition.hpp>
```

Public Types

- enum Type { Prescribed_Displacement , Body_Force }

Public Member Functions

- void InsertNode (int node_set_id)
- void SetActivatedDimensions (bool x, bool y, bool z)
- void SetByDisplacementRate (double x, double y, double z)
- void SetByUltimateDisplacement (double x, double y, double z, double t)
- void SetByLoadingRate (double x, double y, double z)
- void SetByUltimateLoading (double x, double y, double z, double t)
- void WriteInputFile (std::ostream &os, int node_set_id)
- void WriteNodeSetFile (std::string const &result_dir, int node_set_id)
- std::string GetNodeSetFileName (int node_set_id)
- std::string GetDisplacementString (int dim)
- std::string GetLoadingString (int dim)

Public Attributes

- Type type {Type::Prescribed_Displacement}
- VecXT< int > node_indices
- VecNT< bool, 3 > dim_activated {true, true, true}
- bool time_depedent {true}
- Vec3d disp_rate {0, 0, 0}
- Vec3d loading_rate {0, 0, 0}
- Vec3d disp {0, 0, 0}
- Vec3d loading {0, 0, 0}
- double mech_time {0}

7.78.1 Member Enumeration Documentation

7.78.1.1 Type

```
enum netdem::PeriDigmBoundaryCondition::Type
```

Enumerator

Prescribed_Displacement	
Body_Force	

7.78.2 Member Function Documentation

7.78.2.1 GetDisplacementString()

```
std::string netdem::PeriDigmBoundaryCondition::GetDisplacementString (
            int dim )  [inline]
```

7.78.2.2 GetLoadingString()

```
std::string netdem::PeriDigmBoundaryCondition::GetLoadingString (
            int dim )  [inline]
```

7.78.2.3 GetNodeSetFileName()

```
std::string netdem::PeriDigmBoundaryCondition::GetNodeSetFileName (
            int node_set_id )  [inline]
```

7.78.2.4 InsertNode()

```
void netdem::PeriDigmBoundaryCondition::InsertNode (
            int node_set_id )  [inline]
```

7.78.2.5 SetActivatedDimensions()

```
void netdem::PeriDigmBoundaryCondition::SetActivatedDimensions (
            bool x,
            bool y,
            bool z )  [inline]
```

7.78.2.6 SetByDisplacementRate()

```
void netdem::PeriDigmBoundaryCondition::SetByDisplacementRate (
            double x,
            double y,
            double z )  [inline]
```

7.78.2.7 SetByLoadingRate()

```
void netdem::PeriDigmBoundaryCondition::SetByLoadingRate (
            double x,
            double y,
            double z )  [inline]
```

7.78.2.8 SetByUltimateDisplacement()

```
void netdem::PeriDigmBoundaryCondition::SetByUltimateDisplacement (
            double x,
            double y,
            double z,
            double t )  [inline]
```

7.78.2.9 SetByUltimateLoading()

```
void netdem::PeriDigmBoundaryCondition::SetByUltimateLoading (
            double x,
            double y,
            double z,
            double t )  [inline]
```

7.78.2.10 WriteInputFile()

```
void netdem::PeriDigmBoundaryCondition::WriteInputFile (
            std::ostream & os,
            int node_set_id )  [inline]
```

7.78.2.11 WriteNodeSetFile()

```
void netdem::PeriDigmBoundaryCondition::WriteNodeSetFile (
            std::string const & result_dir,
            int node_set_id )  [inline]
```

7.78.3 Member Data Documentation

7.78.3.1 dim_activated

```
VecNT<bool, 3> netdem::PeriDigmBoundaryCondition::dim_activated {true, true, true}
```

7.78.3.2 disp

```
Vec3d netdem::PeriDigmBoundaryCondition::disp {0, 0, 0}
```

7.78.3.3 disp_rate

```
Vec3d netdem::PeriDigmBoundaryCondition::disp_rate {0, 0, 0}
```

7.78.3.4 loading

```
Vec3d netdem::PeriDigmBoundaryCondition::loading {0, 0, 0}
```

7.78.3.5 loading_rate

```
Vec3d netdem::PeriDigmBoundaryCondition::loading_rate {0, 0, 0}
```

7.78.3.6 mech_time

```
double netdem::PeriDigmBoundaryCondition::mech_time {0}
```

7.78.3.7 node_indices

```
VecXT<int> netdem::PeriDigmBoundaryCondition::node_indices
```

7.78.3.8 time_depedent

```
bool netdem::PeriDigmBoundaryCondition::time_depedent {true}
```

7.78.3.9 type

```
Type netdem::PeriDigmBoundaryCondition::type {Type::Prescribed_Displacement}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_boundary_cond

7.79 netdem::PeriDigmDamageModel Class Reference

`#include <peridigm_damage_model.hpp>`

Public Types

- enum Type { Critical_Stretch }

Public Member Functions

- void InitFromEnergyReleaseRate (double youngs_modulus, double poissons_ratio, double horizon, double energy_release_rate)
- void WriteInputFile (std::ostream &os, int damage_model_id)

Static Public Member Functions

- static double GetStretchFromEnergyReleaseRate (double youngs_modulus, double poissons_ratio, double horizon, double energy_release_rate)

Public Attributes

- Type type {Type::Critical_Stretch}
- double critical_stretch {1.0e-2}

7.79.1 Member Enumeration Documentation

7.79.1.1 Type

`enum netdem::PeriDigmDamageModel::Type`

Enumerator

Critical_Stretch

7.79.2 Member Function Documentation

7.79.2.1 GetStretchFromEnergyReleaseRate()

```
static double netdem::PeriDigmDamageModel::GetStretchFromEnergyReleaseRate (
            double youngs_modulus,
```

```
            double poissons_ratio,
            double horizon,
            double energy_release_rate )  [inline], [static]
```

7.79.2.2 InitFromEnergyReleaseRate()

```
void netdem::PeriDigmDamageModel::InitFromEnergyReleaseRate (
            double youngs_modulus,
            double poissons_ratio,
            double horizon,
            double energy_release_rate )  [inline]
```

7.79.2.3 WriteInputFile()

```
void netdem::PeriDigmDamageModel::WriteInputFile (
            std::ostream & os,
            int damage_model_id )  [inline]
```

7.79.3 Member Data Documentation

7.79.3.1 critical_stretch

```
double netdem::PeriDigmDamageModel::critical_stretch {1.0e-2}
```

7.79.3.2 type

```
Type netdem::PeriDigmDamageModel::type {Type::Critical_Stretch}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_damage_model

7.80 netdem::PeriDigmDEMCoupler Class Reference

```
#include <peridigm_dem_coupler.hpp>
```

Public Member Functions

- PeriDigmDEMCoupler ()
- void Init (Particle *p)
- void Solve ()
- void ApplyBoundaryForce (Vec3d const &pos, Vec3d const &force)
- bool CheckBreakage ()
- VecXT< DEMFragment > GetFragments ()

Public Attributes

- std::string base_dir {"tmp/out/"}

  working directory
- int sub_dir_index {0}
- Particle * particle {nullptr}
- PeriDigmSimulator pd_sim
- STLModel surface_stl
- int mesh_res {20}
- double node_size_ave {0.0}
- VecXT< int > fixed_nodes
- VecXT< int > boundary_force_nodes
- VecXT< double > boundary_force_node_vols
- VecXT< Vec3d > boundary_force_values
- VecXT< int > unbalanced_force_nodes
- Vec3d unbalanced_force_values {0, 0, 0}
- double contact_force_max {0.0}
- VecXT< double > contact_force_list
- bool use_customized_loading_rate {false}
- double loading_rate {1.0e5}
- int loading_steps {1000}
- double mech_time {0.0}
- bool is_broken {false}

  fragment reconstruction settings
- double damage_fraction_limit {0.05}
- double fragment_vol_limit {0.001}
- bool ignore_fines {true}
- bool use_alpha_shape {true}
- double fragment_alpha {0.0}
- ParticleStrengthParameters strength_params
- PeriDigmMaterial material_params

Private Member Functions

- void UpdateMaterials ()
- void UpdateMechTime ()
- void UpdateCriticalStretch ()
- std::string GetResultDirectory ()
- VecXT< int > SeperateFragments (const VecXT< Vec2i > &bond_list)
- VecXT< int > GetFragmentNodeIndices (const VecXT< Vec2i > &bond_list, const VecXT< int > &frag_↩ id_list, int frag_id)
- VecXT< Vec3d > GetFragmentNodes (const VecXT< Vec3d > &node_list, const VecXT< int > &node_ids)
- VecXT< Vec3d > GetFragmentNodeVelocities (const VecXT< Vec3d > &velocity_list, const VecXT< int > &node_ids)

- VecXT< double > GetFragmentNodevolumes (const VecXT< double > &node_vol_list, const VecXT< int > &node_ids)
- STLModel GetAlphaShape (const VecXT< Vec3d > &point_list, double alpha)
- void ResolveFragmentOverlap (VecXT< DEMFragment > *const frag_list)
- DEMFragment GetFragmentCombined (const VecXT< DEMFragment > &frag_list)
- void WriteLogFileDEM ()

Private Attributes

- double damage_limit {0.5}
- VecXT< VecXT< double > > damage_data

7.80.1 Constructor & Destructor Documentation

7.80.1.1 PeriDigmDEMCoupler()

```
netdem::PeriDigmDEMCoupler::PeriDigmDEMCoupler ( )
```

7.80.2 Member Function Documentation

7.80.2.1 ApplyBoundaryForce()

```
void netdem::PeriDigmDEMCoupler::ApplyBoundaryForce (
            Vec3d const & pos,
            Vec3d const & force )
```

7.80.2.2 CheckBreakage()

```
bool netdem::PeriDigmDEMCoupler::CheckBreakage ( )
```

7.80.2.3 GetAlphaShape()

```
STLModel netdem::PeriDigmDEMCoupler::GetAlphaShape (
            const VecXT< Vec3d > & point_list,
            double alpha )  [private]
```

7.80.2.4 GetFragmentCombined()

```
DEMFragment netdem::PeriDigmDEMCoupler::GetFragmentCombined (
            const VecXT< DEMFragment > & frag_list )  [private]
```

7.80.2.5 GetFragmentNodeIndices()

```
VecXT< int > netdem::PeriDigmDEMCoupler::GetFragmentNodeIndices (
            const VecXT< Vec2i > & bond_list,
            const VecXT< int > & frag_id_list,
            int frag_id )  [private]
```

7.80.2.6 GetFragmentNodes()

```
VecXT< Vec3d > netdem::PeriDigmDEMCoupler::GetFragmentNodes (
            const VecXT< Vec3d > & node_list,
            const VecXT< int > & node_ids )  [private]
```

7.80.2.7 GetFragmentNodeVelocities()

```
VecXT< Vec3d > netdem::PeriDigmDEMCoupler::GetFragmentNodeVelocities (
            const VecXT< Vec3d > & velocity_list,
            const VecXT< int > & node_ids )  [private]
```

7.80.2.8 GetFragmentNodevolumes()

```
VecXT< double > netdem::PeriDigmDEMCoupler::GetFragmentNodevolumes (
            const VecXT< double > & node_vol_list,
            const VecXT< int > & node_ids )  [private]
```

7.80.2.9 GetFragments()

```
VecXT< DEMFragment > netdem::PeriDigmDEMCoupler::GetFragments ( )
```

7.80.2.10 GetResultDirectory()

```
string netdem::PeriDigmDEMCoupler::GetResultDirectory ( )  [private]
```

7.80.2.11 Init()

```
void netdem::PeriDigmDEMCoupler::Init (
            Particle * p )
```

7.80.2.12 ResolveFragmentOverlap()

```
void netdem::PeriDigmDEMCoupler::ResolveFragmentOverlap (
            VecXT< DEMFragment > *const frag_list )  [private]
```

7.80.2.13 SeperateFragments()

```
VecXT< int > netdem::PeriDigmDEMCoupler::SeperateFragments (
            const VecXT< Vec2i > & bond_list )  [private]
```

7.80.2.14 Solve()

```
void netdem::PeriDigmDEMCoupler::Solve ( )
```

7.80.2.15 UpdateCriticalStretch()

```
void netdem::PeriDigmDEMCoupler::UpdateCriticalStretch ( )  [private]
```

7.80.2.16 UpdateMaterials()

```
void netdem::PeriDigmDEMCoupler::UpdateMaterials ( )  [private]
```

7.80.2.17 UpdateMechTime()

```
void netdem::PeriDigmDEMCoupler::UpdateMechTime ( )  [private]
```

7.80.2.18 WriteLogFileDEM()

```
void netdem::PeriDigmDEMCoupler::WriteLogFileDEM ( )  [private]
```

7.80.3 Member Data Documentation

7.80.3.1 base_dir

```
std::string netdem::PeriDigmDEMCoupler::base_dir {"tmp/out/"}
```

working directory

7.80.3.2 boundary_force_node_vols

```
VecXT<double> netdem::PeriDigmDEMCoupler::boundary_force_node_vols
```

7.80.3.3 boundary_force_nodes

```
VecXT<int> netdem::PeriDigmDEMCoupler::boundary_force_nodes
```

7.80.3.4 boundary_force_values

```
VecXT<Vec3d> netdem::PeriDigmDEMCoupler::boundary_force_values
```

7.80.3.5 contact_force_list

```
VecXT<double> netdem::PeriDigmDEMCoupler::contact_force_list
```

7.80.3.6 contact_force_max

```
double netdem::PeriDigmDEMCoupler::contact_force_max {0.0}
```

7.80.3.7 damage_data

```
VecXT<VecXT<double> > netdem::PeriDigmDEMCoupler::damage_data  [private]
```

7.80.3.8 damage_fraction_limit

```
double netdem::PeriDigmDEMCoupler::damage_fraction_limit {0.05}
```

7.80.3.9 damage_limit

```
double netdem::PeriDigmDEMCoupler::damage_limit {0.5}  [private]
```

7.80.3.10 fixed_nodes

```
VecXT<int> netdem::PeriDigmDEMCoupler::fixed_nodes
```

7.80.3.11 fragment_alpha

```
double netdem::PeriDigmDEMCoupler::fragment_alpha {0.0}
```

7.80.3.12 fragment_vol_limit

```
double netdem::PeriDigmDEMCoupler::fragment_vol_limit {0.001}
```

7.80.3.13 ignore_fines

```
bool netdem::PeriDigmDEMCoupler::ignore_fines {true}
```

7.80.3.14 is_broken

```
bool netdem::PeriDigmDEMCoupler::is_broken {false}
```

fragment reconstruction settings

7.80.3.15 loading_rate

```
double netdem::PeriDigmDEMCoupler::loading_rate {1.0e5}
```

7.80.3.16 loading_steps

```
int netdem::PeriDigmDEMCoupler::loading_steps {1000}
```

7.80.3.17 material_params

```
PeriDigmMaterial netdem::PeriDigmDEMCoupler::material_params
```

7.80.3.18 mech_time

```
double netdem::PeriDigmDEMCoupler::mech_time {0.0}
```

7.80.3.19 mesh_res

```
int netdem::PeriDigmDEMCoupler::mesh_res {20}
```

7.80.3.20 node_size_ave

```
double netdem::PeriDigmDEMCoupler::node_size_ave {0.0}
```

7.80.3.21 particle

```
Particle* netdem::PeriDigmDEMCoupler::particle {nullptr}
```

7.80.3.22 pd_sim

```
PeriDigmSimulator netdem::PeriDigmDEMCoupler::pd_sim
```

7.80.3.23 strength_params

```
ParticleStrengthParameters netdem::PeriDigmDEMCoupler::strength_params
```

7.80.3.24 sub_dir_index

```
int netdem::PeriDigmDEMCoupler::sub_dir_index {0}
```

7.80.3.25 surface_stl

```
STLModel netdem::PeriDigmDEMCoupler::surface_stl
```

7.80.3.26 unbalanced_force_nodes

```
VecXT<int> netdem::PeriDigmDEMCoupler::unbalanced_force_nodes
```

7.80.3.27 unbalanced_force_values

```
Vec3d netdem::PeriDigmDEMCoupler::unbalanced_force_values {0, 0, 0}
```

7.80.3.28 use_alpha_shape

```
bool netdem::PeriDigmDEMCoupler::use_alpha_shape {true}
```

7.80.3.29 use_customized_loading_rate

```
bool netdem::PeriDigmDEMCoupler::use_customized_loading_rate {false}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_dem_coupler.hp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_dem_coupler.cp

7.81 netdem::PeriDigmDiscretization Class Reference

```
#include <peridigm_discretization.hpp>
```

Public Types

- enum Type { level_set , tetmesh }

Public Member Functions

- PeriDigmDiscretization ()
- void InitFromSTL (std::string const &stl_file, int res)
- void InitFromSTL (STLModel const &stl_model, int res)
- void InitFromDistanceMap (std::string const &txt_file)
- void InitFromGrid (double corner_x, double corner_y, double corner_z, double len_x, double len_y, double len_z, int res)
- void MakePorosity (double porosity)
- void WriteNodeFile (std::string const &result_dir)
- double GetNodeSize ()
- ∼PeriDigmDiscretization ()

Public Attributes

- Type type {Type::level_set}
- DomainSplittor ∗ domain_splittor {nullptr}
- VecXT< Vec3d > nodes
- VecXT< int > node_block_indices
- VecXT< double > node_vols

Private Member Functions

- void InitDefaultBlockIndices ()

7.81.1 Member Enumeration Documentation

7.81.1.1 Type

```
enum netdem::PeriDigmDiscretization::Type
```

Enumerator

level_set	
tetmesh	

7.81.2 Constructor & Destructor Documentation

7.81.2.1 PeriDigmDiscretization()

```
PeriDigmDiscretization::PeriDigmDiscretization ( )
```

7.81.2.2 ∼PeriDigmDiscretization()

```
PeriDigmDiscretization::∼PeriDigmDiscretization ( )
```

7.81.3 Member Function Documentation

7.81.3.1 GetNodeSize()

```
double PeriDigmDiscretization::GetNodeSize ( )
```

7.81.3.2 InitDefaultBlockIndices()

```
void PeriDigmDiscretization::InitDefaultBlockIndices ( )  [private]
```

7.81.3.3 InitFromDistanceMap()

```
void PeriDigmDiscretization::InitFromDistanceMap (
            std::string const & txt_file )
```

7.81.3.4 InitFromGrid()

```
void PeriDigmDiscretization::InitFromGrid (
            double corner_x,
            double corner_y,
            double corner_z,
            double len_x,
            double len_y,
            double len_z,
            int res )
```

7.81.3.5 InitFromSTL() [1/2]

```
void netdem::PeriDigmDiscretization::InitFromSTL (
            std::string const & stl_file,
            int res )
```

7.81.3.6 InitFromSTL() [2/2]

```
void PeriDigmDiscretization::InitFromSTL (
            STLModel const & stl_model,
            int res )
```

7.81.3.7 MakePorosity()

```
void PeriDigmDiscretization::MakePorosity (
            double porosity )
```

7.81.3.8 WriteNodeFile()

```
void PeriDigmDiscretization::WriteNodeFile (
            std::string const & result_dir )
```

7.81.4 Member Data Documentation

7.81.4.1 domain_splittor

```
DomainSplittor* netdem::PeriDigmDiscretization::domain_splittor {nullptr}
```

7.81.4.2 node_block_indices

```
VecXT<int> netdem::PeriDigmDiscretization::node_block_indices
```

7.81.4.3 node_vols

```
VecXT<double> netdem::PeriDigmDiscretization::node_vols
```

7.81.4.4 nodes

```
VecXT<Vec3d> netdem::PeriDigmDiscretization::nodes
```

7.81.4.5 type

```
Type netdem::PeriDigmDiscretization::type {Type::level_set}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_discretization.hp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_discretization.cp

7.82 netdem::PeriDigmMaterial Class Reference

```
#include <peridigm_material.hpp>
```

Public Types

- enum Type { Elastic }

Public Member Functions

- void WriteInputFile (std::ostream &os, int material_id)

Public Attributes

- Type type {Type::Elastic}
- double density = 2650.0
- double youngs_modulus = 70.0e9
- double poissons_ratio = 0.15

7.82.1 Member Enumeration Documentation

7.82.1.1 Type

```
enum netdem::PeriDigmMaterial::Type
```

Enumerator

Elastic	

7.82.2 Member Function Documentation

7.82.2.1 WriteInputFile()

```
void netdem::PeriDigmMaterial::WriteInputFile (
            std::ostream & os,
            int material_id )  [inline]
```

7.82.3 Member Data Documentation

7.82.3.1 density

```
double netdem::PeriDigmMaterial::density = 2650.0
```

7.82.3.2 poissons_ratio

```
double netdem::PeriDigmMaterial::poissons_ratio = 0.15
```

7.82.3.3 type

```
Type netdem::PeriDigmMaterial::type {Type::Elastic}
```

7.82.3.4 youngs_modulus

```
double netdem::PeriDigmMaterial::youngs_modulus = 70.0e9
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_material.hpp

7.83 netdem::PeriDigmSettings Class Reference

```
#include <peridigm_settings.hpp>
```

Public Member Functions

- void WriteInputFile (std::ostream &os)

Public Attributes

- std::string result_dir {"tmp/out/peridigm/"}
- std::string peridigm_exe {"Peridigm"}
- double horizon_factor {3.01}
- bool omit_bonds_between_blocks {false}
- bool use_auto_timestep {true}
- double timestep {1.0e-6}
- double timestep_factor {0.95}
- double mech_time {0.0}
- double loading_radius_factor {1.5}
- double constrain_radius_factor {1.5}
- int output_freqency {10}

7.83.1 Member Function Documentation

7.83.1.1 WriteInputFile()

```
void netdem::PeriDigmSettings::WriteInputFile (
            std::ostream & os )  [inline]
```

7.83.2 Member Data Documentation

7.83.2.1 constrain_radius_factor

```
double netdem::PeriDigmSettings::constrain_radius_factor {1.5}
```

7.83.2.2 horizon_factor

```
double netdem::PeriDigmSettings::horizon_factor {3.01}
```

7.83.2.3 loading_radius_factor

```
double netdem::PeriDigmSettings::loading_radius_factor {1.5}
```

7.83.2.4 mech_time

```
double netdem::PeriDigmSettings::mech_time {0.0}
```

7.83.2.5 omit_bonds_between_blocks

```
bool netdem::PeriDigmSettings::omit_bonds_between_blocks {false}
```

7.83.2.6 output_freqency

```
int netdem::PeriDigmSettings::output_freqency {10}
```

7.83.2.7 peridigm_exe

```
std::string netdem::PeriDigmSettings::peridigm_exe {"Peridigm"}
```

7.83.2.8 result_dir

```
std::string netdem::PeriDigmSettings::result_dir {"tmp/out/peridigm/"}
```

7.83.2.9 timestep

```
double netdem::PeriDigmSettings::timestep {1.0e-6}
```

7.83.2.10 timestep_factor

```
double netdem::PeriDigmSettings::timestep_factor {0.95}
```

7.83.2.11 use_auto_timestep

```
bool netdem::PeriDigmSettings::use_auto_timestep {true}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_settings.hpp

7.84 netdem::PeriDigmSimulator Class Reference

```
#include <peridigm_simulator.hpp>
```

Public Member Functions

- PeriDigmSimulator ()
- PeriDigmMaterial * InsertMaterial ()
- PeriDigmDamageModel * InsertDamageModel ()
- PeriDigmBlock * InsertBlock ()
- PeriDigmBoundaryCondition * InsertBoundaryCondition ()
- void Clear ()
- void InitDefaultSetup ()
- void InitAutoTimestep ()
- void WriteNodeFile ()
- void WriteNodeSetFile ()
- void WriteInputFile ()
- void Solve (double mech_time)
- void SetUpResultDirectory ()
- void CleanUpResultDirectory ()

Public Attributes

- PeriDigmDiscretization discretization
- VecXT< PeriDigmMaterial > materials
- VecXT< PeriDigmDamageModel > damage_models
- VecXT< PeriDigmBlock > blocks
- VecXT< PeriDigmBoundaryCondition > boundary_conditions
- PeriDigmSettings settings

7.84.1 Constructor & Destructor Documentation

7.84.1.1 PeriDigmSimulator()

```
netdem::PeriDigmSimulator::PeriDigmSimulator ( )
```

7.84.2 Member Function Documentation

7.84.2.1 CleanUpResultDirectory()

```
void netdem::PeriDigmSimulator::CleanUpResultDirectory ( )
```

7.84.2.2 Clear()

```
void netdem::PeriDigmSimulator::Clear ( )
```

7.84.2.3 InitAutoTimestep()

```
void netdem::PeriDigmSimulator::InitAutoTimestep ( )
```

7.84.2.4 InitDefaultSetup()

```
void netdem::PeriDigmSimulator::InitDefaultSetup ( )
```

7.84.2.5 InsertBlock()

```
PeriDigmBlock * netdem::PeriDigmSimulator::InsertBlock ( )
```

7.84.2.6 InsertBoundaryCondition()

```
PeriDigmBoundaryCondition * netdem::PeriDigmSimulator::InsertBoundaryCondition ( )
```

7.84.2.7 InsertDamageModel()

```
PeriDigmDamageModel * netdem::PeriDigmSimulator::InsertDamageModel ( )
```

7.84.2.8 InsertMaterial()

```
PeriDigmMaterial * netdem::PeriDigmSimulator::InsertMaterial ( )
```

7.84.2.9 SetUpResultDirectory()

```
void netdem::PeriDigmSimulator::SetUpResultDirectory ( )
```

7.84.2.10 Solve()

```
void netdem::PeriDigmSimulator::Solve (
            double mech_time )
```

7.84.2.11 WriteInputFile()

```
void netdem::PeriDigmSimulator::WriteInputFile ( )
```

7.84.2.12 WriteNodeFile()

```
void netdem::PeriDigmSimulator::WriteNodeFile ( )
```

7.84.2.13 WriteNodeSetFile()

```
void netdem::PeriDigmSimulator::WriteNodeSetFile ( )
```

7.84.3 Member Data Documentation

7.84.3.1 blocks

```
VecXT<PeriDigmBlock> netdem::PeriDigmSimulator::blocks
```

7.84.3.2 boundary_conditions

```
VecXT<PeriDigmBoundaryCondition> netdem::PeriDigmSimulator::boundary_conditions
```

7.84.3.3 damage_models

```
VecXT<PeriDigmDamageModel> netdem::PeriDigmSimulator::damage_models
```

7.84.3.4 discretization

```
PeriDigmDiscretization netdem::PeriDigmSimulator::discretization
```

7.84.3.5 materials

```
VecXT<PeriDigmMaterial> netdem::PeriDigmSimulator::materials
```

7.84.3.6 settings

```
PeriDigmSettings netdem::PeriDigmSimulator::settings
```

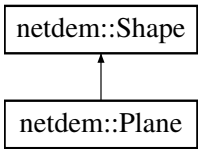
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_simulator.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_simulator.cpp

7.85 netdem::Plane Class Reference

```
#include <shape_plane.hpp>
```

Inheritance diagram for netdem::Plane:



Public Member Functions

- Plane ()
- Plane (Vec3d const &c, Vec3d const &n)
- Plane (double c_x, double c_y, double c_z, double n_x, double n_y, double n_z)
- void UpdateNodes () override
- void UpdateShapeProperties () override
- Shape * Clone () const override
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void Init ()
- void SetExtent (double e)
- void SetCenter (double c_x, double c_y, double c_z)
- void SetNormal (double n_x, double n_y, double n_z)
- std::tuple< Vec3d, Vec3d > GetBoundAABB (Vec3d const &pos, Vec4d const &quat) override
- STLModel GetSTLModel (int num_facets=400) override
- Vec3d SupportPoint (Vec3d const &dir) override
- VecXT< Vec3d > SupportPoints (Vec3d const &dir) override
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- void Print () override

Public Attributes

- Vec3d center {0, 0, 0}
- Vec3d dir_n {0, 0, 1}
- double extent {5}

Additional Inherited Members

7.85.1 Detailed Description

- center: center point on the plane
- dir_n normal of the plane

7.85.2 Constructor & Destructor Documentation

7.85.2.1 Plane() [1/3]

```
Plane::Plane ( )
```

7.85.2.2 Plane() [2/3]

```
Plane::Plane (
            Vec3d const & c,
            Vec3d const & n )
```

7.85.2.3 Plane() [3/3]

```
Plane::Plane (
            double c_x,
            double c_y,
            double c_z,
            double n_x,
            double n_y,
            double n_z )
```

7.85.3 Member Function Documentation

7.85.3.1 Clone()

```
Shape * Plane::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.2 GetBoundAABB()

```
tuple< Vec3d, Vec3d > Plane::GetBoundAABB (
            Vec3d const & pos,
            Vec4d const & quat )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.3 GetSTLModel()

```
STLModel Plane::GetSTLModel (
            int num_facets = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.4 Init()

```
void Plane::Init ( )
```

7.85.3.5 InitFromJson()

```
void Plane::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.6 PackJson()

```
nlohmann::json Plane::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.7 Print()

```
void Plane::Print ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.8 SetCenter()

```
void Plane::SetCenter (
            double c_x,
            double c_y,
            double c_z )
```

7.85.3.9 SetExtent()

```
void Plane::SetExtent (
            double e )
```

7.85.3.10 SetNormal()

```
void Plane::SetNormal (
            double n_x,
            double n_y,
            double n_z )
```

7.85.3.11 SignedDistance()

```
double Plane::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.12 SupportPoint()

```
Vec3d Plane::SupportPoint (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.13 SupportPoints()

```
VecXT< Vec3d > Plane::SupportPoints (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.14 SurfacePoint()

```
Vec3d Plane::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.85.3.15 UpdateNodes()

```
void Plane::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.3.16 UpdateShapeProperties()

```
void Plane::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.85.4 Member Data Documentation

7.85.4.1 center

```
Vec3d netdem::Plane::center {0, 0, 0}
```

7.85.4.2 dir_n

```
Vec3d netdem::Plane::dir_n {0, 0, 1}
```

7.85.4.3 extent

```
double netdem::Plane::extent {5}
```

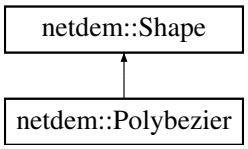
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_plane.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_plane.cpp

7.86 netdem::Polybezier Class Reference

`#include <shape_polybezier.hpp>`

Inheritance diagram for netdem::Polybezier:



Public Member Functions

- Polybezier ()
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- Shape * Clone () const override
- void InitByRandom ()
- void InitFromKernelSTL (std::string const &file)
- void InitFromKernelSTL (STLModel const &stl_model)
- void Init ()
- void UpdateShapeProperties () override
- void SetSize (double d) override
- STLModel GetSTLModel (int res=400) override
- void SaveNormalPatchesSpherical (std::string const &file)
- void SaveNormalPatchesCubic (std::string const &file)
- Vec3d SupportPoint (Vec3d const &dir) override
- VecXT< Vec3d > SupportPoints (Vec3d const &dir) override
- void Print () override

Public Attributes

- int num_patches {30}
- int order {2}
- VecXT< VecXT< Vec3d > > face_patch_knots_list
- VecXT< VecXT< Vec3d > > face_patch_normals_list
- VecXT< VecXT< Vec3d > > edge_patch_knots_list
- VecXT< Vec3i > linked_edges_list
- VecXT< VecXT< int > > linked_patches_list
- int num_cells {4}

Private Member Functions

- void AlignAxes ()
- void UpdateLinkedPatches ()
- void UpdataMatDuDv ()
- Vec3d GetEdgeKnot (Vec3d const &v0, Vec3d const &v1, Vec3d const &n01)
- void GetUniqueEdges (VecXT< Vec2i > *const edges, VecXT< Vec3i > *const linked_list, STLModel const &stl_model)
- Vec3d GetPatchNormal (Vec3d const &v0, Vec3d const &v1, Vec3d const &v2)
- void SortNormalPatchVertices (VecXT< Vec3d > *const normals)
- VecXT< Vec3d > GetCartesianProject (const VecXT< Vec3d > &normals)
- VecXT< Vec3d > GetCartesianProject (Vec3d const &v1, Vec3d const &v2)
- bool ContainCorner (Vec3d const &corner, const VecXT< Vec3d > &normals)
- STLModel GetSTLModel (const VecXT< Vec3d > &knots, int res)
- int GetSupportPatchID (Vec3d const &dir)

Private Attributes

- VecXT< VecNT< Vec3d, 3 > > mat_du_list
- VecXT< VecNT< Vec3d, 3 > > mat_dv_list

Additional Inherited Members

7.86.1 Constructor & Destructor Documentation

7.86.1.1 Polybezier()

```
Polybezier::Polybezier ( )
```

7.86.2 Member Function Documentation

7.86.2.1 AlignAxes()

```
void Polybezier::AlignAxes ( )  [private]
```

7.86.2.2 Clone()

```
Shape * Polybezier::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.86.2.3 ContainCorner()

```
bool Polybezier::ContainCorner (
            Vec3d const & corner,
            const VecXT< Vec3d > & normals )  [private]
```

7.86.2.4 GetCartesianProject() [1/2]

```
VecXT< Vec3d > Polybezier::GetCartesianProject (
            const VecXT< Vec3d > & normals )  [private]
```

7.86.2.5 GetCartesianProject() [2/2]

```
VecXT< Vec3d > Polybezier::GetCartesianProject (
            Vec3d const & v1,
            Vec3d const & v2 )  [private]
```

7.86.2.6 GetEdgeKnot()

```
Vec3d Polybezier::GetEdgeKnot (
            Vec3d const & v0,
            Vec3d const & v1,
            Vec3d const & n01 )  [private]
```

7.86.2.7 GetPatchNormal()

```
Vec3d Polybezier::GetPatchNormal (
            Vec3d const & v0,
            Vec3d const & v1,
            Vec3d const & v2 )  [private]
```

7.86.2.8 GetSTLModel() [1/2]

```
STLModel Polybezier::GetSTLModel (
            const VecXT< Vec3d > & knots,
            int res )  [private]
```

7.86.2.9 GetSTLModel() [2/2]

```
STLModel Polybezier::GetSTLModel (
            int res = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.86.2.10 GetSupportPatchID()

```
int Polybezier::GetSupportPatchID (
            Vec3d const & dir )  [private]
```

7.86.2.11 GetUniqueEdges()

```
void Polybezier::GetUniqueEdges (
            VecXT< Vec2i > *const edges,
            VecXT< Vec3i > *const linked_list,
            STLModel const & stl_model )  [private]
```

7.86.2.12 Init()

```
void Polybezier::Init ( )
```

7.86.2.13 InitByRandom()

```
void Polybezier::InitByRandom ( )
```

7.86.2.14 InitFromJson()

```
void Polybezier::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.86.2.15 InitFromKernelSTL() [1/2]

```
void netdem::Polybezier::InitFromKernelSTL (
            std::string const & file )
```

7.86.2.16 InitFromKernelSTL() [2/2]

```
void Polybezier::InitFromKernelSTL (
            STLModel const & stl_model )
```

7.86.2.17 PackJson()

```
nlohmann::json Polybezier::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.86.2.18 Print()

```
void Polybezier::Print ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.86.2.19 SaveNormalPatchesCubic()

```
void Polybezier::SaveNormalPatchesCubic (
            std::string const & file )
```

7.86.2.20 SaveNormalPatchesSpherical()

```
void Polybezier::SaveNormalPatchesSpherical (
            std::string const & file )
```

7.86.2.21 SetSize()

```
void Polybezier::SetSize (
            double d )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.86.2.22 SortNormalPatchVertices()

```
void Polybezier::SortNormalPatchVertices (
            VecXT< Vec3d > *const normals )  [private]
```

7.86.2.23 SupportPoint()

```
Vec3d Polybezier::SupportPoint (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.86.2.24 SupportPoints()

```
VecXT< Vec3d > Polybezier::SupportPoints (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.86.2.25 UpdataMatDuDv()

```
void Polybezier::UpdataMatDuDv ( )  [private]
```

7.86.2.26 UpdateLinkedPatches()

```
void Polybezier::UpdateLinkedPatches ( )  [private]
```

7.86.2.27 UpdateShapeProperties()

```
void Polybezier::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.86.3 Member Data Documentation

7.86.3.1 edge_patch_knots_list

```
VecXT<VecXT<Vec3d> > netdem::Polybezier::edge_patch_knots_list
```

7.86.3.2 face_patch_knots_list

VecXT<VecXT<Vec3d> > netdem::Polybezier::face_patch_knots_list

7.86.3.3 face_patch_normals_list

VecXT<VecXT<Vec3d> > netdem::Polybezier::face_patch_normals_list

7.86.3.4 linked_edges_list

VecXT<Vec3i> netdem::Polybezier::linked_edges_list

7.86.3.5 linked_patches_list

VecXT<VecXT<int> > netdem::Polybezier::linked_patches_list

7.86.3.6 mat_du_list

VecXT<VecNT<Vec3d, 3> > netdem::Polybezier::mat_du_list [private]

7.86.3.7 mat_dv_list

VecXT<VecNT<Vec3d, 3> > netdem::Polybezier::mat_dv_list [private]

7.86.3.8 num_cells

int netdem::Polybezier::num_cells {4}

7.86.3.9 num_patches

int netdem::Polybezier::num_patches {30}

7.86.3.10 order

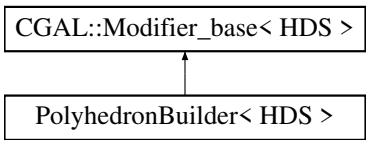
```
int netdem::Polybezier::order {2}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_polybezier.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_polybezier.cpp

7.87 PolyhedronBuilder< HDS > Class Template Reference

Inheritance diagram for PolyhedronBuilder< HDS >:



Public Member Functions

- PolyhedronBuilder (const VecXT< Vec3d > &vv, const VecXT< Vec3i > &ff)
- void operator() (HDS &hds)

Public Attributes

- VecXT< Vec3d > vertices
- VecXT< Vec3i > facets

7.87.1 Constructor & Destructor Documentation

7.87.1.1 PolyhedronBuilder()

```
template<class HDS >
PolyhedronBuilder< HDS >::PolyhedronBuilder (
            const VecXT< Vec3d > & vv,
            const VecXT< Vec3i > & ff )
```

7.87.2 Member Function Documentation

7.87.2.1 operator()()

```
template<class HDS >
void PolyhedronBuilder< HDS >::operator() (
             HDS & hds )
```

7.87.3 Member Data Documentation

7.87.3.1 facets

```
template<class HDS >
VecXT<Vec3i> PolyhedronBuilder< HDS >::facets
```

7.87.3.2 vertices

```
template<class HDS >
VecXT<Vec3d> PolyhedronBuilder< HDS >::vertices
```

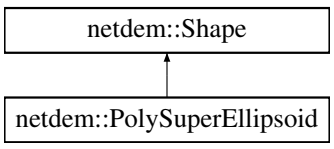
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/cgal_wrapper.cpp

7.88 netdem::PolySuperEllipsoid Class Reference

```
#include <shape_poly_super_ellipsoid.hpp>
```

Inheritance diagram for netdem::PolySuperEllipsoid:



Public Member Functions

- PolySuperEllipsoid ()
- PolySuperEllipsoid (double am, double ap, double bm, double bp, double cm, double cp, double nab, double nc)
- Shape * Clone () const override
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void Init ()
- void UpdateNodes () override
- void UpdateShapeProperties () override
- void SetSize (double d) override
- STLModel GetSTLModel (int num_facets=400) override
- Vec3d SupportPoint (Vec3d const &dir) override
- VecXT< Vec3d > SupportPoints (Vec3d const &dir) override
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- Vec3d ParametrizationPoint (Vec3d const &dir)
- void Print () override

Public Attributes

- Vec2d axis_a {0.5, 0.5}
- Vec2d axis_b {0.5, 0.5}
- Vec2d axis_c {0.5, 0.5}
- double order_ab {1.0}
- double order_c {1.0}
- Vec3d pos_ref {0, 0, 0}
- Vec4d quat_ref {1, 0, 0, 0}
- Vec4d quat_conj {1, 0, 0, 0}

Additional Inherited Members

7.88.1 Detailed Description

((x / axis_a)^(2 / order_ab) + (y / axis_b)^(2 / order_ab)) ^ (order_ab / oder_c) + (z / axis_c)^(2 / order_c) = 1 m and p indicate minus and plus to guarantee convexity, order should be in (0, 2)

7.88.2 Constructor & Destructor Documentation

7.88.2.1 PolySuperEllipsoid() [1/2]

```
PolySuperEllipsoid::PolySuperEllipsoid ( )
```

7.88.2.2 PolySuperEllipsoid() [2/2]

```
PolySuperEllipsoid::PolySuperEllipsoid (
            double am,
            double ap,
            double bm,
            double bp,
            double cm,
            double cp,
            double nab,
            double nc )
```

7.88.3 Member Function Documentation

7.88.3.1 Clone()

```
Shape * PolySuperEllipsoid::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.2 GetSTLModel()

```
STLModel PolySuperEllipsoid::GetSTLModel (
            int num_facets = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.3 Init()

```
void PolySuperEllipsoid::Init ( )
```

7.88.3.4 InitFromJson()

```
void PolySuperEllipsoid::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.5 PackJson()

```
nlohmann::json PolySuperEllipsoid::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.6 ParametrizationPoint()

```
Vec3d PolySuperEllipsoid::ParametrizationPoint (
            Vec3d const & dir )
```

7.88.3.7 Print()

```
void PolySuperEllipsoid::Print ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.8 SetSize()

```
void PolySuperEllipsoid::SetSize (
            double d )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.9 SignedDistance()

```
double PolySuperEllipsoid::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.10 SupportPoint()

```
Vec3d PolySuperEllipsoid::SupportPoint (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.11 SupportPoints()

```
VecXT< Vec3d > PolySuperEllipsoid::SupportPoints (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.12 SurfacePoint()

```
Vec3d PolySuperEllipsoid::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.88.3.13 UpdateNodes()

```
void PolySuperEllipsoid::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.3.14 UpdateShapeProperties()

```
void PolySuperEllipsoid::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.88.4 Member Data Documentation

7.88.4.1 axis_a

```
Vec2d netdem::PolySuperEllipsoid::axis_a {0.5, 0.5}
```

7.88.4.2 axis_b

```
Vec2d netdem::PolySuperEllipsoid::axis_b {0.5, 0.5}
```

7.88.4.3 axis_c

```
Vec2d netdem::PolySuperEllipsoid::axis_c {0.5, 0.5}
```

7.88.4.4 order_ab

```
double netdem::PolySuperEllipsoid::order_ab {1.0}
```

7.88.4.5 order_c

```
double netdem::PolySuperEllipsoid::order_c {1.0}
```

7.88.4.6 pos_ref

```
Vec3d netdem::PolySuperEllipsoid::pos_ref {0, 0, 0}
```

7.88.4.7 quat_conj

```
Vec4d netdem::PolySuperEllipsoid::quat_conj {1, 0, 0, 0}
```

7.88.4.8 quat_ref

```
Vec4d netdem::PolySuperEllipsoid::quat_ref {1, 0, 0, 0}
```

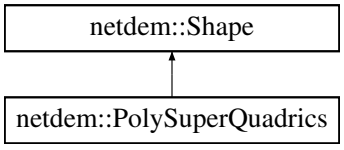
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_poly_super_ellipsoid
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_poly_super_ellipsoid

7.89 netdem::PolySuperQuadrics Class Reference

```
#include <shape_poly_super_quadrics.hpp>
```

Inheritance diagram for netdem::PolySuperQuadrics:



Public Member Functions

- PolySuperQuadrics ()
- PolySuperQuadrics (double am, double ap, double bm, double bp, double cm, double cp, double nam, double nap, double nbm, double nbp, double ncm, double ncp)
- Shape * Clone () const override
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void Init ()
- void UpdateNodes () override
- void UpdateShapeProperties () override
- void SetSize (double d) override
- STLModel GetSTLModel (int res=400) override
- Vec3d SupportPoint (Vec3d const &dir) override
- VecXT< Vec3d > SupportPoints (Vec3d const &dir) override
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- Vec3d ParametrizationPoint (Vec3d const &dir)
- void Print () override

Public Attributes

- Vec2d axis_a {0.5, 0.5}
- Vec2d axis_b {0.5, 0.5}
- Vec2d axis_c {0.5, 0.5}
- Vec2d order_a {1.0, 1.0}
- Vec2d order_b {1.0, 1.0}
- Vec2d order_c {1.0, 1.0}
- Vec3d pos_ref {0, 0, 0}
- Vec4d quat_ref {1, 0, 0, 0}
- Vec4d quat_conj {1, 0, 0, 0}

Additional Inherited Members

7.89.1 Detailed Description

(x / axis_a)^(2 / order_a) + ... = 1 m and p indicate minus and plus to guarantee convexity, order should be in (0, 2)

7.89.2 Constructor & Destructor Documentation

7.89.2.1 PolySuperQuadrics() [1/2]

```
PolySuperQuadrics::PolySuperQuadrics ( )
```

7.89.2.2 PolySuperQuadrics() [2/2]

```
PolySuperQuadrics::PolySuperQuadrics (
            double am,
            double ap,
            double bm,
            double bp,
            double cm,
            double cp,
            double nam,
            double nap,
            double nbm,
            double nbp,
            double ncm,
            double ncp )
```

7.89.3 Member Function Documentation

7.89.3.1 Clone()

```
Shape * PolySuperQuadrics::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.2 GetSTLModel()

```
STLModel PolySuperQuadrics::GetSTLModel (
            int res = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.3 Init()

```
void PolySuperQuadrics::Init ( )
```

7.89.3.4 InitFromJson()

```
void PolySuperQuadrics::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.5 PackJson()

```
nlohmann::json PolySuperQuadrics::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.6 ParametrizationPoint()

```
Vec3d PolySuperQuadrics::ParametrizationPoint (
            Vec3d const & dir )
```

7.89.3.7 Print()

```
void PolySuperQuadrics::Print ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.8 SetSize()

```
void PolySuperQuadrics::SetSize (
            double d )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.9 SignedDistance()

```
double PolySuperQuadrics::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.10 SupportPoint()

```
Vec3d PolySuperQuadrics::SupportPoint (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.11 SupportPoints()

```
VecXT< Vec3d > PolySuperQuadrics::SupportPoints (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.12 SurfacePoint()

```
Vec3d PolySuperQuadrics::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.89.3.13 UpdateNodes()

```
void PolySuperQuadrics::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.3.14 UpdateShapeProperties()

```
void PolySuperQuadrics::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.89.4 Member Data Documentation

7.89.4.1 axis_a

```
Vec2d netdem::PolySuperQuadrics::axis_a {0.5, 0.5}
```

7.89.4.2 axis_b

```
Vec2d netdem::PolySuperQuadrics::axis_b {0.5, 0.5}
```

7.89.4.3 axis_c

```
Vec2d netdem::PolySuperQuadrics::axis_c {0.5, 0.5}
```

7.89.4.4 order_a

```
Vec2d netdem::PolySuperQuadrics::order_a {1.0, 1.0}
```

7.89.4.5 order_b

```
Vec2d netdem::PolySuperQuadrics::order_b {1.0, 1.0}
```

7.89.4.6 order_c

```
Vec2d netdem::PolySuperQuadrics::order_c {1.0, 1.0}
```

7.89.4.7 pos_ref

```
Vec3d netdem::PolySuperQuadrics::pos_ref {0, 0, 0}
```

7.89.4.8 quat_conj

```
Vec4d netdem::PolySuperQuadrics::quat_conj {1, 0, 0, 0}
```

7.89.4.9 quat_ref

```
Vec4d netdem::PolySuperQuadrics::quat_ref {1, 0, 0, 0}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_poly_super_quadrics
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_poly_super_quadrics

7.90 netdem::RegressionNet Class Reference

`#include <regression_net.hpp>`

Public Member Functions

- void ResetModel ()
- void AddLayer (LayerName layer_name,...)
- void Train (const arma::mat &data_x, const arma::mat &data_y)
- arma::mat Predict (const arma::mat &data_x)
- void Load (std::string const &filename, std::string const &label)
- void Save (std::string const &filename, std::string const &label)

Public Attributes

- mlpack::ann::FFN< mlpack::ann::MeanSquaredError<>, mlpack::ann::HeInitialization > model
- double step_size {0.01}
- int batch_size {32}
- double decay_rate_moment {0.9}
- double decay_rate_norm {0.9}
- double gradient_init_param {1e-8}
- int epochs {100}
- double stop_tol {1e-8}

7.90.1 Member Function Documentation

7.90.1.1 AddLayer()

```
void netdem::RegressionNet::AddLayer (
            LayerName layer_name,
             ...  )
```

7.90.1.2 Load()

```
void netdem::RegressionNet::Load (
            std::string const & filename,
            std::string const & label )
```

7.90.1.3 Predict()

```
arma::mat netdem::RegressionNet::Predict (
            const arma::mat & data_x )
```

7.90.1.4 ResetModel()

```
void netdem::RegressionNet::ResetModel ( )
```

7.90.1.5 Save()

```
void netdem::RegressionNet::Save (
            std::string const & filename,
            std::string const & label )
```

7.90.1.6 Train()

```
void netdem::RegressionNet::Train (
            const arma::mat & data_x,
            const arma::mat & data_y )
```

7.90.2 Member Data Documentation

7.90.2.1 batch_size

```
int netdem::RegressionNet::batch_size {32}
```

7.90.2.2 decay_rate_moment

```
double netdem::RegressionNet::decay_rate_moment {0.9}
```

7.90.2.3 decay_rate_norm

```
double netdem::RegressionNet::decay_rate_norm {0.9}
```

7.90.2.4 epochs

```
int netdem::RegressionNet::epochs {100}
```

7.90.2.5 gradient_init_param

```
double netdem::RegressionNet::gradient_init_param {1e-8}
```

7.90.2.6 model

```
mlpack::ann::FFN<mlpack::ann::MeanSquaredError<>, mlpack::ann::HeInitialization> netdem::↩
RegressionNet::model
```

7.90.2.7 step_size

```
double netdem::RegressionNet::step_size {0.01}
```

7.90.2.8 stop_tol

```
double netdem::RegressionNet::stop_tol {1e-8}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/regression_net.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/regression_net.cpp

7.91 netdem::Scene Class Reference

```
#include <scene.hpp>
```

Public Member Functions

- Scene ()
- void Init (Simulation *const sim)
- Shape * InsertShape (const Shape *const s_ptr)
- VecXT< Shape * > InsertShape (const VecXT< Shape * > &s_list)
- Particle * InsertParticle (const Particle *const p_ptr)

  insert particles
- Particle * InsertParticle (Particle const &p)
- VecXT< Particle * > InsertParticle (const VecXT< Particle * > &p_list)
- VecXT< Particle * > InsertParticle (const VecXT< Particle > &p_list)
- void InsertParticle (const BondedSpheres *const p_ptr)
- void InsertParticle (const VecXT< BondedSpheres * > &p_list)
- void InsertParticle (const VecXT< BondedSpheres > &p_list)
- void InsertParticle (const BondedVoronois *const p_ptr)
- void InsertParticle (const VecXT< BondedVoronois * > &p_list)
- void InsertParticle (const VecXT< BondedVoronois > &p_list)
- Particle * InsertDerivedParticle (Particle const *p_ptr)
- VecXT< Particle * > InsertDerivedParticle (const VecXT< Particle * > &p_list)
- Wall * InsertWall (const Wall *const w_ptr)

  insert walls
- Wall * InsertWall (Wall const &w)
- VecXT< Wall * > InsertWall (const VecXT< Wall * > &w_list)
- VecXT< Wall * > InsertWall (const VecXT< Wall > &w_list)
- void RemoveShape (Shape *s_ptr)

  remove shape
- void RemoveParticle (Particle *p_ptr)

  remove particle
- void RemoveWall (Wall *w_ptr)

  remove wall
- ContactModel * InsertContactModel (const ContactModel *const cm_ptr)

  insert contact model
- VecXT< ContactModel * > InsertContactModel (const VecXT< ContactModel * > &cm_list)
- VecXT< Shape * > GetShapes ()

  return a list of the shape pointers
- bool InScene (const Shape *const s_ptr)

  check if object exists
- bool InScene (const ContactModel *const cnt_ptr)
- void SetNumberOfMaterials (int num)
- void SetBondModel (int mat_type_1, int mat_type_2, ContactModel *const cnt_model)
- void SetBondModel (int mat_type_1, int mat_type_2, std::string const &label)
- void SetCollisionModel (int mat_type_1, int mat_type_2, ContactModel *const cnt_model)
- void SetCollisionModel (int mat_type_1, int mat_type_2, std::string const &label)
- void SetGravity (double gx, double gy, double gz)
- ContactModel * GetBondModel (Particle *p1, Particle *p2)
- ContactModel * GetBondModel (Particle *p, Wall *w)
- ContactModel * GetCollisionModel (Particle *p1, Particle *p2)
- ContactModel * GetCollisionModel (Particle *p, Wall *w)
- void AutoReadRestart (std::string const &path, int mech_cyc, int shape_info_case=0)

  restart the simulation using output files
- void ReadRestartShapes (std::string const &file)
- void ReadRestartParticles (std::string const &file)
- void ReadRestartWalls (std::string const &file)

- void ReadRestartContacts (std::string const &file)
- VecXT< ContactPP * > GetContactPPs ()

  return particle-particle contacts (including both real, proxy and ghost)
- VecXT< ContactPW * > GetContactPWs ()

  return particle-wall contacts (including both real, proxy and ghost)
- void ClearShapes ()
- void ClearParticles ()
- void ClearWalls ()
- void ClearContactModels ()
- void ClearContacts ()
- ~Scene ()

Public Attributes

- Vec3d gravity_coef {0, 0, -9.81}

  gravity
- VecXT< Wall * > wall_list
- MiniMap< int, ContactModel * > contact_model_map

  the contact models defined in the scene
- VecXT< VecXT< ContactModel * > > bond_model_table

  the material lookup table for the contact models
- VecXT< VecXT< ContactModel * > > collision_model_table
- VecXT< Particle * > particle_list

  the particles that belong to this sub-domain (scene)
- VecXT< Particle * > particle_proxy_list
- VecXT< Particle * > particle_ghost_list
- VecXT< Wall * > wall_ghost_list
- std::unordered_map< int, Particle * > particle_map
- std::unordered_map< int, Shape * > shape_map
- VecXT< Shape * > local_shape_list

Private Attributes

- Simulation * sim {nullptr}
- int max_id_particles {-1}

  current max id of the particles or shapes in the scene
- int max_id_shapes {-1}

Static Private Attributes

- static const int max_num_particles {2000000}
- static const int max_num_shapes {2000000}

7.91.1 Detailed Description

Scene behaves as a std container:

- the instances inserted is a copy of theoriginal instances. In other words, instances in the scene and the original reference instances are independent. When an instance is inserted, the method will return a pointer to the instance in the scene.
- we use id to differentiate each instance. When an instance is inserted, it will be assigned a unique id. To avoid the id confict between different sub-domains, it assumed that the largest number of availiable ids is 2,000,000, for particles and shapes, respectively. For each sub-domain, the id starts from 2,000,000 by rank, e.g. 2,000,000, 2,000,001, ... for the second processor.
- by default, particles and contacts are owned by DEMObjectPool::GetInstance(). Shapes, walls and contact models are owned by the scene. Be caution about the memery free-up issue. We can improve if one come up with a better design. Basicly, we do not want to tangle with smart pointers (we developers need to be very clear about and should have fully control on the life period of these instances).
- bottom-line: do not directly manupulate the properties (i.e., the particle, wall, contact, shapes, contact models, etc.) in the scene. (These properties are made public for easy acess.) Use method such as Insertxxx, Removexxx. In particular, proxy and ghost is maintained by mpi_manager and dem_solver in a simulation instance.
- walls and contact models are also assumed to exist in all sub-domains with the same ids.

Note and to-do: currently, the scene, dem_solver and mpi_manager are deeply coupuled with each other. Will need to re-consider the code design. e.g.,

- 1. particle and wall owns and maintains the linked lists, which is used by dem_solver.
- 2. scenes owns the particle proxy and ghost, which are maintained by the mpi_manager and dem_solver.
- 3. particle insertion and removal by the scene might destropy the linked lists for dem_solver)

7.91.2 Constructor & Destructor Documentation

7.91.2.1 Scene()

```
Scene::Scene ( )
```

7.91.2.2 ~Scene()

```
Scene::~Scene ( )
```

7.91.3 Member Function Documentation

7.91.3.1 AutoReadRestart()

```
void Scene::AutoReadRestart (
            std::string const & path,
            int mech_cyc,
            int shape_info_case = 0 )
```

restart the simulation using output files

7.91.3.2 ClearContactModels()

```
void Scene::ClearContactModels ( )
```

7.91.3.3 ClearContacts()

```
void Scene::ClearContacts ( )
```

7.91.3.4 ClearParticles()

```
void Scene::ClearParticles ( )
```

7.91.3.5 ClearShapes()

```
void Scene::ClearShapes ( )
```

7.91.3.6 ClearWalls()

```
void Scene::ClearWalls ( )
```

7.91.3.7 GetBondModel() [1/2]

```
ContactModel * Scene::GetBondModel (
            Particle * p,
            Wall * w )
```

7.91.3.8 GetBondModel() [2/2]

```
ContactModel * Scene::GetBondModel (
            Particle * p1,
            Particle * p2 )
```

7.91.3.9 GetCollisionModel() [1/2]

```
ContactModel * Scene::GetCollisionModel (
            Particle * p,
            Wall * w )
```

7.91.3.10 GetCollisionModel() [2/2]

```
ContactModel * Scene::GetCollisionModel (
            Particle * p1,
            Particle * p2 )
```

7.91.3.11 GetContactPPs()

```
VecXT< ContactPP * > Scene::GetContactPPs ( )
```

return particle-particle contacts (including both real, proxy and ghost)

7.91.3.12 GetContactPWs()

```
VecXT< ContactPW * > Scene::GetContactPWs ( )
```

return particle-wall contacts (including both real, proxy and ghost)

7.91.3.13 GetShapes()

```
VecXT< Shape * > Scene::GetShapes ( )
```

return a list of the shape pointers

7.91.3.14 Init()

```
void Scene::Init (
            Simulation *const sim )
```

7.91.3.15 InScene() [1/2]

```
bool Scene::InScene (
            const ContactModel *const cnt_ptr )
```

7.91.3.16 InScene() [2/2]

```
bool Scene::InScene (
            const Shape *const s_ptr )
```

check if object exists

7.91.3.17 InsertContactModel() [1/2]

```
ContactModel * Scene::InsertContactModel (
            const ContactModel *const cm_ptr )
```

insert contact model

7.91.3.18 InsertContactModel() [2/2]

```
VecXT< ContactModel * > Scene::InsertContactModel (
            const VecXT< ContactModel * > & cm_list )
```

7.91.3.19 InsertDerivedParticle() [1/2]

```
VecXT< Particle * > Scene::InsertDerivedParticle (
            const VecXT< Particle * > & p_list )
```

7.91.3.20 InsertDerivedParticle() [2/2]

```
Particle * Scene::InsertDerivedParticle (
            Particle const * p_ptr )
```

7.91.3.21 InsertParticle() [1/10]

```
void Scene::InsertParticle (
            const BondedSpheres *const p_ptr )
```

7.91.3.22 InsertParticle() [2/10]

```
void Scene::InsertParticle (
            const BondedVoronois *const p_ptr )
```

7.91.3.23 InsertParticle() [3/10]

```
Particle * Scene::InsertParticle (
            const Particle *const p_ptr )
```

insert particles

7.91.3.24 InsertParticle() [4/10]

```
void Scene::InsertParticle (
            const VecXT< BondedSpheres * > & p_list )
```

7.91.3.25 InsertParticle() [5/10]

```
void Scene::InsertParticle (
            const VecXT< BondedSpheres > & p_list )
```

7.91.3.26 InsertParticle() [6/10]

```
void Scene::InsertParticle (
            const VecXT< BondedVoronois * > & p_list )
```

7.91.3.27 InsertParticle() [7/10]

```
void Scene::InsertParticle (
            const VecXT< BondedVoronois > & p_list )
```

7.91.3.28 InsertParticle() [8/10]

```
VecXT< Particle * > Scene::InsertParticle (
            const VecXT< Particle * > & p_list )
```

7.91.3.29 InsertParticle() [9/10]

```
VecXT< Particle * > Scene::InsertParticle (
            const VecXT< Particle > & p_list )
```

7.91.3.30 InsertParticle() [10/10]

```
Particle * Scene::InsertParticle (
            Particle const & p )
```

7.91.3.31 InsertShape() [1/2]

```
Shape * Scene::InsertShape (
            const Shape *const s_ptr )
```

insert a shape to the scene. By default, copies of the shape with the ids corresponding to different domains. Set to_all_domains as false if a shape is created for a specific domain (e.g., due to the breakge).

7.91.3.32 InsertShape() [2/2]

```
VecXT< Shape * > Scene::InsertShape (
            const VecXT< Shape * > & s_list )
```

7.91.3.33 InsertWall() [1/4]

```
VecXT< Wall * > Scene::InsertWall (
            const VecXT< Wall * > & w_list )
```

7.91.3.34 InsertWall() [2/4]

```
VecXT< Wall * > Scene::InsertWall (
            const VecXT< Wall > & w_list )
```

7.91.3.35 InsertWall() [3/4]

```
Wall * Scene::InsertWall (
            const Wall *const w_ptr )
```

insert walls

7.91.3.36 InsertWall() [4/4]

```
Wall * Scene::InsertWall (
            Wall const & w )
```

7.91.3.37 ReadRestartContacts()

```
void Scene::ReadRestartContacts (
            std::string const & file )
```

7.91.3.38 ReadRestartParticles()

```
void Scene::ReadRestartParticles (
            std::string const & file )
```

7.91.3.39 ReadRestartShapes()

```
void Scene::ReadRestartShapes (
            std::string const & file )
```

7.91.3.40 ReadRestartWalls()

```
void Scene::ReadRestartWalls (
            std::string const & file )
```

7.91.3.41 RemoveParticle()

```
void Scene::RemoveParticle (
            Particle * p_ptr )
```

remove particle

7.91.3.42 RemoveShape()

```
void Scene::RemoveShape (
            Shape * s_ptr )
```

remove shape

7.91.3.43 RemoveWall()

```
void Scene::RemoveWall (
            Wall * w_ptr )
```

remove wall

7.91.3.44 SetBondModel() [1/2]

```
void Scene::SetBondModel (
            int mat_type_1,
            int mat_type_2,
            ContactModel *const cnt_model )
```

7.91.3.45 SetBondModel() [2/2]

```
void Scene::SetBondModel (
            int mat_type_1,
            int mat_type_2,
            std::string const & label )
```

7.91.3.46 SetCollisionModel() [1/2]

```
void Scene::SetCollisionModel (
            int mat_type_1,
            int mat_type_2,
            ContactModel *const cnt_model )
```

7.91.3.47 SetCollisionModel() [2/2]

```
void Scene::SetCollisionModel (
            int mat_type_1,
            int mat_type_2,
            std::string const & label )
```

7.91.3.48 SetGravity()

```
void Scene::SetGravity (
            double gx,
            double gy,
            double gz )
```

7.91.3.49 SetNumberOfMaterials()

```
void Scene::SetNumberOfMaterials (
            int num )
```

set number of materials. It will initialize the contact lookup table as a matrix of [number of materials] by [number of materials], with the elements of the matrix being the pointers to the contact models.

7.91.4 Member Data Documentation

7.91.4.1 bond_model_table

```
VecXT<VecXT<ContactModel *> > netdem::Scene::bond_model_table
```

the material lookup table for the contact models

7.91.4.2 collision_model_table

```
VecXT<VecXT<ContactModel *> > netdem::Scene::collision_model_table
```

7.91.4.3 contact_model_map

```
MiniMap<int, ContactModel *> netdem::Scene::contact_model_map
```

the contact models defined in the scene

7.91.4.4 gravity_coef

```
Vec3d netdem::Scene::gravity_coef {0, 0, -9.81}
```

gravity

7.91.4.5 local_shape_list

```
VecXT<Shape *> netdem::Scene::local_shape_list
```

stores the shapes that are generated locally and not yet synced with other sub-domains.

7.91.4.6 max_id_particles

```
int netdem::Scene::max_id_particles {-1}  [private]
```

current max id of the particles or shapes in the scene

7.91.4.7 max_id_shapes

```
int netdem::Scene::max_id_shapes {-1}  [private]
```

7.91.4.8 max_num_particles

```
const int netdem::Scene::max_num_particles {2000000}  [static], [private]
```

maximum number of particles that the scene could have. This variable is introduced to initialize particle index. In each core/process, the particle index starts from rank of the process multiplied by max_num_particles, so that each particle would have a unique index. max_num_particles is set to 2,000,000 according to the affordable computational cost in each core.

7.91.4.9 max_num_shapes

```
const int netdem::Scene::max_num_shapes {2000000}  [static], [private]
```

7.91.4.10 particle_ghost_list

```
VecXT<Particle *> netdem::Scene::particle_ghost_list
```

partilces that do not physically exist. They are introduced so that contacts would not have non-defined ends. For example, two particles in this domain are in contact with each other. A contact instance is created for these two particle. If one particle moves out of the domain, then the contact cannot reference to this particle. To avoid this situation, a particle ghost is created so that this contact would have defined particles. If a particle ghost does not contact with any particle or wall in this domain, it is then removed from the list.

7.91.4.11 particle_list

```
VecXT<Particle *> netdem::Scene::particle_list
```

the particles that belong to this sub-domain (scene)

7.91.4.12 particle_map

```
std::unordered_map<int, Particle *> netdem::Scene::particle_map
```

a map with the key being the particle id, and the value being the particle pointer. The map is used for MPI to reconstruct the particle references in the contact instances. For example, a contact instance has two particle pointers, while the particle pointers cannot be directly sent by MPI (as different processes have their own memery space). So, the particle id is sent, and the particle pointers in the contact are reconstructed based on the particle ids and the particle_map.

7.91.4.13 particle_proxy_list

```
VecXT<Particle *> netdem::Scene::particle_proxy_list
```

particle belongs to other domain but with its surface toughs this domain. Particle proxies could potential come in contact with particles in this domain. particle_proxy_list is updated in each DEM cycle.

7.91.4.14 shape_map

```
std::unordered_map<int, Shape *> netdem::Scene::shape_map
```

map from shape id to shape pointer. For convenience, shapes are also assumed to exist in all sub-domains after pre-initialization. If shape evolves (e.g., due to deformaiton or crushing) in one sub-domain, this new shape will be needed to transfer with the particle.

7.91.4.15 sim

```
Simulation* netdem::Scene::sim {nullptr}  [private]
```

7.91.4.16 wall_ghost_list

```
VecXT<Wall *> netdem::Scene::wall_ghost_list
```

similar to particle ghosts, walls are temporalily moved to this list if they are removed. If a wall ghost does not contact with any particle in this domain, it is then removed from the list.

7.91.4.17 wall_list

```
VecXT<Wall *> netdem::Scene::wall_list
```

for convenience, wall is assumed to exist in all sub-domains. No MPI transfer.

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/scene.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/scene.cpp

7.92 netdem::SDFCalculator Class Reference

```
#include <igl_wrapper.hpp>
```

Public Member Functions

- SDFCalculator ()
- void InitFromSTL (const VecXT< Vec3d > &vv, const VecXT< Vec3i > &ff)
- void InitFromSTL (STLModel const &stl_model)
- void Init ()
- double SignedDistance (Vec3d const &pos) const
- Vec3d SurfacePoint (Vec3d const &pos) const
- int ClosestFacet (Vec3d const &pos) const

Public Attributes

- double max_distance {0.0}

Private Attributes

- Eigen::MatrixXd vertices
- Eigen::MatrixXi facets
- Eigen::MatrixXi T
- igl::AABB< Eigen::MatrixXd, 3 > tree
- Eigen::MatrixXd FN
- Eigen::MatrixXd VN
- Eigen::MatrixXd EN
- Eigen::MatrixXi E
- Eigen::VectorXi EMAP

7.92.1 Constructor & Destructor Documentation

7.92.1.1 SDFCalculator()

```
netdem::SDFCalculator::SDFCalculator ( )
```

7.92.2 Member Function Documentation

7.92.2.1 ClosestFacet()

```
int netdem::SDFCalculator::ClosestFacet (
            Vec3d const & pos ) const
```

7.92.2.2 Init()

```
void netdem::SDFCalculator::Init ( )
```

7.92.2.3 InitFromSTL() [1/2]

```
void netdem::SDFCalculator::InitFromSTL (
            const VecXT< Vec3d > & vv,
            const VecXT< Vec3i > & ff )
```

7.92.2.4 InitFromSTL() [2/2]

```
void netdem::SDFCalculator::InitFromSTL (
            STLModel const & stl_model )
```

7.92.2.5 SignedDistance()

```
double netdem::SDFCalculator::SignedDistance (
            Vec3d const & pos ) const
```

7.92.2.6 SurfacePoint()

```
Vec3d netdem::SDFCalculator::SurfacePoint (
            Vec3d const & pos ) const
```

7.92.3 Member Data Documentation

7.92.3.1 E

```
Eigen::MatrixXi netdem::SDFCalculator::E  [private]
```

7.92.3.2 EMAP

```
Eigen::VectorXi netdem::SDFCalculator::EMAP  [private]
```

7.92.3.3 EN

```
Eigen::MatrixXd netdem::SDFCalculator::EN  [private]
```

7.92.3.4 facets

```
Eigen::MatrixXi netdem::SDFCalculator::facets  [private]
```

7.92.3.5 FN

```
Eigen::MatrixXd netdem::SDFCalculator::FN  [private]
```

7.92.3.6 max_distance

```
double netdem::SDFCalculator::max_distance {0.0}
```

7.92.3.7 T

```
Eigen::MatrixXi netdem::SDFCalculator::T  [private]
```

7.92.3.8 tree

```
igl::AABB<Eigen::MatrixXd, 3> netdem::SDFCalculator::tree  [private]
```

7.92.3.9 vertices

```
Eigen::MatrixXd netdem::SDFCalculator::vertices  [private]
```

7.92.3.10 VN

```
Eigen::MatrixXd netdem::SDFCalculator::VN  [private]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/igl_wrapper.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/igl_wrapper.cpp

7.93 netdem::DeformationAnalysis::Settings Class Reference

```
#include <deformation_analysis.hpp>
```

Public Types

- enum SolveBy { cycles , time }

Public Attributes

- double neo_k {6.94e5}
- double neo_mu {5.21e5}
- double density {500.0}
- double damp_coef {0.7}
- Vec3d gravity_coef {0.0, 0.0, -9.81}
- double timestep {1.0e-4}
- int mesh_res {20}
- bool save_by_cycles {false}
- bool save_by_time {false}
- int save_cycle_interval {0}
- double save_time_interval {0}
- std::string root_path {"tmp/out/deformation_analysis/"}
- SolveBy solve_by {SolveBy::cycles}
- int solve_cycle_interval {0}
- double solve_time_interval {0}

7.93.1 Member Enumeration Documentation

7.93.1.1 SolveBy

```
enum netdem::DeformationAnalysis::Settings::SolveBy
```

Enumerator

cycles	
time	

7.93.2 Member Data Documentation

7.93.2.1 damp_coef

```
double netdem::DeformationAnalysis::Settings::damp_coef {0.7}
```

7.93.2.2 density

```
double netdem::DeformationAnalysis::Settings::density {500.0}
```

7.93.2.3 gravity_coef

```
Vec3d netdem::DeformationAnalysis::Settings::gravity_coef {0.0, 0.0, -9.81}
```

7.93.2.4 mesh_res

```
int netdem::DeformationAnalysis::Settings::mesh_res {20}
```

7.93.2.5 neo_k

```
double netdem::DeformationAnalysis::Settings::neo_k {6.94e5}
```

7.93.2.6 neo_mu

```
double netdem::DeformationAnalysis::Settings::neo_mu {5.21e5}
```

7.93.2.7 root_path

```
std::string netdem::DeformationAnalysis::Settings::root_path {"tmp/out/deformation_analysis/"}
```

7.93.2.8 save_by_cycles

```
bool netdem::DeformationAnalysis::Settings::save_by_cycles {false}
```

7.93.2.9 save_by_time

```
bool netdem::DeformationAnalysis::Settings::save_by_time {false}
```

7.93.2.10 save_cycle_interval

```
int netdem::DeformationAnalysis::Settings::save_cycle_interval {0}
```

7.93.2.11 save_time_interval

```
double netdem::DeformationAnalysis::Settings::save_time_interval {0}
```

7.93.2.12 solve_by

```
SolveBy netdem::DeformationAnalysis::Settings::solve_by {SolveBy::cycles}
```

7.93.2.13 solve_cycle_interval

```
int netdem::DeformationAnalysis::Settings::solve_cycle_interval {0}
```

7.93.2.14 solve_time_interval

```
double netdem::DeformationAnalysis::Settings::solve_time_interval {0}
```

7.93.2.15 timestep

```
double netdem::DeformationAnalysis::Settings::timestep {1.0e-4}
```

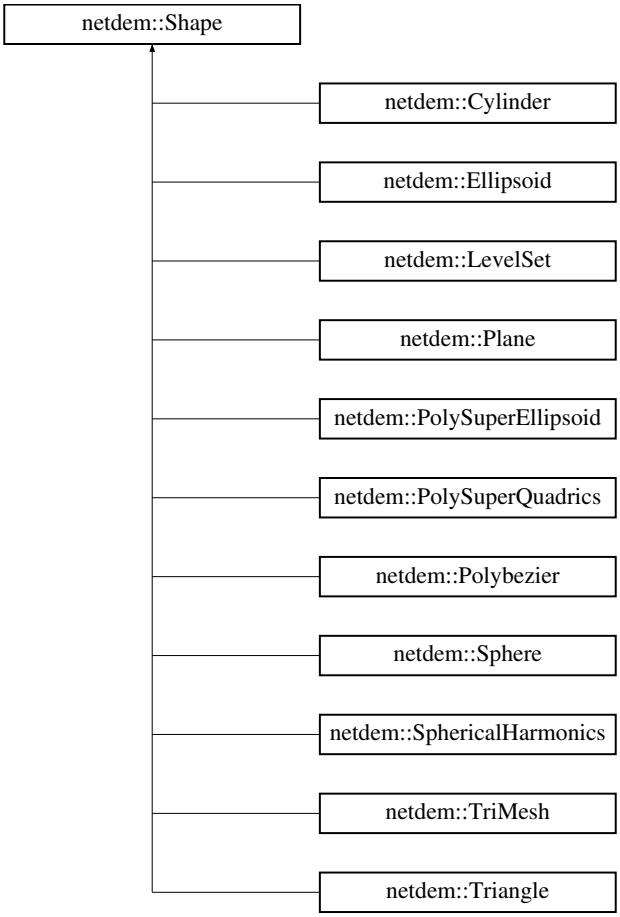
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/deformation_analysis.hpp

7.94 netdem::Shape Class Reference

`#include <shape.hpp>`

Inheritance diagram for netdem::Shape:



Public Types

- enum Type {
  none , sphere , spherical_harmonics , trimesh ,
  trimesh_convex , ellipsoid , polybezier , triangle ,
  plane , cylinder , poly_super_ellipsoid , poly_super_quadrics ,
  level_set , num_shapes }

Public Member Functions

- virtual nlohmann::json PackJson ()
- virtual void InitFromJson (nlohmann::json const &js)
- virtual void InitFromJsonFile (std::string const &js_file)
- virtual void Translate (Vec3d const &pos)
- virtual void UpdateNodes ()
- virtual void UpdateShapeProperties ()
- virtual void SetSize (double d)
- virtual Shape * Clone () const

- virtual STLModel GetSTLModel (int num_facets=400)
- virtual void SaveAsVTK (std::string const &filename)
- virtual void SaveAsSTL (std::string const &filename)
- virtual std::tuple< Vec3d, Vec3d > GetBoundAABB (Vec3d const &pos, Vec4d const &quat)
- virtual Vec3d SupportPoint (Vec3d const &dir)
- virtual VecXT< Vec3d > SupportPoints (Vec3d const &dir)
- virtual double SignedDistance (Vec3d const &pos)
- virtual Vec3d SurfacePoint (Vec3d const &pos)
- virtual bool Enclose (Vec3d const &pos)
- virtual void Print ()
- virtual ~Shape ()
- MatNd< 8, 3 > GetBoundAABBVertices ()

Public Attributes

- int id {0}
- std::string label {"default"}
- int shape_type {0}
- std::string shape_name {"shape"}
- double size {1.0}
- double volume {0.5236}
- Mat3d inertia {{{0.05236, 0, 0}, {0, 0.05236, 0}, {0, 0, 0.05236}}}
- double bound_sphere_radius {1.0}
- double skin {0.05}
- Vec3d bound_aabb_min {-0.5, -0.5, -0.5}
- Vec3d bound_aabb_max {0.5, 0.5, 0.5}
- bool use_node {false}
- int node_num {1000}
- double node_spacing {sqrt(Math::PI / 1000.0)}
- VecXT< Vec3d > nodes
- bool is_convex {true}
- bool use_customized_solver {false}

  use customized contact solver

7.94.1 Detailed Description

shape interface:

- volume, inertia: volume and inertia in the principal directions.
- size: diameter of the sphere with the same volume
- bound_sphere_radius: the radius of the smallest sphere that encloses the particle. This variable is used for broad-phase contact detection.
- skin: for broad-phase contact detection.

7.94.2 Member Enumeration Documentation

7.94.2.1 Type

```
enum netdem::Shape::Type
```

Enumerator

none	
sphere	
spherical_harmonics	
trimesh	
trimesh_convex	
ellipsoid	
polybezier	
triangle	
plane	
cylinder	
poly_super_ellipsoid	
poly_super_quadrics	
level_set	
num_shapes	

7.94.3 Constructor & Destructor Documentation

7.94.3.1 ~Shape()

```
Shape::~Shape ( )  [virtual]
```

7.94.4 Member Function Documentation

7.94.4.1 Clone()

```
Shape * Shape::Clone ( ) const  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::LevelSet, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Polybezier, netdem::Sphere, netdem::SphericalHarmonics, netdem::Triangle, and netdem::TriMesh.

7.94.4.2 Enclose()

```
bool Shape::Enclose (
            Vec3d const & pos )  [virtual]
```

Reimplemented in netdem::Triangle.

7.94.4.3 GetBoundAABB()

```
tuple< Vec3d, Vec3d > Shape::GetBoundAABB (
            Vec3d const & pos,
            Vec4d const & quat )  [virtual]
```

Reimplemented in netdem::Plane, and netdem::Triangle.

7.94.4.4 GetBoundAABBVertices()

```
MatNd< 8, 3 > Shape::GetBoundAABBVertices ( )
```

7.94.4.5 GetSTLModel()

```
STLModel Shape::GetSTLModel (
            int num_facets = 400 )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::Plane, netdem::PolySuperEllipsoid, netdem::Sphere, netdem::Triangle, netdem::TriMesh, netdem::LevelSet, netdem::PolySuperQuadrics, netdem::Polybezier, and netdem::SphericalHarmonics.

7.94.4.6 InitFromJson()

```
void Shape::InitFromJson (
            nlohmann::json const & js )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::LevelSet, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Polybezier, netdem::Sphere, netdem::SphericalHarmonics, netdem::Triangle, and netdem::TriMesh.

7.94.4.7 InitFromJsonFile()

```
void Shape::InitFromJsonFile (
            std::string const & js_file )  [virtual]
```

7.94.4.8 PackJson()

```
nlohmann::json Shape::PackJson ( )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::LevelSet, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Polybezier, netdem::Sphere, netdem::SphericalHarmonics, netdem::Triangle, and netdem::TriMesh.

7.94.4.9 Print()

```
void Shape::Print ( )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::LevelSet, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Polybezier, netdem::Sphere, and netdem::TriMesh.

7.94.4.10 SaveAsSTL()

```
void Shape::SaveAsSTL (
            std::string const & filename )  [virtual]
```

7.94.4.11 SaveAsVTK()

```
void Shape::SaveAsVTK (
            std::string const & filename )  [virtual]
```

7.94.4.12 SetSize()

```
void Shape::SetSize (
            double d )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::LevelSet, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Polybezier, netdem::SphericalHarmonics, and netdem::TriMesh.

7.94.4.13 SignedDistance()

```
double Shape::SignedDistance (
            Vec3d const & pos )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::LevelSet, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Sphere, netdem::SphericalHarmonics, netdem::Triangle, and netdem::TriMesh.

7.94.4.14 SupportPoint()

```
Vec3d Shape::SupportPoint (
            Vec3d const & dir )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Polybezier, netdem::Sphere, netdem::Triangle, and netdem::TriMesh.

7.94.4.15 SupportPoints()

```
VecXT< Vec3d > Shape::SupportPoints (
            Vec3d const & dir )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Polybezier, netdem::Sphere, netdem::Triangle, and netdem::TriMesh.

7.94.4.16 SurfacePoint()

```
Vec3d Shape::SurfacePoint (
            Vec3d const & pos )  [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::LevelSet, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Sphere, netdem::SphericalHarmonics, netdem::Triangle, and netdem::TriMesh.

7.94.4.17 Translate()

```
void Shape::Translate (
            Vec3d const & pos )  [virtual]
```

Reimplemented in netdem::Triangle.

7.94.4.18 UpdateNodes()

```
void Shape::UpdateNodes ( )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::LevelSet, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Sphere, netdem::SphericalHarmonics, netdem::Triangle, and netdem::TriMesh.

7.94.4.19 UpdateShapeProperties()

```
void Shape::UpdateShapeProperties ( )  [virtual]
```

Reimplemented in netdem::Cylinder, netdem::Ellipsoid, netdem::LevelSet, netdem::Plane, netdem::PolySuperEllipsoid, netdem::PolySuperQuadrics, netdem::Polybezier, netdem::Sphere, netdem::SphericalHarmonics, netdem::Triangle, and netdem::TriMesh.

7.94.5 Member Data Documentation

7.94.5.1 bound_aabb_max

```
Vec3d netdem::Shape::bound_aabb_max {0.5, 0.5, 0.5}
```

7.94.5.2 bound_aabb_min

```
Vec3d netdem::Shape::bound_aabb_min {-0.5, -0.5, -0.5}
```

7.94.5.3 bound_sphere_radius

```
double netdem::Shape::bound_sphere_radius {1.0}
```

7.94.5.4 id

```
int netdem::Shape::id {0}
```

7.94.5.5 inertia

```
Mat3d netdem::Shape::inertia {{{0.05236, 0, 0}, {0, 0.05236, 0}, {0, 0, 0.05236}}}
```

7.94.5.6 is_convex

```
bool netdem::Shape::is_convex {true}
```

7.94.5.7 label

```
std::string netdem::Shape::label {"default"}
```

7.94.5.8 node_num

```
int netdem::Shape::node_num {1000}
```

7.94.5.9 node_spacing

```
double netdem::Shape::node_spacing {sqrt(Math::PI / 1000.0)}
```

7.94.5.10 nodes

```
VecXT<Vec3d> netdem::Shape::nodes
```

7.94.5.11 shape_name

```
std::string netdem::Shape::shape_name {"shape"}
```

7.94.5.12 shape_type

```
int netdem::Shape::shape_type {0}
```

7.94.5.13 size

```
double netdem::Shape::size {1.0}
```

7.94.5.14 skin

```
double netdem::Shape::skin {0.05}
```

7.94.5.15 use_customized_solver

```
bool netdem::Shape::use_customized_solver {false}
```

use customized contact solver

7.94.5.16 use_node

```
bool netdem::Shape::use_node {false}
```

7.94.5.17 volume

```
double netdem::Shape::volume {0.5236}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape.cpp

7.95 netdem::ShapeFactory Class Reference

```
#include <shape_factory.hpp>
```

Static Public Member Functions

- static Shape * NewShape (std::string const &shape_name, nlohmann::json const &js)

Static Public Attributes

- static std::unordered_map< std::string, Shape::Type > shape_map

7.95.1 Member Function Documentation

7.95.1.1 NewShape()

```
Shape * ShapeFactory::NewShape (
            std::string const & shape_name,
            nlohmann::json const & js )  [static]
```

7.95.2 Member Data Documentation

7.95.2.1 shape_map

```
unordered_map< string, Shape::Type > ShapeFactory::shape_map  [static]
```

Initial value:

```
= {
    {"none", Shape::Type::none},
    {"sphere", Shape::Type::sphere},
    {"spherical_harmonics", Shape::Type::spherical_harmonics},
    {"trimesh", Shape::Type::trimesh},
    {"trimesh_convex", Shape::Type::trimesh_convex},
    {"ellipsoid", Shape::Type::ellipsoid},
    {"polybezier", Shape::Type::polybezier},
    {"triangle", Shape::Type::triangle},
    {"plane", Shape::Type::plane},
    {"cylinder", Shape::Type::cylinder},
    {"poly_super_ellipsoid", Shape::Type::poly_super_ellipsoid},
    {"poly_super_quadrics", Shape::Type::poly_super_quadrics},
    {"level_set", Shape::Type::level_set}}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_factory.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_factory.cpp

7.96 netdem::Simplex Class Reference

```
#include <gjk_simplex.hpp>
```

Public Member Functions

- Simplex ()
- Simplex (Vec3d const &a)
- Simplex (Vec3d const &a, Vec3d const &b)
- Simplex (Vec3d const &a, Vec3d const &b, Vec3d const &c)
- Simplex (Vec3d const &a, Vec3d const &b, Vec3d const &c, Vec3d const &d)
- void PushBack (Vec3d const &p)
- void PushFront (Vec3d const &p)

Public Attributes

- VecNT< Vec3d, 4 > points
- int size {0}

7.96.1 Constructor & Destructor Documentation

7.96.1.1 Simplex() [1/5]

```
netdem::Simplex::Simplex ( )  [inline]
```

7.96.1.2 Simplex() [2/5]

```
netdem::Simplex::Simplex (
            Vec3d const & a )  [inline]
```

7.96.1.3 Simplex() [3/5]

```
netdem::Simplex::Simplex (
            Vec3d const & a,
            Vec3d const & b )  [inline]
```

7.96.1.4 Simplex() [4/5]

```
netdem::Simplex::Simplex (
            Vec3d const & a,
            Vec3d const & b,
            Vec3d const & c )  [inline]
```

7.96.1.5 Simplex() [5/5]

```
netdem::Simplex::Simplex (
            Vec3d const & a,
            Vec3d const & b,
            Vec3d const & c,
            Vec3d const & d )  [inline]
```

7.96.2 Member Function Documentation

7.96.2.1 PushBack()

```
void netdem::Simplex::PushBack (
            Vec3d const & p )  [inline]
```

7.96.2.2 PushFront()

```
void netdem::Simplex::PushFront (
            Vec3d const & p )  [inline]
```

7.96.3 Member Data Documentation

7.96.3.1 points

```
VecNT<Vec3d, 4> netdem::Simplex::points
```

7.96.3.2 size

```
int netdem::Simplex::size {0}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/gjk_simplex.hpp

7.97 netdem::Simulation Class Reference

```
#include <simulation.hpp>
```

Public Member Functions

- Simulation ()
- void Init ()

  initialize with default settings
- void AutoReadRestart (std::string const &path, int mech_cyc, int shape_info_case=0)

  restart the simulation using output files
- void Run (double time)

  Run the simulaiton for a preiod.

Public Attributes

- InputProcessor input_processor

  create or modify a DEM simulation based on commands defined by json file
- DomainManager domain_manager

  manages the calculations involving domain and sub-domains
- MPIManager mpi_manager

  manages the calculations involving data exchange between sub-domains
- Scene scene
- DEMSolver dem_solver

  implments DEM algorithms and using DEM to solve the scene
- ModifierManager modifier_manager
- double mech_time {0}

  the mechanical time in the simulation world
- int mech_cycles {0}
- bool log_flag {true}

  if output the log information onto screen

7.97.1 Constructor & Destructor Documentation

7.97.1.1 Simulation()

```
Simulation::Simulation ( )
```

7.97.2 Member Function Documentation

7.97.2.1 AutoReadRestart()

```
void Simulation::AutoReadRestart (
            std::string const & path,
            int mech_cyc,
            int shape_info_case = 0 )
```

restart the simulation using output files

7.97.2.2 Init()

```
void Simulation::Init ( )
```

initialize with default settings

7.97.2.3 Run()

```
void Simulation::Run (
            double time )
```

Run the simulaiton for a preiod.

7.97.3 Member Data Documentation

7.97.3.1 dem_solver

```
DEMSolver netdem::Simulation::dem_solver
```

implments DEM algorithms and using DEM to solve the scene

7.97.3.2 domain_manager

```
DomainManager netdem::Simulation::domain_manager
```

manages the calculations involving domain and sub-domains

7.97.3.3 input_processor

```
InputProcessor netdem::Simulation::input_processor
```

create or modify a DEM simulation based on commands defined by json file

7.97.3.4 log_flag

```
bool netdem::Simulation::log_flag {true}
```

if output the log information onto screen

7.97.3.5 mech_cycles

```
int netdem::Simulation::mech_cycles {0}
```

the mechanical cycles. Each cycle accouts for a [timestep] time incretement of the simulation mechanical time.

7.97.3.6 mech_time

```
double netdem::Simulation::mech_time {0}
```

the mechanical time in the simulation world

7.97.3.7 modifier_manager

```
ModifierManager netdem::Simulation::modifier_manager
```

manage all the add-on features (i.e., those customized evaluations that are usually not hard coded in a DEM calculation cycle) for a DEM simulation

7.97.3.8 mpi_manager

```
MPIManager netdem::Simulation::mpi_manager
```

manages the calculations involving data exchange between sub-domains

7.97.3.9 scene

```
Scene netdem::Simulation::scene
```

contains and manages the basic DEM objects (e.g., shapes, particle, and wall) for a DEM simulation

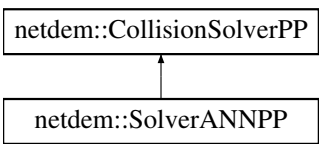
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/simulation.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/simulation.cpp

7.98 netdem::SolverANNPP Class Reference

```
#include <solver_ann_pp.hpp>
```

Inheritance diagram for netdem::SolverANNPP:



Public Member Functions

- SolverANNPP ()
- SolverANNPP (Particle ∗const p1, Particle ∗const p2)
- CollisionSolverPP ∗ Clone () const override
- void Init (std::string const &classifier_file, std::string const &regressor_file)
- void Init (Particle ∗const p1, Particle ∗const p2) override
- bool Detect () override
- void ResolveInit (ContactPP ∗const cnt, double timestep) override
- void ResolveUpdate (ContactPP ∗const cnt, double timestep) override
- void ResolveInit_LinearSpring (CollisionGeometries ∗const cnt_geoms, double timestep)
- void ResolveUpdate_LinearSpring (CollisionGeometries ∗const cnt_geoms, double timestep)
- void ResolveInit_VolumeBased (CollisionGeometries ∗const cnt_geoms, double timestep)
- void ResolveUpdate_VolumeBased (CollisionGeometries ∗const cnt_geoms, double timestep)
- void ResolveInit_PotentialBased (CollisionGeometries ∗const cnt_geoms, double timestep)
- void ResolveUpdate_PotentialBased (CollisionGeometries ∗const cnt_geoms, double timestep)
- std::tuple< double, Vec3d > Potential (Vec3d const &pos, Vec4d const &quat)
- std::tuple< Vec3d, Vec3d, Vec3d > EvaluateContactForces ()

Public Attributes

- netdem::GeneralNet classifier
- netdem::RegressionNet regressor
- bool is_initialized {false}

Private Attributes

- Shape ∗ shape_1 {nullptr}
- Shape ∗ shape_2 {nullptr}
- double bound_sphere_radius_1
- double bound_sphere_radius_2
- double scale
- Vec3d dpos_12
- Vec3d dpos_12_ref
- Vec4d dquat_12

Additional Inherited Members

7.98.1 Detailed Description

concrete geometric solver for sphere and sphere contacts.

7.98.2 Constructor & Destructor Documentation

7.98.2.1 SolverANNPP() [1/2]

```
netdem::SolverANNPP::SolverANNPP ( )
```

7.98.2.2 SolverANNPP() [2/2]

```
netdem::SolverANNPP::SolverANNPP (
            Particle *const p1,
            Particle *const p2 )
```

7.98.3 Member Function Documentation

7.98.3.1 Clone()

```
CollisionSolverPP * netdem::SolverANNPP::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.98.3.2 Detect()

```
bool netdem::SolverANNPP::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.98.3.3 EvaluateContactForces()

```
std::tuple< Vec3d, Vec3d, Vec3d > netdem::SolverANNPP::EvaluateContactForces ( )
```

7.98.3.4 Init() [1/2]

```
void netdem::SolverANNPP::Init (
            Particle *const p1,
            Particle *const p2 )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPP.

7.98.3.5 Init() [2/2]

```
void netdem::SolverANNPP::Init (
            std::string const & classifier_file,
            std::string const & regressor_file )
```

7.98.3.6 Potential()

```
std::tuple< double, Vec3d > netdem::SolverANNPP::Potential (
            Vec3d const & pos,
            Vec4d const & quat )
```

7.98.3.7 ResolveInit()

```
void netdem::SolverANNPP::ResolveInit (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.98.3.8 ResolveInit_LinearSpring()

```
void netdem::SolverANNPP::ResolveInit_LinearSpring (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.98.3.9 ResolveInit_PotentialBased()

```
void netdem::SolverANNPP::ResolveInit_PotentialBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.98.3.10 ResolveInit_VolumeBased()

```
void netdem::SolverANNPP::ResolveInit_VolumeBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.98.3.11 ResolveUpdate()

```
void netdem::SolverANNPP::ResolveUpdate (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.98.3.12 ResolveUpdate_LinearSpring()

```
void netdem::SolverANNPP::ResolveUpdate_LinearSpring (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.98.3.13 ResolveUpdate_PotentialBased()

```
void netdem::SolverANNPP::ResolveUpdate_PotentialBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.98.3.14 ResolveUpdate_VolumeBased()

```
void netdem::SolverANNPP::ResolveUpdate_VolumeBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.98.4 Member Data Documentation

7.98.4.1 bound_sphere_radius_1

```
double netdem::SolverANNPP::bound_sphere_radius_1  [private]
```

7.98.4.2 bound_sphere_radius_2

```
double netdem::SolverANNPP::bound_sphere_radius_2  [private]
```

7.98.4.3 classifier

netdem::GeneralNet netdem::SolverANNPP::classifier

7.98.4.4 dpos_12

Vec3d netdem::SolverANNPP::dpos_12 [private]

7.98.4.5 dpos_12_ref

Vec3d netdem::SolverANNPP::dpos_12_ref [private]

7.98.4.6 dquat_12

Vec4d netdem::SolverANNPP::dquat_12 [private]

7.98.4.7 is_initialized

bool netdem::SolverANNPP::is_initialized {false}

7.98.4.8 regressor

netdem::RegressionNet netdem::SolverANNPP::regressor

7.98.4.9 scale

double netdem::SolverANNPP::scale [private]

7.98.4.10 shape_1

Shape* netdem::SolverANNPP::shape_1 {nullptr} [private]

7.98.4.11 shape_2

```
Shape * netdem::SolverANNPP::shape_2 {nullptr}  [private]
```

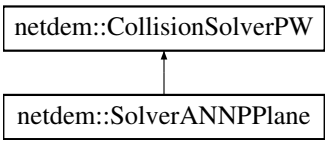
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/solver_ann_pp.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/solver_ann_pp.cpp

7.99 netdem::SolverANNPPlane Class Reference

```
#include <solver_ann_pplane.hpp>
```

Inheritance diagram for netdem::SolverANNPPlane:



Public Member Functions

- SolverANNPPlane ()
- SolverANNPPlane (Particle *const p, Wall *const w)
- CollisionSolverPW * Clone () const override
- void Init (std::string const &classifier_file, std::string const &regressor_file)
- void Init (Particle *const p, Wall *const w) override
- bool Detect () override
- void ResolveInit (ContactPW *const cnt, double timestep) override
- void ResolveUpdate (ContactPW *const cnt, double timestep) override
- void ResolveInit_LinearSpring (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate_LinearSpring (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveInit_VolumeBased (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate_VolumeBased (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveInit_PotentialBased (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate_PotentialBased (CollisionGeometries *const cnt_geoms, double timestep)
- std::tuple< double, Vec3d > Potential (double dist, Vec3d const &nn)
- std::tuple< double, Vec3d, Vec3d > EvaluateContactForces ()

Public Attributes

- netdem::GeneralNet classifier
- netdem::RegressionNet regressor
- bool is_initialized {false}

Private Attributes

- double bound_sphere_radius_1
- double dist_pc_to_plane
- double scale
- Vec3d dir_n
- Vec3d dir_n_ref

Additional Inherited Members

7.99.1 Detailed Description

concrete geometric solver for sphere and sphere contacts.

7.99.2 Constructor & Destructor Documentation

7.99.2.1 SolverANNPPlane() [1/2]

```
netdem::SolverANNPPlane::SolverANNPPlane ( )
```

7.99.2.2 SolverANNPPlane() [2/2]

```
netdem::SolverANNPPlane::SolverANNPPlane (
            Particle *const p,
            Wall *const w )
```

7.99.3 Member Function Documentation

7.99.3.1 Clone()

```
CollisionSolverPW * netdem::SolverANNPPlane::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.99.3.2 Detect()

```
bool netdem::SolverANNPPlane::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.99.3.3 EvaluateContactForces()

```
std::tuple< double, Vec3d, Vec3d > netdem::SolverANNPPlane::EvaluateContactForces ( )
```

7.99.3.4 Init() [1/2]

```
void netdem::SolverANNPPlane::Init (
            Particle *const p,
            Wall *const w )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPW.

7.99.3.5 Init() [2/2]

```
void netdem::SolverANNPPlane::Init (
            std::string const & classifier_file,
            std::string const & regressor_file )
```

7.99.3.6 Potential()

```
std::tuple< double, Vec3d > netdem::SolverANNPPlane::Potential (
            double dist,
            Vec3d const & nn )
```

7.99.3.7 ResolveInit()

```
void netdem::SolverANNPPlane::ResolveInit (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.99.3.8 ResolveInit_LinearSpring()

```
void netdem::SolverANNPPlane::ResolveInit_LinearSpring (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.99.3.9 ResolveInit_PotentialBased()

```
void netdem::SolverANNPPlane::ResolveInit_PotentialBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.99.3.10 ResolveInit_VolumeBased()

```
void netdem::SolverANNPPlane::ResolveInit_VolumeBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.99.3.11 ResolveUpdate()

```
void netdem::SolverANNPPlane::ResolveUpdate (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.99.3.12 ResolveUpdate_LinearSpring()

```
void netdem::SolverANNPPlane::ResolveUpdate_LinearSpring (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.99.3.13 ResolveUpdate_PotentialBased()

```
void netdem::SolverANNPPlane::ResolveUpdate_PotentialBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.99.3.14 ResolveUpdate_VolumeBased()

```
void netdem::SolverANNPPlane::ResolveUpdate_VolumeBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.99.4 Member Data Documentation

7.99.4.1 bound_sphere_radius_1

```
double netdem::SolverANNPPlane::bound_sphere_radius_1  [private]
```

7.99.4.2 classifier

```
netdem::GeneralNet netdem::SolverANNPPlane::classifier
```

7.99.4.3 dir_n

```
Vec3d netdem::SolverANNPPlane::dir_n  [private]
```

7.99.4.4 dir_n_ref

```
Vec3d netdem::SolverANNPPlane::dir_n_ref  [private]
```

7.99.4.5 dist_pc_to_plane

```
double netdem::SolverANNPPlane::dist_pc_to_plane  [private]
```

7.99.4.6 is_initialized

```
bool netdem::SolverANNPPlane::is_initialized {false}
```

7.99.4.7 regressor

```
netdem::RegressionNet netdem::SolverANNPPlane::regressor
```

7.99.4.8 scale

```
double netdem::SolverANNPPlane::scale  [private]
```

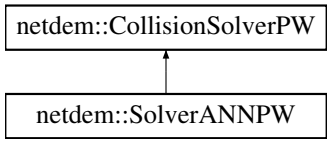
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/solver_ann_pplane.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/solver_ann_pplane.cpp

7.100 netdem::SolverANNPW Class Reference

```
#include <solver_ann_pw.hpp>
```

Inheritance diagram for netdem::SolverANNPW:



Public Member Functions

- SolverANNPW ()
- SolverANNPW (Particle *const p, Wall *const w)
- CollisionSolverPW * Clone () const override
- void Init (std::string const &classifier_file, std::string const &regressor_file)
- void Init (Particle *const p, Wall *const w) override
- bool Detect () override
- void ResolveInit (ContactPW *const cnt, double timestep) override
- void ResolveUpdate (ContactPW *const cnt, double timestep) override
- void ResolveInit_LinearSpring (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate_LinearSpring (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveInit_VolumeBased (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate_VolumeBased (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveInit_PotentialBased (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate_PotentialBased (CollisionGeometries *const cnt_geoms, double timestep)
- std::tuple< double, Vec3d > Potential (Vec3d const &pos, Vec4d const &quat)
- std::tuple< Vec3d, Vec3d, Vec3d > EvaluateContactForces ()

Public Attributes

- netdem::GeneralNet classifier
- netdem::RegressionNet regressor
- bool is_initialized {false}

Private Attributes

- Shape * shape_1 {nullptr}
- Shape * shape_2 {nullptr}
- double bound_sphere_radius_1
- double bound_sphere_radius_2
- double scale
- Vec3d dpos_12
- Vec3d dpos_12_ref
- Vec4d dquat_12

Additional Inherited Members

7.100.1 Detailed Description

concrete geometric solver for sphere and sphere contacts.

7.100.2 Constructor & Destructor Documentation

7.100.2.1 SolverANNPW() [1/2]

```
netdem::SolverANNPW::SolverANNPW ( )
```

7.100.2.2 SolverANNPW() [2/2]

```
netdem::SolverANNPW::SolverANNPW (
            Particle *const p,
            Wall *const w )
```

7.100.3 Member Function Documentation

7.100.3.1 Clone()

```
CollisionSolverPW * netdem::SolverANNPW::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.100.3.2 Detect()

```
bool netdem::SolverANNPW::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.100.3.3 EvaluateContactForces()

```
std::tuple< Vec3d, Vec3d, Vec3d > netdem::SolverANNPW::EvaluateContactForces ( )
```

7.100.3.4 Init() [1/2]

```
void netdem::SolverANNPW::Init (
            Particle *const p,
            Wall *const w )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPW.

7.100.3.5 Init() [2/2]

```
void netdem::SolverANNPW::Init (
            std::string const & classifier_file,
            std::string const & regressor_file )
```

7.100.3.6 Potential()

```
std::tuple< double, Vec3d > netdem::SolverANNPW::Potential (
            Vec3d const & pos,
            Vec4d const & quat )
```

7.100.3.7 ResolveInit()

```
void netdem::SolverANNPW::ResolveInit (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.100.3.8 ResolveInit_LinearSpring()

```
void netdem::SolverANNPW::ResolveInit_LinearSpring (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.100.3.9 ResolveInit_PotentialBased()

```
void netdem::SolverANNPW::ResolveInit_PotentialBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.100.3.10 ResolveInit_VolumeBased()

```
void netdem::SolverANNPW::ResolveInit_VolumeBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.100.3.11 ResolveUpdate()

```
void netdem::SolverANNPW::ResolveUpdate (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.100.3.12 ResolveUpdate_LinearSpring()

```
void netdem::SolverANNPW::ResolveUpdate_LinearSpring (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.100.3.13 ResolveUpdate_PotentialBased()

```
void netdem::SolverANNPW::ResolveUpdate_PotentialBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.100.3.14 ResolveUpdate_VolumeBased()

```
void netdem::SolverANNPW::ResolveUpdate_VolumeBased (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.100.4 Member Data Documentation

7.100.4.1 bound_sphere_radius_1

```
double netdem::SolverANNPW::bound_sphere_radius_1  [private]
```

7.100.4.2 bound_sphere_radius_2

```
double netdem::SolverANNPW::bound_sphere_radius_2  [private]
```

7.100.4.3 classifier

```
netdem::GeneralNet netdem::SolverANNPW::classifier
```

7.100.4.4 dpos_12

```
Vec3d netdem::SolverANNPW::dpos_12  [private]
```

7.100.4.5 dpos_12_ref

```
Vec3d netdem::SolverANNPW::dpos_12_ref  [private]
```

7.100.4.6 dquat_12

```
Vec4d netdem::SolverANNPW::dquat_12  [private]
```

7.100.4.7 is_initialized

```
bool netdem::SolverANNPW::is_initialized {false}
```

7.100.4.8 regressor

```
netdem::RegressionNet netdem::SolverANNPW::regressor
```

7.100.4.9 scale

```
double netdem::SolverANNPW::scale  [private]
```

7.100.4.10 shape_1

```
Shape* netdem::SolverANNPW::shape_1 {nullptr}  [private]
```

7.100.4.11 shape_2

```
Shape * netdem::SolverANNPW::shape_2 {nullptr}  [private]
```

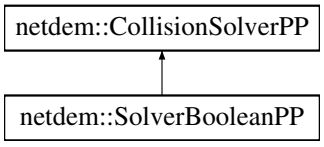
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/solver_ann_pw.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/solver_ann_pw.cpp

7.101 netdem::SolverBooleanPP Class Reference

```
#include <solver_boolean_pp.hpp>
```

Inheritance diagram for netdem::SolverBooleanPP:



Public Member Functions

- SolverBooleanPP ()
- SolverBooleanPP (Particle ∗const p1, Particle ∗const p2)
- CollisionSolverPP ∗ Clone () const override
- void Init (Particle ∗const p1, Particle ∗const p2) override
- bool Detect () override
- void ResolveInit (ContactPP ∗const cnt, double timestep) override
- void ResolveUpdate (ContactPP ∗const cnt, double timestep) override
- void ResolveInit (CollisionGeometries ∗const cnt_geoms, double timestep, const VecXT< Vec3d > &vertices, const VecXT< Vec3i > &facets, const VecXT< int > &facets_of_1or2)
- void ResolveUpdate (CollisionGeometries ∗const cnt_geoms, double timestep, const VecXT< Vec3d > &vertices, const VecXT< Vec3i > &facets, const VecXT< int > &facets_of_1or2)
- void ResolveInit_Equivalent (CollisionGeometries ∗const cnt_geoms, double timestep)
- void ResolveUpdate_Equivalent (CollisionGeometries ∗const cnt_geoms, double timestep)
- STLModel GetContactTriMesh (int id)

Protected Member Functions

- void ClearIntersectInfo ()
- void SeperateComponents ()

Protected Attributes

- double bound_sphere_radius_1
- double bound_sphere_radius_2
- Vec3d dpos_12
- STLModel ∗ stl_model_1
- STLModel ∗ stl_model_2
- VecXT< Vec3d > vertices_isct
- VecXT< Vec3i > facets_isct
- VecXT< int > facets_birth_ids
- VecXT< VecXT< Vec3i > > comp_facets_list
- VecXT< VecXT< int > > comp_facets_of_1or2_list

Additional Inherited Members

7.101.1 Detailed Description

concrete geometric solver for triangle mesh and triangle mesh contacts.

7.101.2 Constructor & Destructor Documentation

7.101.2.1 SolverBooleanPP() [1/2]

```
netdem::SolverBooleanPP::SolverBooleanPP ( )
```

7.101.2.2 SolverBooleanPP() [2/2]

```
netdem::SolverBooleanPP::SolverBooleanPP (
            Particle *const p1,
            Particle *const p2 )
```

7.101.3 Member Function Documentation

7.101.3.1 ClearIntersectInfo()

```
void netdem::SolverBooleanPP::ClearIntersectInfo ( )  [protected]
```

7.101.3.2 Clone()

```
CollisionSolverPP * netdem::SolverBooleanPP::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.101.3.3 Detect()

```
bool netdem::SolverBooleanPP::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.101.3.4 GetContactTriMesh()

```
STLModel netdem::SolverBooleanPP::GetContactTriMesh (
            int id )
```

7.101.3.5 Init()

```
void netdem::SolverBooleanPP::Init (
            Particle *const p1,
            Particle *const p2 )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPP.

7.101.3.6 ResolveInit() [1/2]

```
void netdem::SolverBooleanPP::ResolveInit (
            CollisionGeometries *const cnt_geoms,
            double timestep,
            const VecXT< Vec3d > & vertices,
            const VecXT< Vec3i > & facets,
            const VecXT< int > & facets_of_1or2 )
```

7.101.3.7 ResolveInit() [2/2]

```
void netdem::SolverBooleanPP::ResolveInit (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.101.3.8 ResolveInit_Equivalent()

```
void netdem::SolverBooleanPP::ResolveInit_Equivalent (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.101.3.9 ResolveUpdate() [1/2]

```
void netdem::SolverBooleanPP::ResolveUpdate (
            CollisionGeometries *const cnt_geoms,
            double timestep,
            const VecXT< Vec3d > & vertices,
            const VecXT< Vec3i > & facets,
            const VecXT< int > & facets_of_1or2 )
```

7.101.3.10 ResolveUpdate() [2/2]

```
void netdem::SolverBooleanPP::ResolveUpdate (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.101.3.11 ResolveUpdate_Equivalent()

```
void netdem::SolverBooleanPP::ResolveUpdate_Equivalent (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.101.3.12 SeperateComponents()

```
void netdem::SolverBooleanPP::SeperateComponents ( )  [protected]
```

7.101.4 Member Data Documentation

7.101.4.1 bound_sphere_radius_1

```
double netdem::SolverBooleanPP::bound_sphere_radius_1  [protected]
```

7.101.4.2 bound_sphere_radius_2

```
double netdem::SolverBooleanPP::bound_sphere_radius_2  [protected]
```

7.101.4.3 comp_facets_list

```
VecXT<VecXT<Vec3i> > netdem::SolverBooleanPP::comp_facets_list  [protected]
```

7.101.4.4 comp_facets_of_1or2_list

```
VecXT<VecXT<int> > netdem::SolverBooleanPP::comp_facets_of_1or2_list  [protected]
```

7.101.4.5 dpos_12

```
Vec3d netdem::SolverBooleanPP::dpos_12  [protected]
```

7.101.4.6 facets_birth_ids

```
VecXT<int> netdem::SolverBooleanPP::facets_birth_ids  [protected]
```

7.101.4.7 facets_isct

```
VecXT<Vec3i> netdem::SolverBooleanPP::facets_isct  [protected]
```

7.101.4.8 stl_model_1

```
STLModel* netdem::SolverBooleanPP::stl_model_1  [protected]
```

7.101.4.9 stl_model_2

```
STLModel * netdem::SolverBooleanPP::stl_model_2  [protected]
```

7.101.4.10 vertices_isct

```
VecXT<Vec3d> netdem::SolverBooleanPP::vertices_isct  [protected]
```

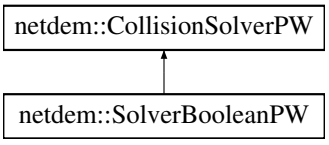
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_boolean_pp.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_boolean_pp.cpp

7.102 netdem::SolverBooleanPW Class Reference

```
#include <solver_boolean_pw.hpp>
```

Inheritance diagram for netdem::SolverBooleanPW:



Public Member Functions

- SolverBooleanPW ()
- SolverBooleanPW (Particle ∗const p, Wall ∗const w)
- CollisionSolverPW ∗ Clone () const override
- void Init (Particle ∗const p, Wall ∗const w) override
- bool Detect () override
- void ResolveInit (ContactPW ∗const cnt, double timestep) override
- void ResolveUpdate (ContactPW ∗const cnt, double timestep) override
- void ResolveInit (CollisionGeometries ∗const cnt_geoms, double timestep, const VecXT< Vec3d > &vertices, const VecXT< Vec3i > &facets, const VecXT< int > &facets_of_1or2)
- void ResolveUpdate (CollisionGeometries ∗const cnt_geoms, double timestep, const VecXT< Vec3d > &vertices, const VecXT< Vec3i > &facets, const VecXT< int > &facets_of_1or2)
- void ResolveInit_Equivalent (CollisionGeometries ∗const cnt_geoms, double timestep)
- void ResolveUpdate_Equivalent (CollisionGeometries ∗const cnt_geoms, double timestep)
- STLModel GetContactTriMesh (int id)

Protected Member Functions

- void ClearIntersectInfo ()
- void SeperateComponents ()

Protected Attributes

- double bound_sphere_radius_1
- double bound_sphere_radius_2
- Vec3d dpos_12
- STLModel ∗ stl_model_1
- STLModel ∗ stl_model_2
- VecXT< Vec3d > vertices_isct
- VecXT< Vec3i > facets_isct
- VecXT< int > facets_birth_ids
- VecXT< VecXT< Vec3i > > comp_facets_list
- VecXT< VecXT< int > > comp_facets_of_1or2_list

Additional Inherited Members

7.102.1 Detailed Description

concrete geometric solver for triangle mesh and triangle mesh contacts.

7.102.2 Constructor & Destructor Documentation

7.102.2.1 SolverBooleanPW() [1/2]

```
netdem::SolverBooleanPW::SolverBooleanPW ( )
```

7.102.2.2 SolverBooleanPW() [2/2]

```
netdem::SolverBooleanPW::SolverBooleanPW (
            Particle *const p,
            Wall *const w )
```

7.102.3 Member Function Documentation

7.102.3.1 ClearIntersectInfo()

```
void netdem::SolverBooleanPW::ClearIntersectInfo ( )  [protected]
```

7.102.3.2 Clone()

```
CollisionSolverPW * netdem::SolverBooleanPW::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.102.3.3 Detect()

```
bool netdem::SolverBooleanPW::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.102.3.4 GetContactTriMesh()

```
STLModel netdem::SolverBooleanPW::GetContactTriMesh (
            int id )
```

7.102.3.5 Init()

```
void netdem::SolverBooleanPW::Init (
            Particle *const p,
            Wall *const w )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPW.

7.102.3.6 ResolveInit() [1/2]

```
void netdem::SolverBooleanPW::ResolveInit (
            CollisionGeometries *const cnt_geoms,
            double timestep,
            const VecXT< Vec3d > & vertices,
            const VecXT< Vec3i > & facets,
            const VecXT< int > & facets_of_1or2 )
```

7.102.3.7 ResolveInit() [2/2]

```
void netdem::SolverBooleanPW::ResolveInit (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.102.3.8 ResolveInit_Equivalent()

```
void netdem::SolverBooleanPW::ResolveInit_Equivalent (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.102.3.9 ResolveUpdate() [1/2]

```
void netdem::SolverBooleanPW::ResolveUpdate (
            CollisionGeometries *const cnt_geoms,
            double timestep,
            const VecXT< Vec3d > & vertices,
            const VecXT< Vec3i > & facets,
            const VecXT< int > & facets_of_1or2 )
```

7.102.3.10 ResolveUpdate() [2/2]

```
void netdem::SolverBooleanPW::ResolveUpdate (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.102.3.11 ResolveUpdate_Equivalent()

```
void netdem::SolverBooleanPW::ResolveUpdate_Equivalent (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.102.3.12 SeperateComponents()

```
void netdem::SolverBooleanPW::SeperateComponents ( )  [protected]
```

7.102.4 Member Data Documentation

7.102.4.1 bound_sphere_radius_1

```
double netdem::SolverBooleanPW::bound_sphere_radius_1  [protected]
```

7.102.4.2 bound_sphere_radius_2

```
double netdem::SolverBooleanPW::bound_sphere_radius_2  [protected]
```

7.102.4.3 comp_facets_list

```
VecXT<VecXT<Vec3i> > netdem::SolverBooleanPW::comp_facets_list  [protected]
```

7.102.4.4 comp_facets_of_1or2_list

```
VecXT<VecXT<int> > netdem::SolverBooleanPW::comp_facets_of_1or2_list  [protected]
```

7.102.4.5 dpos_12

```
Vec3d netdem::SolverBooleanPW::dpos_12  [protected]
```

7.102.4.6 facets_birth_ids

```
VecXT<int> netdem::SolverBooleanPW::facets_birth_ids  [protected]
```

7.102.4.7 facets_isct

```
VecXT<Vec3i> netdem::SolverBooleanPW::facets_isct  [protected]
```

7.102.4.8 stl_model_1

```
STLModel* netdem::SolverBooleanPW::stl_model_1  [protected]
```

7.102.4.9 stl_model_2

```
STLModel * netdem::SolverBooleanPW::stl_model_2  [protected]
```

7.102.4.10 vertices_isct

```
VecXT<Vec3d> netdem::SolverBooleanPW::vertices_isct  [protected]
```

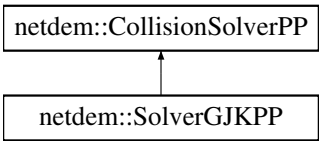
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_boolean_pw.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_boolean_pw.cpp

7.103 netdem::SolverGJKPP Class Reference

```
#include <solver_gjk_pp.hpp>
```

Inheritance diagram for netdem::SolverGJKPP:



Public Member Functions

- SolverGJKPP ()
- SolverGJKPP (Particle *const p1, Particle *const p2)
- CollisionSolverPP * Clone () const override
- void Init (Particle *const p1, Particle *const p2) override
- bool Detect () override
- void ResolveInit (ContactPP *const cnt, double timestep) override
- void ResolveUpdate (ContactPP *const cnt, double timestep) override
- void ResolveInit (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate (CollisionGeometries *const cnt_geoms, double timestep)

Public Attributes

- double erosion_ratio_initial {0.01}
- double erosion_ratio_increment {0.01}
- bool use_erosion {false}

Protected Member Functions

- bool GJK ()
- std::tuple< double, Vec3d, Vec3d > GJK_EROSION ()
- std::tuple< double, Vec3d, Vec3d > EPA ()
- std::tuple< Vec3d, bool > GetContactPoint (Vec3d const &dir)
- std::tuple< Vec3d, bool > GetContactPoint_PlaneCase (Vec3d const &dir, const VecXT< Vec3d > &pos_↩ vec_1, const VecXT< Vec3d > &pos_vec_2)
- Vec3d MinkowskiDiff (Vec3d const &dir, double erosion_ratio=0)
- void UpdateSimplex (Simplex *const s, Vec3d *const dir, double *const min_dist, bool *const cnt_flag)
- void UpdateSimplexLine (Simplex *const s, Vec3d *const dir, double *const min_dist, bool *const cnt_flag)
- void UpdateSimplexTriangle (Simplex *const s, Vec3d *const dir, double *const min_dist, bool *const cnt_↩ flag)
- void UpdateSimplexTetrahedron (Simplex *const s, Vec3d *const dir, double *const min_dist, bool *const cnt_flag)
- std::tuple< Vec3d, double > GetFacetNormal (Vec3d const &a, Vec3d const &b, Vec3d const &c)
- void GetLooseEdges (VecXT< Vec2i > *const edges, Vec3i const &facet)
- void GetIntersections (VecXT< Vec3d > *const intersects, Vec3d const &dir_n, Vec3d const &l1_p1, Vec3d const &l1_p2, Vec3d const &l2_p1, Vec3d const &l2_p2)
- void GetIntersectionsAggresive (VecXT< Vec3d > *const intersects, Vec3d const &dir_n, Vec3d const &l1↩ _p1, Vec3d const &l1_p2, Vec3d const &l2_p1, Vec3d const &l2_p2)
- void SortVertices (VecXT< Vec3d > *const pos_vec, Vec3d const &dir_n)
- bool IsInsidePolygon (VecXT< Vec3d > const &pos_vec, Vec3d const &dir_n, Vec3d const &pos)
- Vec3d GetPolygonCentroid (VecXT< Vec3d > const &pos_vec, Vec3d const &dir_n)

Protected Attributes

- Shape * shape_1 {nullptr}
- Shape * shape_2 {nullptr}
- double bound_sphere_radius_1
- double bound_sphere_radius_2
- Vec3d dpos_12
- Vec3d dpos_12_ref
- Vec4d dquat_12
- Vec4d dquat_12_conj
- Simplex simplex_after_gjk

7.103.1 Detailed Description

gjk solver for convex geometries

7.103.2 Constructor & Destructor Documentation

7.103.2.1 SolverGJKPP() [1/2]

```
netdem::SolverGJKPP::SolverGJKPP ( )
```

7.103.2.2 SolverGJKPP() [2/2]

```
netdem::SolverGJKPP::SolverGJKPP (
            Particle *const p1,
            Particle *const p2 )
```

7.103.3 Member Function Documentation

7.103.3.1 Clone()

```
CollisionSolverPP * netdem::SolverGJKPP::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.103.3.2 Detect()

```
bool netdem::SolverGJKPP::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.103.3.3 EPA()

```
tuple< double, Vec3d, Vec3d > netdem::SolverGJKPP::EPA ( )  [protected]
```

7.103.3.4 GetContactPoint()

```
tuple< Vec3d, bool > netdem::SolverGJKPP::GetContactPoint (
            Vec3d const & dir )  [protected]
```

7.103.3.5 GetContactPoint_PlaneCase()

```
tuple< Vec3d, bool > netdem::SolverGJKPP::GetContactPoint_PlaneCase (
            Vec3d const & dir,
            const VecXT< Vec3d > & pos_vec_1,
            const VecXT< Vec3d > & pos_vec_2 )  [protected]
```

7.103.3.6 GetFacetNormal()

```
std::tuple< Vec3d, double > netdem::SolverGJKPP::GetFacetNormal (
            Vec3d const & a,
            Vec3d const & b,
            Vec3d const & c )  [inline], [protected]
```

7.103.3.7 GetIntersections()

```
void netdem::SolverGJKPP::GetIntersections (
            VecXT< Vec3d > *const intersects,
            Vec3d const & dir_n,
            Vec3d const & l1_p1,
            Vec3d const & l1_p2,
            Vec3d const & l2_p1,
            Vec3d const & l2_p2 )  [protected]
```

7.103.3.8 GetIntersectionsAggresive()

```
void netdem::SolverGJKPP::GetIntersectionsAggresive (
            VecXT< Vec3d > *const intersects,
            Vec3d const & dir_n,
            Vec3d const & l1_p1,
            Vec3d const & l1_p2,
            Vec3d const & l2_p1,
            Vec3d const & l2_p2 )  [protected]
```

7.103.3.9 GetLooseEdges()

```
void netdem::SolverGJKPP::GetLooseEdges (
            VecXT< Vec2i > *const edges,
            Vec3i const & facet )  [protected]
```

7.103.3.10 GetPolygonCentroid()

```
Vec3d netdem::SolverGJKPP::GetPolygonCentroid (
            VecXT< Vec3d > const & pos_vec,
            Vec3d const & dir_n )  [protected]
```

7.103.3.11 GJK()

```
bool netdem::SolverGJKPP::GJK ( )  [protected]
```

7.103.3.12 GJK_EROSION()

```
tuple< double, Vec3d, Vec3d > netdem::SolverGJKPP::GJK_EROSION ( )  [protected]
```

7.103.3.13 Init()

```
void netdem::SolverGJKPP::Init (
            Particle *const p1,
            Particle *const p2 )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPP.

7.103.3.14 IsInsidePolygon()

```
bool netdem::SolverGJKPP::IsInsidePolygon (
            VecXT< Vec3d > const & pos_vec,
            Vec3d const & dir_n,
            Vec3d const & pos )  [protected]
```

7.103.3.15 MinkowskiDiff()

```
Vec3d netdem::SolverGJKPP::MinkowskiDiff (
            Vec3d const & dir,
            double erosion_ratio = 0 )  [inline], [protected]
```

7.103.3.16 ResolveInit() [1/2]

```
void netdem::SolverGJKPP::ResolveInit (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.103.3.17 ResolveInit() [2/2]

```
void netdem::SolverGJKPP::ResolveInit (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.103.3.18 ResolveUpdate() [1/2]

```
void netdem::SolverGJKPP::ResolveUpdate (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.103.3.19 ResolveUpdate() [2/2]

```
void netdem::SolverGJKPP::ResolveUpdate (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.103.3.20 SortVertices()

```
void netdem::SolverGJKPP::SortVertices (
            VecXT< Vec3d > *const pos_vec,
            Vec3d const & dir_n )  [protected]
```

7.103.3.21 UpdateSimplex()

```
void netdem::SolverGJKPP::UpdateSimplex (
            Simplex *const s,
            Vec3d *const dir,
            double *const min_dist,
            bool *const cnt_flag )  [protected]
```

7.103.3.22 UpdateSimplexLine()

```
void netdem::SolverGJKPP::UpdateSimplexLine (
            Simplex *const s,
            Vec3d *const dir,
            double *const min_dist,
            bool *const cnt_flag )  [protected]
```

7.103.3.23 UpdateSimplexTetrahedron()

```
void netdem::SolverGJKPP::UpdateSimplexTetrahedron (
            Simplex *const s,
            Vec3d *const dir,
            double *const min_dist,
            bool *const cnt_flag )  [protected]
```

7.103.3.24 UpdateSimplexTriangle()

```
void netdem::SolverGJKPP::UpdateSimplexTriangle (
            Simplex *const s,
            Vec3d *const dir,
            double *const min_dist,
            bool *const cnt_flag )  [protected]
```

7.103.4 Member Data Documentation

7.103.4.1 bound_sphere_radius_1

```
double netdem::SolverGJKPP::bound_sphere_radius_1  [protected]
```

7.103.4.2 bound_sphere_radius_2

```
double netdem::SolverGJKPP::bound_sphere_radius_2  [protected]
```

7.103.4.3 dpos_12

```
Vec3d netdem::SolverGJKPP::dpos_12  [protected]
```

7.103.4.4 dpos_12_ref

```
Vec3d netdem::SolverGJKPP::dpos_12_ref  [protected]
```

7.103.4.5 dquat_12

```
Vec4d netdem::SolverGJKPP::dquat_12  [protected]
```

7.103.4.6 dquat_12_conj

```
Vec4d netdem::SolverGJKPP::dquat_12_conj  [protected]
```

7.103.4.7 erosion_ratio_increment

```
double netdem::SolverGJKPP::erosion_ratio_increment {0.01}
```

7.103.4.8 erosion_ratio_initial

```
double netdem::SolverGJKPP::erosion_ratio_initial {0.01}
```

7.103.4.9 shape_1

```
Shape* netdem::SolverGJKPP::shape_1 {nullptr}  [protected]
```

7.103.4.10 shape_2

```
Shape * netdem::SolverGJKPP::shape_2 {nullptr}  [protected]
```

7.103.4.11 simplex_after_gjk

```
Simplex netdem::SolverGJKPP::simplex_after_gjk  [protected]
```

7.103.4.12 use_erosion

```
bool netdem::SolverGJKPP::use_erosion {false}
```

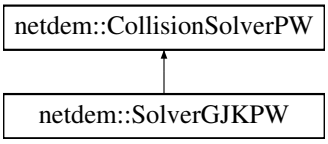
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_gjk_pp.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_gjk_pp.cpp

7.104 netdem::SolverGJKPW Class Reference

```
#include <solver_gjk_pw.hpp>
```

Inheritance diagram for netdem::SolverGJKPW:



Public Member Functions

- SolverGJKPW ()
- SolverGJKPW (Particle *const p, Wall *const w)
- CollisionSolverPW * Clone () const override
- void Init (Particle *const p, Wall *const w) override
- bool Detect () override
- void ResolveInit (ContactPW *const cnt, double timestep) override
- void ResolveUpdate (ContactPW *const cnt, double timestep) override
- void ResolveInit (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate (CollisionGeometries *const cnt_geoms, double timestep)

Public Attributes

- double erosion_ratio_initial {0.01}
- double erosion_ratio_increment {0.01}
- bool use_erosion {false}

Protected Member Functions

- bool GJK ()
- std::tuple< double, Vec3d, Vec3d > GJK_EROSION ()
- std::tuple< double, Vec3d, Vec3d > EPA ()
- std::tuple< Vec3d, bool > GetContactPoint (Vec3d const &dir)
- std::tuple< Vec3d, bool > GetContactPoint_PlaneCase (Vec3d const &dir, const VecXT< Vec3d > &pos_↩ vec_1, const VecXT< Vec3d > &pos_vec_2)
- Vec3d MinkowskiDiff (Vec3d const &dir, double erosion_ratio=0)
- void UpdateSimplex (Simplex *const s, Vec3d *const dir, double *const min_dist, bool *const cnt_flag)
- void UpdateSimplexLine (Simplex *const s, Vec3d *const dir, double *const min_dist, bool *const cnt_flag)
- void UpdateSimplexTriangle (Simplex *const s, Vec3d *const dir, double *const min_dist, bool *const cnt_↩ flag)
- void UpdateSimplexTetrahedron (Simplex *const s, Vec3d *const dir, double *const min_dist, bool *const cnt_flag)
- std::tuple< Vec3d, double > GetFacetNormal (Vec3d const &a, Vec3d const &b, Vec3d const &c)
- void GetLooseEdges (VecXT< Vec2i > *const edges, Vec3i const &facet)
- void GetIntersections (VecXT< Vec3d > *const intersects, Vec3d const &dir_n, Vec3d const &l1_p, Vec3d const &l1_w, Vec3d const &l2_p, Vec3d const &l2_w)
- void GetIntersectionsAggresive (VecXT< Vec3d > *const intersects, Vec3d const &dir_n, Vec3d const &l1_p, Vec3d const &l1_w, Vec3d const &l2_p, Vec3d const &l2_w)
- void SortVertices (VecXT< Vec3d > *const pos_vec, Vec3d const &dir_n)
- bool IsInsidePolygon (VecXT< Vec3d > const &pos_vec, Vec3d const &dir_n, Vec3d const &pos)
- Vec3d GetPolygonCentroid (VecXT< Vec3d > const &pos_vec, Vec3d const &dir_n)

Protected Attributes

- Shape * shape_1 {nullptr}
- Shape * shape_2 {nullptr}
- double bound_sphere_radius_1
- double bound_sphere_radius_2
- Vec3d dpos_12
- Vec3d dpos_12_ref
- Vec4d dquat_12
- Vec4d dquat_12_conj
- Simplex simplex_after_gjk

7.104.1 Detailed Description

gjk solver for convex geometries

7.104.2 Constructor & Destructor Documentation

7.104.2.1 SolverGJKPW() [1/2]

```
netdem::SolverGJKPW::SolverGJKPW ( )
```

7.104.2.2 SolverGJKPW() [2/2]

```
netdem::SolverGJKPW::SolverGJKPW (
            Particle *const p,
            Wall *const w )
```

7.104.3 Member Function Documentation

7.104.3.1 Clone()

```
CollisionSolverPW * netdem::SolverGJKPW::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.104.3.2 Detect()

```
bool netdem::SolverGJKPW::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.104.3.3 EPA()

```
tuple< double, Vec3d, Vec3d > netdem::SolverGJKPW::EPA ( )  [protected]
```

7.104.3.4 GetContactPoint()

```
tuple< Vec3d, bool > netdem::SolverGJKPW::GetContactPoint (
            Vec3d const & dir )  [protected]
```

7.104.3.5 GetContactPoint_PlaneCase()

```
tuple< Vec3d, bool > netdem::SolverGJKPW::GetContactPoint_PlaneCase (
            Vec3d const & dir,
            const VecXT< Vec3d > & pos_vec_1,
            const VecXT< Vec3d > & pos_vec_2 )  [protected]
```

7.104.3.6 GetFacetNormal()

```
std::tuple< Vec3d, double > netdem::SolverGJKPW::GetFacetNormal (
            Vec3d const & a,
            Vec3d const & b,
            Vec3d const & c )  [inline], [protected]
```

7.104.3.7 GetIntersections()

```
void netdem::SolverGJKPW::GetIntersections (
            VecXT< Vec3d > *const intersects,
            Vec3d const & dir_n,
            Vec3d const & l1_p,
            Vec3d const & l1_w,
            Vec3d const & l2_p,
            Vec3d const & l2_w )  [protected]
```

7.104.3.8 GetIntersectionsAggresive()

```
void netdem::SolverGJKPW::GetIntersectionsAggresive (
            VecXT< Vec3d > *const intersects,
            Vec3d const & dir_n,
            Vec3d const & l1_p,
            Vec3d const & l1_w,
            Vec3d const & l2_p,
            Vec3d const & l2_w )  [protected]
```

7.104.3.9 GetLooseEdges()

```
void netdem::SolverGJKPW::GetLooseEdges (
            VecXT< Vec2i > *const edges,
            Vec3i const & facet )  [protected]
```

7.104.3.10 GetPolygonCentroid()

```
Vec3d netdem::SolverGJKPW::GetPolygonCentroid (
            VecXT< Vec3d > const & pos_vec,
            Vec3d const & dir_n )  [protected]
```

7.104.3.11 GJK()

```
bool netdem::SolverGJKPW::GJK ( )  [protected]
```

7.104.3.12 GJK_EROSION()

```
tuple< double, Vec3d, Vec3d > netdem::SolverGJKPW::GJK_EROSION ( )  [protected]
```

7.104.3.13 Init()

```
void netdem::SolverGJKPW::Init (
            Particle *const p,
            Wall *const w )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPW.

7.104.3.14 IsInsidePolygon()

```
bool netdem::SolverGJKPW::IsInsidePolygon (
            VecXT< Vec3d > const & pos_vec,
            Vec3d const & dir_n,
            Vec3d const & pos )  [protected]
```

7.104.3.15 MinkowskiDiff()

```
Vec3d netdem::SolverGJKPW::MinkowskiDiff (
            Vec3d const & dir,
            double erosion_ratio = 0 )  [inline], [protected]
```

7.104.3.16 ResolveInit() [1/2]

```
void netdem::SolverGJKPW::ResolveInit (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.104.3.17 ResolveInit() [2/2]

```
void netdem::SolverGJKPW::ResolveInit (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.104.3.18 ResolveUpdate() [1/2]

```
void netdem::SolverGJKPW::ResolveUpdate (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.104.3.19 ResolveUpdate() [2/2]

```
void netdem::SolverGJKPW::ResolveUpdate (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.104.3.20 SortVertices()

```
void netdem::SolverGJKPW::SortVertices (
            VecXT< Vec3d > *const pos_vec,
            Vec3d const & dir_n )  [protected]
```

7.104.3.21 UpdateSimplex()

```
void netdem::SolverGJKPW::UpdateSimplex (
            Simplex *const s,
            Vec3d *const dir,
            double *const min_dist,
            bool *const cnt_flag )  [protected]
```

7.104.3.22 UpdateSimplexLine()

```
void netdem::SolverGJKPW::UpdateSimplexLine (
            Simplex *const s,
            Vec3d *const dir,
            double *const min_dist,
            bool *const cnt_flag )  [protected]
```

7.104.3.23 UpdateSimplexTetrahedron()

```
void netdem::SolverGJKPW::UpdateSimplexTetrahedron (
            Simplex *const s,
            Vec3d *const dir,
            double *const min_dist,
            bool *const cnt_flag )  [protected]
```

7.104.3.24 UpdateSimplexTriangle()

```
void netdem::SolverGJKPW::UpdateSimplexTriangle (
            Simplex *const s,
            Vec3d *const dir,
            double *const min_dist,
            bool *const cnt_flag )  [protected]
```

7.104.4 Member Data Documentation

7.104.4.1 bound_sphere_radius_1

```
double netdem::SolverGJKPW::bound_sphere_radius_1  [protected]
```

7.104.4.2 bound_sphere_radius_2

```
double netdem::SolverGJKPW::bound_sphere_radius_2  [protected]
```

7.104.4.3 dpos_12

```
Vec3d netdem::SolverGJKPW::dpos_12  [protected]
```

7.104.4.4 dpos_12_ref

```
Vec3d netdem::SolverGJKPW::dpos_12_ref  [protected]
```

7.104.4.5 dquat_12

```
Vec4d netdem::SolverGJKPW::dquat_12  [protected]
```

7.104.4.6 dquat_12_conj

```
Vec4d netdem::SolverGJKPW::dquat_12_conj  [protected]
```

7.104.4.7 erosion_ratio_increment

```
double netdem::SolverGJKPW::erosion_ratio_increment {0.01}
```

7.104.4.8 erosion_ratio_initial

```
double netdem::SolverGJKPW::erosion_ratio_initial {0.01}
```

7.104.4.9 shape_1

```
Shape* netdem::SolverGJKPW::shape_1 {nullptr}  [protected]
```

7.104.4.10 shape_2

```
Shape * netdem::SolverGJKPW::shape_2 {nullptr}  [protected]
```

7.104.4.11 simplex_after_gjk

```
Simplex netdem::SolverGJKPW::simplex_after_gjk  [protected]
```

7.104.4.12 use_erosion

```
bool netdem::SolverGJKPW::use_erosion {false}
```

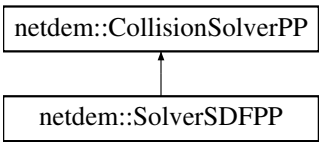
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_gjk_pw.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_gjk_pw.cpp

7.105 netdem::SolverSDFPP Class Reference

```
#include <solver_sdf_pp.hpp>
```

Inheritance diagram for netdem::SolverSDFPP:



Public Types

- enum PotentialType { linear , hertz }

Public Member Functions

- SolverSDFPP ()
- SolverSDFPP (Particle *const p1, Particle *const p2)
- CollisionSolverPP * Clone () const override
- void Init (Particle *const p1, Particle *const p2) override
- bool Detect () override
- void ResolveInit (ContactPP *const cnt, double timestep) override
- void ResolveUpdate (ContactPP *const cnt, double timestep) override
- void ResolveInit (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveInitP2ToP1 (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdateP2ToP1 (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveInitP1ToP2 (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdateP1ToP2 (CollisionGeometries *const cnt_geoms, double timestep)
- double Potential (double dist, Shape *shape)

Public Attributes

- int potential_type {PotentialType::linear}
- bool solve_two_sides {false}

Private Attributes

- bool solve_p1_to_p2 {false}
- bool solve_p2_to_p1 {false}
- double bound_sphere_radius_1
- double bound_sphere_radius_2
- Vec3d pos_1
- Vec3d dpos_12
- Vec3d dpos_12_ref
- Vec4d quat_1
- Vec4d dquat_12
- Vec3d pos_2
- Vec3d dpos_21
- Vec3d dpos_21_ref
- Vec4d quat_2
- Vec4d dquat_21
- Shape * shape_1 {nullptr}
- Shape * shape_2 {nullptr}
- VecXT< int > node_id_list
- VecXT< double > node_dist_list

Additional Inherited Members

7.105.1 Detailed Description

Signed distance field-based contact solver

7.105.2 Member Enumeration Documentation

7.105.2.1 PotentialType

```
enum netdem::SolverSDFPP::PotentialType
```

Enumerator

linear	
hertz	

7.105.3 Constructor & Destructor Documentation

7.105.3.1 SolverSDFPP() [1/2]

```
netdem::SolverSDFPP::SolverSDFPP ( )
```

7.105.3.2 SolverSDFPP() [2/2]

```
netdem::SolverSDFPP::SolverSDFPP (
            Particle *const p1,
            Particle *const p2 )
```

7.105.4 Member Function Documentation

7.105.4.1 Clone()

```
CollisionSolverPP * netdem::SolverSDFPP::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.105.4.2 Detect()

```
bool netdem::SolverSDFPP::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.105.4.3 Init()

```
void netdem::SolverSDFPP::Init (
            Particle *const p1,
            Particle *const p2 )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPP.

7.105.4.4 Potential()

```
double netdem::SolverSDFPP::Potential (
            double dist,
            Shape * shape )
```

7.105.4.5 ResolveInit() [1/2]

```
void netdem::SolverSDFPP::ResolveInit (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.105.4.6 ResolveInit() [2/2]

```
void netdem::SolverSDFPP::ResolveInit (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.105.4.7 ResolveInitP1ToP2()

```
void netdem::SolverSDFPP::ResolveInitP1ToP2 (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.105.4.8 ResolveInitP2ToP1()

```
void netdem::SolverSDFPP::ResolveInitP2ToP1 (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.105.4.9 ResolveUpdate() [1/2]

```
void netdem::SolverSDFPP::ResolveUpdate (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.105.4.10 ResolveUpdate() [2/2]

```
void netdem::SolverSDFPP::ResolveUpdate (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.105.4.11 ResolveUpdateP1ToP2()

```
void netdem::SolverSDFPP::ResolveUpdateP1ToP2 (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.105.4.12 ResolveUpdateP2ToP1()

```
void netdem::SolverSDFPP::ResolveUpdateP2ToP1 (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.105.5 Member Data Documentation

7.105.5.1 bound_sphere_radius_1

```
double netdem::SolverSDFPP::bound_sphere_radius_1  [private]
```

7.105.5.2 bound_sphere_radius_2

```
double netdem::SolverSDFPP::bound_sphere_radius_2  [private]
```

7.105.5.3 dpos_12

```
Vec3d netdem::SolverSDFPP::dpos_12  [private]
```

7.105.5.4 dpos_12_ref

```
Vec3d netdem::SolverSDFPP::dpos_12_ref  [private]
```

7.105.5.5 dpos_21

```
Vec3d netdem::SolverSDFPP::dpos_21  [private]
```

7.105.5.6 dpos_21_ref

```
Vec3d netdem::SolverSDFPP::dpos_21_ref  [private]
```

7.105.5.7 dquat_12

```
Vec4d netdem::SolverSDFPP::dquat_12  [private]
```

7.105.5.8 dquat_21

```
Vec4d netdem::SolverSDFPP::dquat_21  [private]
```

7.105.5.9 node_dist_list

```
VecXT<double> netdem::SolverSDFPP::node_dist_list  [private]
```

7.105.5.10 node_id_list

```
VecXT<int> netdem::SolverSDFPP::node_id_list  [private]
```

7.105.5.11 pos_1

```
Vec3d netdem::SolverSDFPP::pos_1  [private]
```

7.105.5.12 pos_2

```
Vec3d netdem::SolverSDFPP::pos_2  [private]
```

7.105.5.13 potential_type

```
int netdem::SolverSDFPP::potential_type {PotentialType::linear}
```

7.105.5.14 quat_1

```
Vec4d netdem::SolverSDFPP::quat_1  [private]
```

7.105.5.15 quat_2

```
Vec4d netdem::SolverSDFPP::quat_2  [private]
```

7.105.5.16 shape_1

```
Shape* netdem::SolverSDFPP::shape_1 {nullptr}  [private]
```

7.105.5.17 shape_2

```
Shape * netdem::SolverSDFPP::shape_2 {nullptr}  [private]
```

7.105.5.18 solve_p1_to_p2

```
bool netdem::SolverSDFPP::solve_p1_to_p2 {false}  [private]
```

7.105.5.19 solve_p2_to_p1

```
bool netdem::SolverSDFPP::solve_p2_to_p1 {false}  [private]
```

7.105.5.20 solve_two_sides

```
bool netdem::SolverSDFPP::solve_two_sides {false}
```

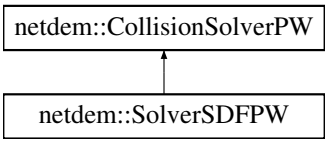
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sdf_pp.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sdf_pp.cpp

7.106 netdem::SolverSDFPW Class Reference

`#include <solver_sdf_pw.hpp>`

Inheritance diagram for netdem::SolverSDFPW:



Public Types

- enum PotentialType { linear , hertz }

Public Member Functions

- SolverSDFPW ()
- SolverSDFPW (Particle *const p, Wall *const w)
- CollisionSolverPW * Clone () const override
- void Init (Particle *const p, Wall *const w) override
- bool Detect () override
- void ResolveInit (ContactPW *const cnt, double timestep) override
- void ResolveUpdate (ContactPW *const cnt, double timestep) override
- void ResolveInit (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveInitWToP (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdateWToP (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveInitPToW (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdatePToW (CollisionGeometries *const cnt_geoms, double timestep)
- double Potential (double dist, Shape *shape)

Public Attributes

- int potential_type {PotentialType::linear}
- bool solve_two_sides {false}

Private Attributes

- bool solve_p_to_w {false}
- bool solve_w_to_p {false}
- double bound_sphere_radius_1
- double bound_sphere_radius_2
- Vec3d pos_1
- Vec3d dpos_12
- Vec3d dpos_12_ref
- Vec4d quat_1
- Vec4d dquat_12
- Vec3d pos_2
- Vec3d dpos_21
- Vec3d dpos_21_ref
- Vec4d quat_2
- Vec4d dquat_21
- Shape * shape_1 {nullptr}
- Shape * shape_2 {nullptr}
- VecXT< int > node_id_list
- VecXT< double > node_dist_list

Additional Inherited Members

7.106.1 Detailed Description

Signed distance field-based contact solver

7.106.2 Member Enumeration Documentation

7.106.2.1 PotentialType

```
enum netdem::SolverSDFPW::PotentialType
```

Enumerator

linear	
hertz	

7.106.3 Constructor & Destructor Documentation

7.106.3.1 SolverSDFPW() [1/2]

```
netdem::SolverSDFPW::SolverSDFPW ( )
```

7.106.3.2 SolverSDFPW() [2/2]

```
netdem::SolverSDFPW::SolverSDFPW (
            Particle *const p,
            Wall *const w )
```

7.106.4 Member Function Documentation

7.106.4.1 Clone()

```
CollisionSolverPW * netdem::SolverSDFPW::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.106.4.2 Detect()

```
bool netdem::SolverSDFPW::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.106.4.3 Init()

```
void netdem::SolverSDFPW::Init (
            Particle *const p,
            Wall *const w )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPW.

7.106.4.4 Potential()

```
double netdem::SolverSDFPW::Potential (
            double dist,
            Shape * shape )
```

7.106.4.5 ResolveInit() [1/2]

```
void netdem::SolverSDFPW::ResolveInit (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.106.4.6 ResolveInit() [2/2]

```
void netdem::SolverSDFPW::ResolveInit (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.106.4.7 ResolveInitPToW()

```
void netdem::SolverSDFPW::ResolveInitPToW (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.106.4.8 ResolveInitWToP()

```
void netdem::SolverSDFPW::ResolveInitWToP (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.106.4.9 ResolveUpdate() [1/2]

```
void netdem::SolverSDFPW::ResolveUpdate (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.106.4.10 ResolveUpdate() [2/2]

```
void netdem::SolverSDFPW::ResolveUpdate (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.106.4.11 ResolveUpdatePToW()

```
void netdem::SolverSDFPW::ResolveUpdatePToW (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.106.4.12 ResolveUpdateWToP()

```
void netdem::SolverSDFPW::ResolveUpdateWToP (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.106.5 Member Data Documentation

7.106.5.1 bound_sphere_radius_1

```
double netdem::SolverSDFPW::bound_sphere_radius_1  [private]
```

7.106.5.2 bound_sphere_radius_2

```
double netdem::SolverSDFPW::bound_sphere_radius_2  [private]
```

7.106.5.3 dpos_12

```
Vec3d netdem::SolverSDFPW::dpos_12  [private]
```

7.106.5.4 dpos_12_ref

```
Vec3d netdem::SolverSDFPW::dpos_12_ref  [private]
```

7.106.5.5 dpos_21

```
Vec3d netdem::SolverSDFPW::dpos_21  [private]
```

7.106.5.6 dpos_21_ref

```
Vec3d netdem::SolverSDFPW::dpos_21_ref  [private]
```

7.106.5.7 dquat_12

```
Vec4d netdem::SolverSDFPW::dquat_12  [private]
```

7.106.5.8 dquat_21

```
Vec4d netdem::SolverSDFPW::dquat_21  [private]
```

7.106.5.9 node_dist_list

```
VecXT<double> netdem::SolverSDFPW::node_dist_list  [private]
```

7.106.5.10 node_id_list

```
VecXT<int> netdem::SolverSDFPW::node_id_list  [private]
```

7.106.5.11 pos_1

```
Vec3d netdem::SolverSDFPW::pos_1  [private]
```

7.106.5.12 pos_2

```
Vec3d netdem::SolverSDFPW::pos_2  [private]
```

7.106.5.13 potential_type

```
int netdem::SolverSDFPW::potential_type {PotentialType::linear}
```

7.106.5.14 quat_1

```
Vec4d netdem::SolverSDFPW::quat_1  [private]
```

7.106.5.15 quat_2

```
Vec4d netdem::SolverSDFPW::quat_2  [private]
```

7.106.5.16 shape_1

```
Shape* netdem::SolverSDFPW::shape_1 {nullptr}  [private]
```

7.106.5.17 shape_2

```
Shape * netdem::SolverSDFPW::shape_2 {nullptr}  [private]
```

7.106.5.18 solve_p_to_w

```
bool netdem::SolverSDFPW::solve_p_to_w {false}  [private]
```

7.106.5.19 solve_two_sides

```
bool netdem::SolverSDFPW::solve_two_sides {false}
```

7.106.5.20 solve_w_to_p

```
bool netdem::SolverSDFPW::solve_w_to_p {false}  [private]
```

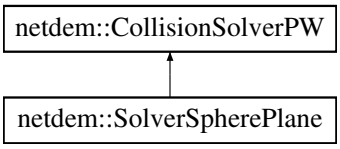
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sdf_pw.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sdf_pw.cpp

7.107 netdem::SolverSpherePlane Class Reference

```
#include <solver_sphere_plane.hpp>
```

Inheritance diagram for netdem::SolverSpherePlane:



Public Member Functions

- SolverSpherePlane ()
- SolverSpherePlane (Particle *const p, Wall *const w)
- CollisionSolverPW * Clone () const override
- void Init (Particle *const p, Wall *const w) override
- bool Detect () override
- void ResolveInit (ContactPW *const cnt, double timestep) override
- void ResolveUpdate (ContactPW *const cnt, double timestep) override
- void ResolveInit (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate (CollisionGeometries *const cnt_geoms, double timestep)

Private Attributes

- Vec3d dpos_12
- double radius_1
- double dist_pc_to_plane
- Vec3d dir_n
- Vec3d cnt_pos
- Plane * plane

Additional Inherited Members

7.107.1 Detailed Description

concrete geometric solver for sphere and plane contacts.

- len_n: overlapping depth of sphere and plane
- dist_pc_to_plane: distance from the particle centroid to the plane

7.107.2 Constructor & Destructor Documentation

7.107.2.1 SolverSpherePlane() [1/2]

```
netdem::SolverSpherePlane::SolverSpherePlane ( )
```

7.107.2.2 SolverSpherePlane() [2/2]

```
netdem::SolverSpherePlane::SolverSpherePlane (
            Particle *const p,
            Wall *const w )
```

7.107.3 Member Function Documentation

7.107.3.1 Clone()

```
CollisionSolverPW * netdem::SolverSpherePlane::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.107.3.2 Detect()

```
bool netdem::SolverSpherePlane::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.107.3.3 Init()

```
void netdem::SolverSpherePlane::Init (
            Particle *const p,
            Wall *const w )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPW.

7.107.3.4 ResolveInit() [1/2]

```
void netdem::SolverSpherePlane::ResolveInit (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.107.3.5 ResolveInit() [2/2]

```
void netdem::SolverSpherePlane::ResolveInit (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.107.3.6 ResolveUpdate() [1/2]

```
void netdem::SolverSpherePlane::ResolveUpdate (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.107.3.7 ResolveUpdate() [2/2]

```
void netdem::SolverSpherePlane::ResolveUpdate (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.107.4 Member Data Documentation

7.107.4.1 cnt_pos

```
Vec3d netdem::SolverSpherePlane::cnt_pos  [private]
```

7.107.4.2 dir_n

```
Vec3d netdem::SolverSpherePlane::dir_n  [private]
```

7.107.4.3 dist_pc_to_plane

```
double netdem::SolverSpherePlane::dist_pc_to_plane  [private]
```

7.107.4.4 dpos_12

```
Vec3d netdem::SolverSpherePlane::dpos_12  [private]
```

7.107.4.5 plane

```
Plane* netdem::SolverSpherePlane::plane  [private]
```

7.107.4.6 radius_1

```
double netdem::SolverSpherePlane::radius_1  [private]
```

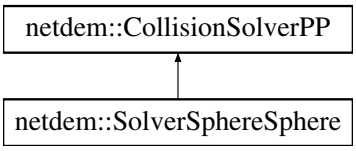
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sphere_plane.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sphere_plane.cpp

7.108 netdem::SolverSphereSphere Class Reference

`#include <solver_sphere_sphere.hpp>`

Inheritance diagram for netdem::SolverSphereSphere:



Public Member Functions

- SolverSphereSphere ()
- SolverSphereSphere (Particle *const p1, Particle *const p2)
- CollisionSolverPP * Clone () const override
- void Init (Particle *const p1, Particle *const p2) override
- bool Detect () override
- void ResolveInit (ContactPP *const cnt, double timestep) override
- void ResolveUpdate (ContactPP *const cnt, double timestep) override
- void ResolveInit (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate (CollisionGeometries *const cnt_geoms, double timestep)

Private Attributes

- double radius_1
- double radius_2
- Vec3d dpos_12

Additional Inherited Members

7.108.1 Detailed Description

concrete geometric solver for sphere and sphere contacts.

7.108.2 Constructor & Destructor Documentation

7.108.2.1 SolverSphereSphere() [1/2]

`netdem::SolverSphereSphere::SolverSphereSphere ( )`

7.108.2.2 SolverSphereSphere() [2/2]

```
netdem::SolverSphereSphere::SolverSphereSphere (
            Particle *const p1,
            Particle *const p2 )
```

7.108.3 Member Function Documentation

7.108.3.1 Clone()

```
CollisionSolverPP * netdem::SolverSphereSphere::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.108.3.2 Detect()

```
bool netdem::SolverSphereSphere::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.108.3.3 Init()

```
void netdem::SolverSphereSphere::Init (
            Particle *const p1,
            Particle *const p2 )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPP.

7.108.3.4 ResolveInit() [1/2]

```
void netdem::SolverSphereSphere::ResolveInit (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.108.3.5 ResolveInit() [2/2]

```
void netdem::SolverSphereSphere::ResolveInit (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.108.3.6 ResolveUpdate() [1/2]

```
void netdem::SolverSphereSphere::ResolveUpdate (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.108.3.7 ResolveUpdate() [2/2]

```
void netdem::SolverSphereSphere::ResolveUpdate (
            ContactPP *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPP.

7.108.4 Member Data Documentation

7.108.4.1 dpos_12

```
Vec3d netdem::SolverSphereSphere::dpos_12  [private]
```

7.108.4.2 radius_1

```
double netdem::SolverSphereSphere::radius_1  [private]
```

7.108.4.3 radius_2

```
double netdem::SolverSphereSphere::radius_2  [private]
```

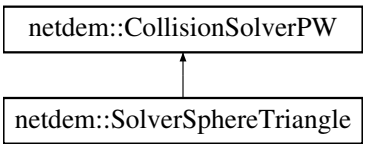
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sphere_sphere.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sphere_sphere.cpp

7.109 netdem::SolverSphereTriangle Class Reference

`#include <solver_sphere_triangle.hpp>`

Inheritance diagram for netdem::SolverSphereTriangle:



Public Member Functions

- SolverSphereTriangle ()
- SolverSphereTriangle (Particle *const p, Wall *const w)
- CollisionSolverPW * Clone () const override
- void Init (Particle *const p, Wall *const w) override
- bool Detect () override
- void ResolveInit (ContactPW *const cnt, double timestep) override
- void ResolveUpdate (ContactPW *const cnt, double timestep) override
- void ResolveInit (CollisionGeometries *const cnt_geoms, double timestep)
- void ResolveUpdate (CollisionGeometries *const cnt_geoms, double timestep)

Private Member Functions

- void UpdateLocalTriangle ()
- void ResolvePotentialContact ()
- Vec3d GetLineCircleIntersection (double cr, Vec3d const &cc, double dist_to_line, Vec3d const &dir_n_↩ cross_line, Vec3d const &v0, Vec3d const &v1)
- double GetTriangleArea (Vec3d const &v0, Vec3d const &v1, Vec3d const &v2)
- double GetCircleSegmentArea (double cr, double signed_d)

Private Attributes

- double radius_1
- double dist_pc_to_tri
- Vec3d cnt_pos
- Vec3d cnt_dir_n
- double cnt_len_n
- double cnt_weight
- Triangle triangle

Additional Inherited Members

7.109.1 Constructor & Destructor Documentation

7.109.1.1 SolverSphereTriangle() [1/2]

```
SolverSphereTriangle::SolverSphereTriangle ( )
```

7.109.1.2 SolverSphereTriangle() [2/2]

```
SolverSphereTriangle::SolverSphereTriangle (
            Particle *const p,
            Wall *const w )
```

7.109.2 Member Function Documentation

7.109.2.1 Clone()

```
CollisionSolverPW * SolverSphereTriangle::Clone ( ) const  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.109.2.2 Detect()

```
bool SolverSphereTriangle::Detect ( )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.109.2.3 GetCircleSegmentArea()

```
double SolverSphereTriangle::GetCircleSegmentArea (
            double cr,
            double signed_d )  [inline], [private]
```

7.109.2.4 GetLineCircleIntersection()

```
Vec3d SolverSphereTriangle::GetLineCircleIntersection (
            double cr,
            Vec3d const & cc,
            double dist_to_line,
            Vec3d const & dir_n_cross_line,
            Vec3d const & v0,
            Vec3d const & v1 )  [inline], [private]
```

7.109.2.5 GetTriangleArea()

```
double SolverSphereTriangle::GetTriangleArea (
            Vec3d const & v0,
            Vec3d const & v1,
            Vec3d const & v2 )  [inline], [private]
```

7.109.2.6 Init()

```
void SolverSphereTriangle::Init (
            Particle *const p,
            Wall *const w )  [override], [virtual]
```

Reimplemented from netdem::CollisionSolverPW.

7.109.2.7 ResolveInit() [1/2]

```
void SolverSphereTriangle::ResolveInit (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.109.2.8 ResolveInit() [2/2]

```
void SolverSphereTriangle::ResolveInit (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.109.2.9 ResolvePotentialContact()

```
void SolverSphereTriangle::ResolvePotentialContact ( )  [private]
```

7.109.2.10 ResolveUpdate() [1/2]

```
void SolverSphereTriangle::ResolveUpdate (
            CollisionGeometries *const cnt_geoms,
            double timestep )
```

7.109.2.11 ResolveUpdate() [2/2]

```
void SolverSphereTriangle::ResolveUpdate (
            ContactPW *const cnt,
            double timestep )  [override], [virtual]
```

Implements netdem::CollisionSolverPW.

7.109.2.12 UpdateLocalTriangle()

```
void SolverSphereTriangle::UpdateLocalTriangle ( )  [private]
```

7.109.3 Member Data Documentation

7.109.3.1 cnt_dir_n

```
Vec3d netdem::SolverSphereTriangle::cnt_dir_n  [private]
```

7.109.3.2 cnt_len_n

```
double netdem::SolverSphereTriangle::cnt_len_n  [private]
```

7.109.3.3 cnt_pos

```
Vec3d netdem::SolverSphereTriangle::cnt_pos  [private]
```

7.109.3.4 cnt_weight

```
double netdem::SolverSphereTriangle::cnt_weight  [private]
```

7.109.3.5 dist_pc_to_tri

```
double netdem::SolverSphereTriangle::dist_pc_to_tri  [private]
```

7.109.3.6 radius_1

```
double netdem::SolverSphereTriangle::radius_1  [private]
```

7.109.3.7 triangle

```
Triangle netdem::SolverSphereTriangle::triangle  [private]
```

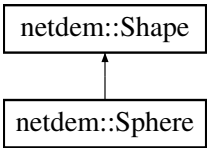
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sphere_triangle.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sphere_triangle.cpp

7.110 netdem::Sphere Class Reference

```
#include <shape_sphere.hpp>
```

Inheritance diagram for netdem::Sphere:



Public Member Functions

- Sphere ()
- Sphere (double d)
- Shape * Clone () const override
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void Init ()
- void UpdateNodes () override
- void UpdateShapeProperties () override
- STLModel GetSTLModel (int num_facets=400) override
- Vec3d SupportPoint (Vec3d const &dir) override
- VecXT< Vec3d > SupportPoints (Vec3d const &dir) override
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- void Print () override

Additional Inherited Members

7.110.1 Constructor & Destructor Documentation

7.110.1.1 Sphere() [1/2]

```
Sphere::Sphere ( )
```

7.110.1.2 Sphere() [2/2]

```
Sphere::Sphere (
            double d )
```

7.110.2 Member Function Documentation

7.110.2.1 Clone()

```
Shape * Sphere::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.110.2.2 GetSTLModel()

```
STLModel Sphere::GetSTLModel (
            int num_facets = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.110.2.3 Init()

```
void Sphere::Init ( )
```

7.110.2.4 InitFromJson()

```
void Sphere::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.110.2.5 PackJson()

```
nlohmann::json Sphere::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.110.2.6 Print()

```
void Sphere::Print ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.110.2.7 SignedDistance()

```
double Sphere::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.110.2.8 SupportPoint()

```
Vec3d Sphere::SupportPoint (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.110.2.9 SupportPoints()

```
VecXT< Vec3d > Sphere::SupportPoints (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.110.2.10 SurfacePoint()

```
Vec3d Sphere::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.110.2.11 UpdateNodes()

```
void Sphere::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.110.2.12 UpdateShapeProperties()

```
void Sphere::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

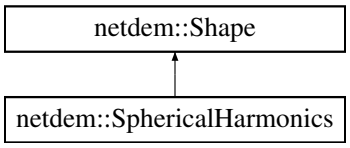
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_sphere.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_sphere.cpp

7.111 netdem::SphericalHarmonics Class Reference

```
#include <shape_spherical_harmonics.hpp>
```

Inheritance diagram for netdem::SphericalHarmonics:



Public Member Functions

- SphericalHarmonics ()
- SphericalHarmonics (int n)
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void InitFromSTL (std::string const &file)
- void InitFromSTL (STLModel const &stl_model)
- void Init ()
- void UpdateNodes () override
- void UpdateShapeProperties () override
- void SetSize (double d) override
- STLModel GetSTLModel (int res=400) override
- Shape * Clone () const override
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- double CalculateRho (double theta, double phi)
- double CalculateRho (Vec3d const &dir)

Static Public Member Functions

- static VecXT< double > CalculateYnm (double theta, double phi, int deg)
- static VecXT< VecXT< double > > CalculateYnm (const VecXT< double > &theta, const VecXT< double > &phi, int deg)
- static VecXT< double > CalculateYnm_Fast (double theta, double phi, int deg)
- static VecXT< VecXT< double > > CalculateYnm_Fast (const VecXT< double > &theta, const VecXT< double > &phi, int deg)
- static VecXT< double > CalculateYnm_Fast (Vec3d const &dir, int deg)
- static VecXT< VecXT< double > > CalculateYnm_Fast (const VecXT< Vec3d > &dir_list, int deg)

Public Attributes

- int degree {8}
- VecXT< double > a_nm

Static Private Member Functions

- static VecXT< VecXT< double > > sph_legendre_fast (double theta, int deg)
- static VecXT< VecXT< double > > sph_legendre_fast (Vec3d const &dir, int deg)

Additional Inherited Members

7.111.1 Constructor & Destructor Documentation

7.111.1.1 SphericalHarmonics() [1/2]

```
SphericalHarmonics::SphericalHarmonics ( )
```

7.111.1.2 SphericalHarmonics() [2/2]

```
SphericalHarmonics::SphericalHarmonics (
            int n )
```

7.111.2 Member Function Documentation

7.111.2.1 CalculateRho() [1/2]

```
double SphericalHarmonics::CalculateRho (
            double theta,
            double phi )
```

7.111.2.2 CalculateRho() [2/2]

```
double SphericalHarmonics::CalculateRho (
            Vec3d const & dir )
```

7.111.2.3 CalculateYnm() [1/2]

```
VecXT< VecXT< double > > SphericalHarmonics::CalculateYnm (
            const VecXT< double > & theta,
            const VecXT< double > & phi,
            int deg )  [static]
```

7.111.2.4 CalculateYnm() [2/2]

```
VecXT< double > SphericalHarmonics::CalculateYnm (
            double theta,
            double phi,
            int deg )  [static]
```

7.111.2.5 CalculateYnm_Fast() [1/4]

```
VecXT< VecXT< double > > SphericalHarmonics::CalculateYnm_Fast (
            const VecXT< double > & theta,
            const VecXT< double > & phi,
            int deg )  [static]
```

7.111.2.6 CalculateYnm_Fast() [2/4]

```
VecXT< VecXT< double > > SphericalHarmonics::CalculateYnm_Fast (
            const VecXT< Vec3d > & dir_list,
            int deg )  [static]
```

7.111.2.7 CalculateYnm_Fast() [3/4]

```
VecXT< double > SphericalHarmonics::CalculateYnm_Fast (
            double theta,
            double phi,
            int deg )  [static]
```

7.111.2.8 CalculateYnm_Fast() [4/4]

```
VecXT< double > SphericalHarmonics::CalculateYnm_Fast (
            Vec3d const & dir,
            int deg )  [static]
```

7.111.2.9 Clone()

```
Shape * SphericalHarmonics::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.111.2.10 GetSTLModel()

```
STLModel SphericalHarmonics::GetSTLModel (
            int res = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.111.2.11 Init()

```
void SphericalHarmonics::Init ( )
```

7.111.2.12 InitFromJson()

```
void SphericalHarmonics::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.111.2.13 InitFromSTL() [1/2]

```
void netdem::SphericalHarmonics::InitFromSTL (
            std::string const & file )
```

7.111.2.14 InitFromSTL() [2/2]

```
void SphericalHarmonics::InitFromSTL (
            STLModel const & stl_model )
```

7.111.2.15 PackJson()

```
nlohmann::json SphericalHarmonics::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.111.2.16 SetSize()

```
void SphericalHarmonics::SetSize (
            double d )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.111.2.17 SignedDistance()

```
double SphericalHarmonics::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.111.2.18 sph_legendre_fast() [1/2]

```
VecXT< VecXT< double > > SphericalHarmonics::sph_legendre_fast (
            double theta,
            int deg )  [static], [private]
```

7.111.2.19 sph_legendre_fast() [2/2]

```
VecXT< VecXT< double > > SphericalHarmonics::sph_legendre_fast (
            Vec3d const & dir,
            int deg )  [static], [private]
```

7.111.2.20 SurfacePoint()

```
Vec3d SphericalHarmonics::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.111.2.21 UpdateNodes()

```
void SphericalHarmonics::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.111.2.22 UpdateShapeProperties()

```
void SphericalHarmonics::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.111.3 Member Data Documentation

7.111.3.1 a_nm

```
VecXT<double> netdem::SphericalHarmonics::a_nm
```

7.111.3.2 degree

```
int netdem::SphericalHarmonics::degree {8}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_spherical_harmonics
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_spherical_harmonics

7.112 netdem::SphericalVoronoi Class Reference

```
#include <spherical_voronoi.hpp>
```

Static Public Member Functions

- static std::tuple< VecXT< Vec3d >, VecXT< VecXT< int > > > Solve (VecXT< Vec3d > const &vt_seeds)

  basic vorono. return: vt_nodes, vt_cells
- static std::tuple< VecXT< Vec3d >, VecXT< VecXT< int > > > Solve (VecXT< Vec3d > const &vt_seeds, VecXT< double > const &vt_weights)

  weighted voronoi. return: vt_nodes, vt_cells
- static VecXT< Vec3d > Solve (int num_seeds, int max_iter=10000, double tol=1.0e-4)

  centroidal voronoi. return: vt_seeds
- static VecXT< Vec3d > Solve (int num_seeds, VecXT< double > const &weights_sh_coff, int max_iter, double tol)

  assuming the weights are interplated from a spheircal harmonics function
- static void SaveAsVTK (std::string const &file, VecXT< Vec3d > const &vt_nodes, VecXT< VecXT< int > > const &vt_cells, VecXT< Vec3d > const &vt_seeds)

  save voronoi as mesh for paraview visualization

Static Private Member Functions

- static int Find (VecXT< int > const &ids, int id)
- static int Find (Vec3i const &ids, int id)
- static bool IsSharingEdge (Vec3i const &facet_i, Vec3i const &facet_j)
- static VecXT< int > FacetsContainVertex (VecXT< Vec3i > const &facets, int vid)
- static Vec3d WeightedMiddle (Vec3d const &v1, Vec3d const &v2, double w1, double w2)
- static Vec3d LineIntersection (Vec3d const &v1, Vec3d const &n1, Vec3d const &v2, Vec3d const &n2)
- static Vec3d WeightedCentroid (VecXT< Vec3d > const &vertices, VecXT< double > const &weights, Vec3i const &facet)
- static Vec3d PolyCentroid (VecXT< Vec3d > const &verts, VecXT< int > const &facet)

7.112.1 Member Function Documentation

7.112.1.1 FacetsContainVertex()

```
VecXT< int > SphericalVoronoi::FacetsContainVertex (
            VecXT< Vec3i > const & facets,
            int vid )  [static], [private]
```

7.112.1.2 Find() [1/2]

```
int SphericalVoronoi::Find (
            Vec3i const & ids,
            int id )  [static], [private]
```

7.112.1.3 Find() [2/2]

```
int SphericalVoronoi::Find (
            VecXT< int > const & ids,
            int id )  [static], [private]
```

7.112.1.4 IsSharingEdge()

```
bool SphericalVoronoi::IsSharingEdge (
            Vec3i const & facet_i,
            Vec3i const & facet_j )  [static], [private]
```

7.112.1.5 LineIntersection()

```
Vec3d SphericalVoronoi::LineIntersection (
            Vec3d const & v1,
            Vec3d const & n1,
            Vec3d const & v2,
            Vec3d const & n2 )  [static], [private]
```

7.112.1.6 PolyCentroid()

```
Vec3d SphericalVoronoi::PolyCentroid (
            VecXT< Vec3d > const & verts,
            VecXT< int > const & facet )  [static], [private]
```

7.112.1.7 SaveAsVTK()

```
void SphericalVoronoi::SaveAsVTK (
            std::string const & file,
            VecXT< Vec3d > const & vt_nodes,
            VecXT< VecXT< int > > const & vt_cells,
            VecXT< Vec3d > const & vt_seeds )  [static]
```

save voronoi as mesh for paraview visualization

7.112.1.8 Solve() [1/4]

```
VecXT< Vec3d > SphericalVoronoi::Solve (
            int num_seeds,
            int max_iter = 10000,
            double tol = 1.0e-4 )  [static]
```

centroidal voronoi. return: vt_seeds

7.112.1.9 Solve() [2/4]

```
VecXT< Vec3d > SphericalVoronoi::Solve (
            int num_seeds,
            VecXT< double > const & weights_sh_coff,
            int max_iter,
            double tol )  [static]
```

assuming the weights are interplated from a spheircal harmonics function

assuming the weights are interplated from a spheircal harmonics function return: vt_seeds

7.112.1.10 Solve() [3/4]

```
tuple< VecXT< Vec3d >, VecXT< VecXT< int > > > SphericalVoronoi::Solve (
            VecXT< Vec3d > const & vt_seeds )  [static]
```

basic vorono. return: vt_nodes, vt_cells

7.112.1.11 Solve() [4/4]

```
tuple< VecXT< Vec3d >, VecXT< VecXT< int > > > SphericalVoronoi::Solve (
            VecXT< Vec3d > const & vt_seeds,
            VecXT< double > const & vt_weights )  [static]
```

weighted voronoi. return: vt_nodes, vt_cells

7.112.1.12 WeightedCentroid()

```
Vec3d SphericalVoronoi::WeightedCentroid (
            VecXT< Vec3d > const & vertices,
            VecXT< double > const & weights,
            Vec3i const & facet )  [static], [private]
```

7.112.1.13 WeightedMiddle()

```
Vec3d SphericalVoronoi::WeightedMiddle (
            Vec3d const & v1,
            Vec3d const & v2,
            double w1,
            double w2 )  [static], [private]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/spherical_voronoi.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/spherical_voronoi.cpp

7.113 netdem::STLModel Class Reference

`#include <stl_model.hpp>`

Public Member Functions

- STLModel ()
- STLModel (const VecXT< Vec3d > &vv, const VecXT< Vec3i > &ff)
- void InitFromSTL (std::string const &file)
- void InitFromOFF (std::string const &file)
- void Translate (Vec3d const &disp)
- void Rotate (Vec4d const &quat)
- void SaveAsVTK (std::string const &file) const
- void SaveAsSTL (std::string const &file) const
- void RemoveUnreferencedVertices ()
- void RemoveDuplicateVertices ()
- void ReorientFacets ()
- void Decimate (int num_facets)
- void Standardize ()
- void SetSize (double size)
- void MakeConvex ()
- void Refine (int num_refines=1)
- void SmoothMesh (int num_trials=1)
- void MergeSTLModel (STLModel const &stl_model)
- VecXT< int > GetTriangleStrips () const

  get the triangle strip for VTK visualization
- bool IsFaceOutside (bool flip_outside=true)
- bool IsConvex ()

- bool Enclose (Vec3d const &pos) const

  check if a pos is inside the STLModel
- void Print ()

  print out information onto screen
- std::tuple< Vec3d, Vec3d > GetBoundAABB () const
- Vec3d GetCenter () const
- double GetSurfaceArea () const
- double GetVolume () const
- Mat3d GetInertia () const

Static Public Member Functions

- static Vec3d GetCenter (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f)
- static double GetSurfaceArea (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f)
- static double GetVolume (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f)
- static Mat3d GetInertia (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f)
- static bool IsConvex (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f)

Public Attributes

- VecXT< Vec3d > vertices

  N by 3 matrix, which defines the points on the 3D model surface.
- VecXT< Vec3i > facets

Private Member Functions

- int VertexIndexInFacet (const Vec3d &facet, int vertex_id)

  get the index (e.g., 0 or 1 or 2) of a vertex for a facet

7.113.1 Constructor & Destructor Documentation

7.113.1.1 STLModel() [1/2]

```
STLModel::STLModel ( )
```

7.113.1.2 STLModel() [2/2]

```
STLModel::STLModel (
            const VecXT< Vec3d > & vv,
            const VecXT< Vec3i > & ff )
```

7.113.2 Member Function Documentation

7.113.2.1 Decimate()

```
void STLModel::Decimate (
            int num_facets )
```

7.113.2.2 Enclose()

```
bool STLModel::Enclose (
            Vec3d const & pos ) const
```

check if a pos is inside the STLModel

only for convex mesh

7.113.2.3 GetBoundAABB()

```
std::tuple< Vec3d, Vec3d > STLModel::GetBoundAABB ( ) const
```

7.113.2.4 GetCenter() [1/2]

```
Vec3d STLModel::GetCenter ( ) const
```

7.113.2.5 GetCenter() [2/2]

```
Vec3d STLModel::GetCenter (
            const VecXT< Vec3d > & v,
            const VecXT< Vec3i > & f )  [static]
```

7.113.2.6 GetInertia() [1/2]

```
Mat3d STLModel::GetInertia ( ) const
```

7.113.2.7 GetInertia() [2/2]

```
Mat3d STLModel::GetInertia (
            const VecXT< Vec3d > & v,
            const VecXT< Vec3i > & f )  [static]
```

7.113.2.8 GetSurfaceArea() [1/2]

```
double STLModel::GetSurfaceArea ( ) const
```

7.113.2.9 GetSurfaceArea() [2/2]

```
double STLModel::GetSurfaceArea (
            const VecXT< Vec3d > & v,
            const VecXT< Vec3i > & f )  [static]
```

7.113.2.10 GetTriangleStrips()

```
VecXT< int > STLModel::GetTriangleStrips ( ) const
```

get the triangle strip for VTK visualization

7.113.2.11 GetVolume() [1/2]

```
double STLModel::GetVolume ( ) const
```

7.113.2.12 GetVolume() [2/2]

```
double STLModel::GetVolume (
            const VecXT< Vec3d > & v,
            const VecXT< Vec3i > & f )  [static]
```

7.113.2.13 InitFromOFF()

```
void STLModel::InitFromOFF (
            std::string const & file )
```

7.113.2.14 InitFromSTL()

```
void STLModel::InitFromSTL (
            std::string const & file )
```

7.113.2.15 IsConvex() [1/2]

```
bool STLModel::IsConvex ( )
```

7.113.2.16 IsConvex() [2/2]

```
bool STLModel::IsConvex (
            const VecXT< Vec3d > & v,
            const VecXT< Vec3i > & f )  [static]
```

7.113.2.17 IsFaceOutside()

```
bool STLModel::IsFaceOutside (
            bool flip_outside = true )
```

7.113.2.18 MakeConvex()

```
void STLModel::MakeConvex ( )
```

7.113.2.19 MergeSTLModel()

```
void STLModel::MergeSTLModel (
            STLModel const & stl_model )
```

7.113.2.20 Print()

```
void STLModel::Print ( )
```

print out information onto screen

7.113.2.21 Refine()

```
void STLModel::Refine (
            int num_refines = 1 )
```

7.113.2.22 RemoveDuplicateVertices()

```
void STLModel::RemoveDuplicateVertices ( )
```

7.113.2.23 RemoveUnreferencedVertices()

```
void STLModel::RemoveUnreferencedVertices ( )
```

7.113.2.24 ReorientFacets()

```
void STLModel::ReorientFacets ( )
```

7.113.2.25 Rotate()

```
void STLModel::Rotate (
            Vec4d const & quat )
```

7.113.2.26 SaveAsSTL()

```
void STLModel::SaveAsSTL (
            std::string const & file ) const
```

7.113.2.27 SaveAsVTK()

```
void STLModel::SaveAsVTK (
            std::string const & file ) const
```

7.113.2.28 SetSize()

```
void STLModel::SetSize (
            double size )
```

7.113.2.29 SmoothMesh()

```
void STLModel::SmoothMesh (
            int num_trials = 1 )
```

7.113.2.30 Standardize()

```
void STLModel::Standardize ( )
```

7.113.2.31 Translate()

```
void STLModel::Translate (
            Vec3d const & disp )
```

7.113.2.32 VertexIndexInFacet()

```
int STLModel::VertexIndexInFacet (
            const Vec3d & facet,
            int vertex_id )  [private]
```

get the index (e.g., 0 or 1 or 2) of a vertex for a facet

7.113.3 Member Data Documentation

7.113.3.1 facets

```
VecXT<Vec3i> netdem::STLModel::facets
```

M by 3 matrix. Each row defines a facet, with the row elements being the indices of the vertices.

7.113.3.2 vertices

```
VecXT<Vec3d> netdem::STLModel::vertices
```

N by 3 matrix, which defines the points on the 3D model surface.

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/stl_model.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/stl_model.cpp

7.114 netdem::STLReader Class Reference

```
#include <stl_reader.hpp>
```

Static Public Member Functions

- static STLModel ReadFile (std::string const &filename)

Static Private Member Functions

- static bool IsASCII (std::string const &filename)
- static STLModel ReadASCII (const char *buffer)
- static STLModel ReadBinary (const char *buffer)
- static int cpyint (const char *&p)
- static double cpydouble (const char *&p)

7.114.1 Member Function Documentation

7.114.1.1 cpydouble()

```
double STLReader::cpydouble (
            const char *& p )  [static], [private]
```

7.114.1.2 cpyint()

```
int STLReader::cpyint (
            const char *& p )  [static], [private]
```

7.114.1.3 IsASCII()

```
bool STLReader::IsASCII (
            std::string const & filename )  [static], [private]
```

7.114.1.4 ReadASCII()

```
STLModel STLReader::ReadASCII (
            const char * buffer )  [static], [private]
```

7.114.1.5 ReadBinary()

```
STLModel STLReader::ReadBinary (
            const char * buffer )  [static], [private]
```

7.114.1.6 ReadFile()

```
STLModel STLReader::ReadFile (
            std::string const & filename )  [static]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/stl_reader.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/stl_reader.cpp

7.115 netdem::TetMesh Class Reference

```
#include <tetmesh.hpp>
```

Public Member Functions

- TetMesh ()
- TetMesh (const VecXT< Vec3d > &tv, const VecXT< Vec4i > &tt)
- TetMesh (STLModel const &stl_model, double mesh_size)
- TetMesh (const VecXT< Vec3d > &vv, const VecXT< Vec3i > &ff, double mesh_size)
- void Init ()
- STLModel GetSurfaceSTL ()
- void SaveAsVTK (std::string const &file)

Public Attributes

- VecXT< Vec3d > nodes
- VecXT< Vec4i > elements
- VecXT< Vec3i > bound_facets
- VecXT< Vec2i > bound_edges
- VecXT< int > bound_nodes
- VecXT< Vec3d > surface_nodes
- VecXT< Vec3i > surface_facets
- VecXT< VecXT< int > > surface_node_linked_bounaries
- VecXT< double > bound_facet_areas

Private Member Functions

- void InitBoundary ()

7.115.1 Constructor & Destructor Documentation

7.115.1.1 TetMesh() [1/4]

```
TetMesh::TetMesh ( )
```

7.115.1.2 TetMesh() [2/4]

```
TetMesh::TetMesh (
            const VecXT< Vec3d > & tv,
            const VecXT< Vec4i > & tt )
```

7.115.1.3 TetMesh() [3/4]

```
TetMesh::TetMesh (
            STLModel const & stl_model,
            double mesh_size )
```

7.115.1.4 TetMesh() [4/4]

```
TetMesh::TetMesh (
            const VecXT< Vec3d > & vv,
            const VecXT< Vec3i > & ff,
            double mesh_size )
```

7.115.2 Member Function Documentation

7.115.2.1 GetSurfaceSTL()

```
STLModel TetMesh::GetSurfaceSTL ( )
```

7.115.2.2 Init()

```
void TetMesh::Init ( )
```

7.115.2.3 InitBoundary()

```
void TetMesh::InitBoundary ( )  [private]
```

7.115.2.4 SaveAsVTK()

```
void TetMesh::SaveAsVTK (
            std::string const & file )
```

7.115.3 Member Data Documentation

7.115.3.1 bound_edges

VecXT<Vec2i> netdem::TetMesh::bound_edges

7.115.3.2 bound_facet_areas

VecXT<double> netdem::TetMesh::bound_facet_areas

7.115.3.3 bound_facets

VecXT<Vec3i> netdem::TetMesh::bound_facets

7.115.3.4 bound_nodes

VecXT<int> netdem::TetMesh::bound_nodes

7.115.3.5 elements

VecXT<Vec4i> netdem::TetMesh::elements

7.115.3.6 nodes

VecXT<Vec3d> netdem::TetMesh::nodes

7.115.3.7 surface_facets

VecXT<Vec3i> netdem::TetMesh::surface_facets

7.115.3.8 surface_node_linked_bounaries

VecXT<VecXT<int> > netdem::TetMesh::surface_node_linked_bounaries

7.115.3.9 surface_nodes

```
VecXT<Vec3d> netdem::TetMesh::surface_nodes
```

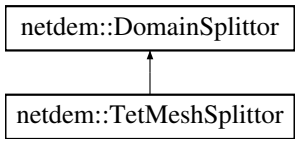
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/tetmesh.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/tetmesh.cpp

7.116 netdem::TetMeshSplittor Class Reference

```
#include <tetmesh_splittor.hpp>
```

Inheritance diagram for netdem::TetMeshSplittor:



Public Member Functions

- TetMeshSplittor ()
- void InitFromSTL (STLModel const &stl_model, int res) override
- void GetPeriDigmNodes (VecXT< Vec3d > *const nodes, VecXT< double > *const node_vols) override
- void MakePorosity (double porosity) override
- STLModel GetSTLModel () override
- STLModel GetSTLModel (const VecXT< int > &tet_indices) override

Public Attributes

- TetMesh tetmesh

7.116.1 Constructor & Destructor Documentation

7.116.1.1 TetMeshSplittor()

```
netdem::TetMeshSplittor::TetMeshSplittor ( )
```

7.116.2 Member Function Documentation

7.116.2.1 GetPeriDigmNodes()

```
void netdem::TetMeshSplittor::GetPeriDigmNodes (
            VecXT< Vec3d > *const nodes,
            VecXT< double > *const node_vols )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.116.2.2 GetSTLModel() [1/2]

```
STLModel netdem::TetMeshSplittor::GetSTLModel ( )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.116.2.3 GetSTLModel() [2/2]

```
STLModel netdem::TetMeshSplittor::GetSTLModel (
            const VecXT< int > & tet_indices )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.116.2.4 InitFromSTL()

```
void netdem::TetMeshSplittor::InitFromSTL (
            STLModel const & stl_model,
            int res )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.116.2.5 MakePorosity()

```
void netdem::TetMeshSplittor::MakePorosity (
            double porosity )  [override], [virtual]
```

Implements netdem::DomainSplittor.

7.116.3 Member Data Documentation

7.116.3.1 tetmesh

```
TetMesh netdem::TetMeshSplittor::tetmesh
```

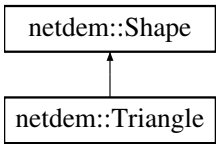
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/tetmesh_splittor.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/tetmesh_splittor.cpp

7.117 netdem::Triangle Class Reference

```
#include <shape_triangle.hpp>
```

Inheritance diagram for netdem::Triangle:



Public Member Functions

- Triangle ()
- Triangle (Vec3d const &a, Vec3d const &b, Vec3d const &c)
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void SetVertices (Vec3d const &a, Vec3d const &b, Vec3d const &c)
- void Init ()
- void Translate (Vec3d const &pos) override
- void UpdateNodes () override
- void UpdateShapeProperties () override
- Shape * Clone () const override
- STLModel GetSTLModel (int num_facets=400) override
- virtual std::tuple< Vec3d, Vec3d > GetBoundAABB (Vec3d const &pos, Vec4d const &quat) override
- Vec3d SupportPoint (Vec3d const &dir) override
- VecXT< Vec3d > SupportPoints (Vec3d const &dir) override
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- bool Enclose (Vec3d const &pos) override

Public Attributes

- Mat3d vertices
- Vec3d dir_n {0, 0, 1}

Additional Inherited Members

7.117.1 Constructor & Destructor Documentation

7.117.1.1 Triangle() [1/2]

```
Triangle::Triangle ( )
```

7.117.1.2 Triangle() [2/2]

```
Triangle::Triangle (
            Vec3d const & a,
            Vec3d const & b,
            Vec3d const & c )
```

7.117.2 Member Function Documentation

7.117.2.1 Clone()

```
Shape * Triangle::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.2 Enclose()

```
bool Triangle::Enclose (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.3 GetBoundAABB()

```
tuple< Vec3d, Vec3d > Triangle::GetBoundAABB (
            Vec3d const & pos,
            Vec4d const & quat )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.4 GetSTLModel()

```
STLModel Triangle::GetSTLModel (
            int num_facets = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.5 Init()

```
void Triangle::Init ( )
```

7.117.2.6 InitFromJson()

```
void Triangle::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.7 PackJson()

```
nlohmann::json Triangle::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.8 SetVertices()

```
void Triangle::SetVertices (
            Vec3d const & a,
            Vec3d const & b,
            Vec3d const & c )
```

7.117.2.9 SignedDistance()

```
double Triangle::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.10 SupportPoint()

```
Vec3d Triangle::SupportPoint (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.11 SupportPoints()

```
VecXT< Vec3d > Triangle::SupportPoints (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.12 SurfacePoint()

```
Vec3d Triangle::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.117.2.13 Translate()

```
void Triangle::Translate (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.14 UpdateNodes()

```
void Triangle::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.2.15 UpdateShapeProperties()

```
void Triangle::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.117.3 Member Data Documentation

7.117.3.1 dir_n

```
Vec3d netdem::Triangle::dir_n {0, 0, 1}
```

7.117.3.2 vertices

```
Mat3d netdem::Triangle::vertices
```

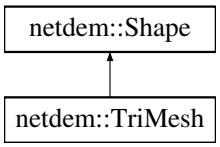
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_triangle.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_triangle.cpp

7.118 netdem::TriMesh Class Reference

```
#include <shape_trimesh.hpp>
```

Inheritance diagram for netdem::TriMesh:



Public Member Functions

- TriMesh ()
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void InitFromSTL (std::string const &file)
- void InitFromOFF (std::string const &file)
- void InitFromSTL (STLModel const &stl_model)
- void Init ()
- void AlignAxes ()
- void Decimate (int num_facets)
- void MakeConvex ()
- void UpdateNodes () override
- void UpdateShapeProperties () override
- void SetSize (double d) override
- Shape * Clone () const override
- STLModel GetSTLModel (int num_facets=400) override

- Vec3d SupportPoint (Vec3d const &dir) override
- VecXT< Vec3d > SupportPoints (Vec3d const &dir) override
- Vec3d SupportPoint_HillClimb (Vec3d const &dir)
- VecXT< Vec3d > SupportPoints_HillClimb (Vec3d const &dir)
- void SupportPoints_HillClimbCheckCoplane (int vert_id, Vec3d const &max_pos, VecXT< int > *const vert↩ _id_list, Vec3d const &dir)
- Vec3d SupportPoint_Sweep (Vec3d const &dir)
- VecXT< Vec3d > SupportPoints_Sweep (Vec3d const &dir)
- Vec3d SupportPoint_LinkedVertices (Vec3d const &dir)
- VecXT< Vec3d > SupportPoints_LinkedVertices (Vec3d const &dir)
- double SignedDistance (Vec3d const &pos) override
- Vec3d SurfacePoint (Vec3d const &pos) override
- int ClosestFacet (Vec3d const &pos)
- void Print () override
- void SaveNormalPatchesSpherical (std::string const &file)
- void SaveNormalPatchesCubic (std::string const &file)

Public Attributes

- VecXT< Vec3d > vertices
- VecXT< Vec3i > facets
- VecXT< VecXT< int > > vertices_neighs
- bool use_linked_patches {false}
- int num_cells {8}
- VecXT< VecXT< int > > linked_vertices
- SDFCalculator sdf_calculator

Private Member Functions

- void UpdateVerticesNeighs ()
- void UpdateLinkedVertices ()
- void UpdateLinkedVerticesSub (int vid)
- VecXT< Vec3d > ComputeNormalPatch (int vid)
- void SortNormalPatchVertices (VecXT< Vec3d > *const normals)
- VecXT< Vec3d > ComputeCartesianProject (const VecXT< Vec3d > &normals)
- VecXT< Vec3d > ComputeCartesianProject (const Vec3d &v1, const Vec3d &v2)
- bool ContainCorner (Vec3d const &corner, const VecXT< Vec3d > &normals)
- bool Find (const VecXT< int > &vert_id_list, int id)

Additional Inherited Members

7.118.1 Constructor & Destructor Documentation

7.118.1.1 TriMesh()

```
TriMesh::TriMesh ( )
```

7.118.2 Member Function Documentation

7.118.2.1 AlignAxes()

```
void TriMesh::AlignAxes ( )
```

7.118.2.2 Clone()

```
Shape * TriMesh::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.3 ClosestFacet()

```
int TriMesh::ClosestFacet (
            Vec3d const & pos )
```

7.118.2.4 ComputeCartesianProject() [1/2]

```
VecXT< Vec3d > TriMesh::ComputeCartesianProject (
            const Vec3d & v1,
            const Vec3d & v2 )  [private]
```

7.118.2.5 ComputeCartesianProject() [2/2]

```
VecXT< Vec3d > TriMesh::ComputeCartesianProject (
            const VecXT< Vec3d > & normals )  [private]
```

7.118.2.6 ComputeNormalPatch()

```
VecXT< Vec3d > TriMesh::ComputeNormalPatch (
            int vid )  [private]
```

7.118.2.7 ContainCorner()

```
bool TriMesh::ContainCorner (
            Vec3d const & corner,
            const VecXT< Vec3d > & normals )  [private]
```

7.118.2.8 Decimate()

```
void TriMesh::Decimate (
            int num_facets )
```

7.118.2.9 Find()

```
bool TriMesh::Find (
            const VecXT< int > & vert_id_list,
            int id )  [private]
```

7.118.2.10 GetSTLModel()

```
STLModel TriMesh::GetSTLModel (
            int num_facets = 400 )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.11 Init()

```
void TriMesh::Init ( )
```

7.118.2.12 InitFromJson()

```
void TriMesh::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.13 InitFromOFF()

```
void TriMesh::InitFromOFF (
            std::string const & file )
```

7.118.2.14 InitFromSTL() [1/2]

```
void netdem::TriMesh::InitFromSTL (
            std::string const & file )
```

7.118.2.15 InitFromSTL() [2/2]

```
void TriMesh::InitFromSTL (
            STLModel const & stl_model )
```

7.118.2.16 MakeConvex()

```
void TriMesh::MakeConvex ( )
```

7.118.2.17 PackJson()

```
nlohmann::json TriMesh::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.18 Print()

```
void TriMesh::Print ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.19 SaveNormalPatchesCubic()

```
void TriMesh::SaveNormalPatchesCubic (
            std::string const & file )
```

7.118.2.20 SaveNormalPatchesSpherical()

```
void TriMesh::SaveNormalPatchesSpherical (
            std::string const & file )
```

7.118.2.21 SetSize()

```
void TriMesh::SetSize (
            double d )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.22 SignedDistance()

```
double TriMesh::SignedDistance (
            Vec3d const & pos )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.23 SortNormalPatchVertices()

```
void TriMesh::SortNormalPatchVertices (
            VecXT< Vec3d > *const normals )  [private]
```

7.118.2.24 SupportPoint()

```
Vec3d TriMesh::SupportPoint (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.25 SupportPoint_HillClimb()

```
Vec3d TriMesh::SupportPoint_HillClimb (
            Vec3d const & dir )
```

7.118.2.26 SupportPoint_LinkedVertices()

```
Vec3d TriMesh::SupportPoint_LinkedVertices (
            Vec3d const & dir )
```

7.118.2.27 SupportPoint_Sweep()

```
Vec3d TriMesh::SupportPoint_Sweep (
            Vec3d const & dir )
```

7.118.2.28 SupportPoints()

```
VecXT< Vec3d > TriMesh::SupportPoints (
            Vec3d const & dir )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.29 SupportPoints_HillClimb()

```
VecXT< Vec3d > TriMesh::SupportPoints_HillClimb (
            Vec3d const & dir )
```

7.118.2.30 SupportPoints_HillClimbCheckCoplane()

```
void TriMesh::SupportPoints_HillClimbCheckCoplane (
            int vert_id,
            Vec3d const & max_pos,
            VecXT< int > *const vert_id_list,
            Vec3d const & dir )
```

7.118.2.31 SupportPoints_LinkedVertices()

```
VecXT< Vec3d > TriMesh::SupportPoints_LinkedVertices (
            Vec3d const & dir )
```

7.118.2.32 SupportPoints_Sweep()

```
VecXT< Vec3d > TriMesh::SupportPoints_Sweep (
            Vec3d const & dir )
```

7.118.2.33 SurfacePoint()

```
Vec3d TriMesh::SurfacePoint (
            Vec3d const & pos )  [override], [virtual]
```

calculate the surface point corrsponding to a intruding node. It will be used to compute the contact point

Reimplemented from netdem::Shape.

7.118.2.34 UpdateLinkedVertices()

```
void TriMesh::UpdateLinkedVertices ( )  [private]
```

7.118.2.35 UpdateLinkedVerticesSub()

```
void TriMesh::UpdateLinkedVerticesSub (
            int vid )  [private]
```

7.118.2.36 UpdateNodes()

```
void TriMesh::UpdateNodes ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.37 UpdateShapeProperties()

```
void TriMesh::UpdateShapeProperties ( )  [override], [virtual]
```

Reimplemented from netdem::Shape.

7.118.2.38 UpdateVerticesNeighs()

```
void TriMesh::UpdateVerticesNeighs ( )  [private]
```

7.118.3 Member Data Documentation

7.118.3.1 facets

```
VecXT<Vec3i> netdem::TriMesh::facets
```

7.118.3.2 linked_vertices

```
VecXT<VecXT<int> > netdem::TriMesh::linked_vertices
```

7.118.3.3 num_cells

```
int netdem::TriMesh::num_cells {8}
```

7.118.3.4 sdf_calculator

```
SDFCalculator netdem::TriMesh::sdf_calculator
```

7.118.3.5 use_linked_patches

```
bool netdem::TriMesh::use_linked_patches {false}
```

7.118.3.6 vertices

```
VecXT<Vec3d> netdem::TriMesh::vertices
```

7.118.3.7 vertices_neighs

```
VecXT<VecXT<int> > netdem::TriMesh::vertices_neighs
```

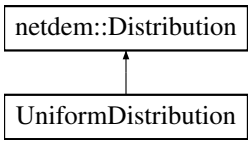
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_trimesh.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_trimesh.cpp

7.119 UniformDistribution Class Reference

```
#include <distribution_uniform.hpp>
```

Inheritance diagram for UniformDistribution:



Public Member Functions

- UniformDistribution ()
- UniformDistribution (double bound_min, double bound_max)
- double Get () override
- VecXT< double > Get (int num) override

Public Attributes

- double bound_min

  lower and upper bounds of the uniform random numbers
- double bound_max

Private Attributes

- std::mt19937 mt_eng
- std::uniform_real_distribution< double > real_dist

7.119.1 Constructor & Destructor Documentation

7.119.1.1 UniformDistribution() [1/2]

```
UniformDistribution::UniformDistribution ( )  [inline]
```

7.119.1.2 UniformDistribution() [2/2]

```
UniformDistribution::UniformDistribution (
            double bound_min,
            double bound_max )  [inline]
```

7.119.2 Member Function Documentation

7.119.2.1 Get() [1/2]

```
double UniformDistribution::Get ( )  [inline], [override], [virtual]
```

Implements netdem::Distribution.

7.119.2.2 Get() [2/2]

```
VecXT< double > UniformDistribution::Get (
            int num )  [inline], [override], [virtual]
```

Implements netdem::Distribution.

7.119.3 Member Data Documentation

7.119.3.1 bound_max

```
double UniformDistribution::bound_max
```

7.119.3.2 bound_min

```
double UniformDistribution::bound_min
```

lower and upper bounds of the uniform random numbers

7.119.3.3 mt_eng

```
std::mt19937 UniformDistribution::mt_eng  [private]
```

7.119.3.4 real_dist

```
std::uniform_real_distribution<double> UniformDistribution::real_dist  [private]
```

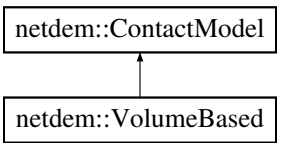
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/distribution_uniform.hpp

7.120 netdem::VolumeBased Class Reference

```
#include <model_volume_based.hpp>
```

Inheritance diagram for netdem::VolumeBased:



Public Member Functions

- VolumeBased ()
- VolumeBased (double kn, double kt, double beta, double mu)
- nlohmann::json PackJson () override
- void InitFromJson (nlohmann::json const &js) override
- void SetProperty (nlohmann::json const &js) override
- ContactModel * Clone () const override
- void EvaluateForceMoment (ContactForces *const cnt_forces, CollisionGeometries &cnt_geoms, ContactPP *const cnt, double dt) const override
- void EvaluateForceMoment (ContactForces *const cnt_forces, CollisionGeometries &cnt_geoms, ContactPW *const cnt, double dt) const override
- void Print () const override

Public Attributes

- int order {2}
- double kn {2e6}
- double kt {1e6}
- double beta {0.7}
- double mu {0.5}

Additional Inherited Members

7.120.1 Constructor & Destructor Documentation

7.120.1.1 VolumeBased() [1/2]

```
netdem::VolumeBased::VolumeBased ( )
```

7.120.1.2 VolumeBased() [2/2]

```
netdem::VolumeBased::VolumeBased (
            double kn,
            double kt,
            double beta,
            double mu )
```

7.120.2 Member Function Documentation

7.120.2.1 Clone()

```
ContactModel * netdem::VolumeBased::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.120.2.2 EvaluateForceMoment() [1/2]

```
void netdem::VolumeBased::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            CollisionGeometries & cnt_geoms,
            ContactPP *const cnt,
            double dt ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.120.2.3 EvaluateForceMoment() [2/2]

```
void netdem::VolumeBased::EvaluateForceMoment (
            ContactForces *const cnt_forces,
            CollisionGeometries & cnt_geoms,
            ContactPW *const cnt,
            double dt ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.120.2.4 InitFromJson()

```
void netdem::VolumeBased::InitFromJson (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.120.2.5 PackJson()

```
nlohmann::json netdem::VolumeBased::PackJson ( )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.120.2.6 Print()

```
void netdem::VolumeBased::Print ( ) const  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.120.2.7 SetProperty()

```
void netdem::VolumeBased::SetProperty (
            nlohmann::json const & js )  [override], [virtual]
```

Reimplemented from netdem::ContactModel.

7.120.3 Member Data Documentation

7.120.3.1 beta

```
double netdem::VolumeBased::beta {0.7}
```

7.120.3.2 kn

```
double netdem::VolumeBased::kn {2e6}
```

7.120.3.3 kt

```
double netdem::VolumeBased::kt {1e6}
```

7.120.3.4 mu

```
double netdem::VolumeBased::mu {0.5}
```

7.120.3.5 order

```
int netdem::VolumeBased::order {2}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_volume_based.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_volume_based.cpp

7.121 netdem::Voronoi Class Reference

```
#include <voronoi.hpp>
```

Static Public Member Functions

- static std::tuple< VecXT< Vec3d >, VecXT< VecXT< Vec3i > > > Solve (VecXT< Vec3d > const &vt_↩ seeds, STLModel const &stl_model, bool use_cork=true)
- static std::tuple< VecXT< Vec3d >, VecXT< VecXT< Vec3i > >, VecXT< Vec3d > > Solve (STLModel const &stl_model, int num_seeds, int max_iter=1000, double tol=1.0e-3, bool use_cork=true)
- static void SaveAsVTK (std::string const &file, VecXT< Vec3d > const &vt_nodes, VecXT< VecXT< Vec3i > > const &vt_cells, VecXT< Vec3d > const &vt_seeds)

Static Private Member Functions

- static std::tuple< VecXT< Vec3d >, VecXT< VecXT< Vec3i > > > Solve (VecXT< Vec3d > const &vt_↩ seeds, STLModel const &stl_model, SDFCalculator const &sdf_calculator, bool use_cork=true)

7.121.1 Member Function Documentation

7.121.1.1 SaveAsVTK()

```
void Voronoi::SaveAsVTK (
            std::string const & file,
            VecXT< Vec3d > const & vt_nodes,
            VecXT< VecXT< Vec3i > > const & vt_cells,
            VecXT< Vec3d > const & vt_seeds )  [static]
```

7.121.1.2 Solve() [1/3]

```
tuple< VecXT< Vec3d >, VecXT< VecXT< Vec3i > >, VecXT< Vec3d > > Voronoi::Solve (
            STLModel const & stl_model,
            int num_seeds,
            int max_iter = 1000,
            double tol = 1.0e-3,
            bool use_cork = true )  [static]
```

7.121.1.3 Solve() [2/3]

```
tuple< VecXT< Vec3d >, VecXT< VecXT< Vec3i > > > Voronoi::Solve (
            VecXT< Vec3d > const & vt_seeds,
            STLModel const & stl_model,
            bool use_cork = true )  [static]
```

7.121.1.4 Solve() [3/3]

```
tuple< VecXT< Vec3d >, VecXT< VecXT< Vec3i > > > Voronoi::Solve (
            VecXT< Vec3d > const & vt_seeds,
            STLModel const & stl_model,
            SDFCalculator const & sdf_calculator,
            bool use_cork = true )  [static], [private]
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/voronoi.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/voronoi.cpp

7.122 netdem::Wall Class Reference

`#include <wall.hpp>`

Public Member Functions

- Wall ()
- Wall (Shape *const shape)
- Wall * Clone () const
- void Init ()
- void SetShape (Shape *const shape, bool auto_adapt=false)
- void SetPosition (double pos_x, double pos_y, double pos_z)
- void SetRodrigues (double angle, double axis_x, double axis_y, double axis_z)
- void SetQuaternion (double q_0, double q_1, double q_2, double q_3)
- void SetVelocity (double v_x, double v_y, double v_z)
- void SetSpin (double spin_x, double spin_y, double spin_z)
- void SetVelocitySpin (double spin_x, double spin_y, double spin_z)
- Vec3d GetVelocity (Vec3d const &cnt_pos)
- void AddForce (const Vec3d &f)
- void AddMoment (const Vec3d &m)
- void AddForceAtomic (const Vec3d &f)
- void AddMomentAtomic (const Vec3d &m)
- void ClearForce ()
- void ClearMoment ()
- void ApplyContactForce (ContactPW const *cnt)
- void UpdateContactForce ()
- void UpdateMotion (double timestep)
- void UpdateMotion (const Vec3d &v, const Vec3d &s, double timestep)
- void UpdateMotion (const Vec3d &dpos, const Vec4d &dquat)
- void UpdateBound ()
- void ClearLinkedCells ()
- void ClearLinkedNeighs ()
- void BuildContactRef ()
- void ClearContactRef ()
- void UpdateLinkedCells (DomainManager *const dm)
- void UpdateLinkedNeighs (DomainManager *const dm)
- VecXT< ContactPW * > GetContactPWs ()
- void UpdateSTLModel ()
- void SaveAsVTK (std::string const &filename)
- void Print ()
- ~Wall ()
- int FindLinked (Particle *const p)
- int FindContactRef (Particle *const p)

Public Attributes

- int id {0}
- std::string label {"default"}
- Shape * shape
- int material_type {0}
- bool enable_rotation {false}
- bool enable_bound_aabb {true}
- Vec3d bound_min

  bounds of the wall, disp is used as a skin for broad-phase contact detect
- Vec3d bound_max
- Vec3d bound_disp {0, 0, 0}
- Vec3d pos {0, 0, 0}

  pos and quaternion for translation and rotation
- Vec4d quaternion {1, 0, 0, 0}
- Vec3d force {0, 0, 0}

  forces and moments in the global coordinate system
- Vec3d moment {0, 0, 0}
- Vec3d vel {0, 0, 0}

  translational rotational velocities
- Vec3d spin {0, 0, 0}
- Vec3d vel_spin {0, 0, 0}
- MiniMap< std::string, double > dynamic_properties

  customized properties
- bool need_update_linked_list {true}

  linked-list algorithm, please refer to the description in particle.hpp
- VecXT< std::pair< Cell *, int > > linked_cell_list
- VecXT< NeighPofW > linked_particle_list
- VecXT< NeighPofW > contact_pw_ref_table
- bool need_update_stl_model {false}
- STLModel stl_model

7.122.1 Constructor & Destructor Documentation

7.122.1.1 Wall() [1/2]

```
Wall::Wall ( )
```

7.122.1.2 Wall() [2/2]

```
Wall::Wall (
            Shape *const shape )
```

7.122.1.3 ~Wall()

```
Wall::~Wall ( )
```

7.122.2 Member Function Documentation

7.122.2.1 AddForce()

```
void Wall::AddForce (
            const Vec3d & f )
```

7.122.2.2 AddForceAtomic()

```
void Wall::AddForceAtomic (
            const Vec3d & f )
```

7.122.2.3 AddMoment()

```
void Wall::AddMoment (
            const Vec3d & m )
```

7.122.2.4 AddMomentAtomic()

```
void Wall::AddMomentAtomic (
            const Vec3d & m )
```

7.122.2.5 ApplyContactForce()

```
void Wall::ApplyContactForce (
            ContactPW const * cnt )
```

7.122.2.6 BuildContactRef()

```
void Wall::BuildContactRef ( )
```

7.122.2.7 ClearContactRef()

```
void Wall::ClearContactRef ( )
```

7.122.2.8 ClearForce()

```
void Wall::ClearForce ( )
```

7.122.2.9 ClearLinkedCells()

```
void Wall::ClearLinkedCells ( )
```

7.122.2.10 ClearLinkedNeighs()

```
void Wall::ClearLinkedNeighs ( )
```

7.122.2.11 ClearMoment()

```
void Wall::ClearMoment ( )
```

7.122.2.12 Clone()

```
Wall * Wall::Clone ( ) const
```

7.122.2.13 FindContactRef()

```
int Wall::FindContactRef (
            Particle *const p )
```

7.122.2.14 FindLinked()

```
int Wall::FindLinked (
            Particle *const p )
```

7.122.2.15 GetContactPWs()

```
VecXT< ContactPW * > Wall::GetContactPWs ( )
```

7.122.2.16 GetVelocity()

```
Vec3d Wall::GetVelocity (
            Vec3d const & cnt_pos )
```

7.122.2.17 Init()

```
void Wall::Init ( )
```

7.122.2.18 Print()

```
void Wall::Print ( )
```

7.122.2.19 SaveAsVTK()

```
void Wall::SaveAsVTK (
            std::string const & filename )
```

7.122.2.20 SetPosition()

```
void Wall::SetPosition (
            double pos_x,
            double pos_y,
            double pos_z )
```

7.122.2.21 SetQuaternion()

```
void Wall::SetQuaternion (
            double q_0,
            double q_1,
            double q_2,
            double q_3 )
```

7.122.2.22 SetRodrigues()

```
void Wall::SetRodrigues (
            double angle,
            double axis_x,
            double axis_y,
            double axis_z )
```

7.122.2.23 SetShape()

```
void Wall::SetShape (
            Shape *const shape,
            bool auto_adapt = false )
```

7.122.2.24 SetSpin()

```
void Wall::SetSpin (
            double spin_x,
            double spin_y,
            double spin_z )
```

7.122.2.25 SetVelocity()

```
void Wall::SetVelocity (
            double v_x,
            double v_y,
            double v_z )
```

7.122.2.26 SetVelocitySpin()

```
void Wall::SetVelocitySpin (
            double spin_x,
            double spin_y,
            double spin_z )
```

7.122.2.27 UpdateBound()

```
void Wall::UpdateBound ( )
```

7.122.2.28 UpdateContactForce()

```
void Wall::UpdateContactForce ( )
```

7.122.2.29 UpdateLinkedCells()

```
void Wall::UpdateLinkedCells (
            DomainManager *const dm )
```

7.122.2.30 UpdateLinkedNeighs()

```
void Wall::UpdateLinkedNeighs (
            DomainManager *const dm )
```

7.122.2.31 UpdateMotion() [1/3]

```
void Wall::UpdateMotion (
            const Vec3d & dpos,
            const Vec4d & dquat )
```

7.122.2.32 UpdateMotion() [2/3]

```
void Wall::UpdateMotion (
            const Vec3d & v,
            const Vec3d & s,
            double timestep )
```

7.122.2.33 UpdateMotion() [3/3]

```
void Wall::UpdateMotion (
            double timestep )
```

7.122.2.34 UpdateSTLModel()

```
void Wall::UpdateSTLModel ( )
```

7.122.3 Member Data Documentation

7.122.3.1 bound_disp

```
Vec3d netdem::Wall::bound_disp {0, 0, 0}
```

7.122.3.2 bound_max

```
Vec3d netdem::Wall::bound_max
```

7.122.3.3 bound_min

```
Vec3d netdem::Wall::bound_min
```

bounds of the wall, disp is used as a skin for broad-phase contact detect

7.122.3.4 contact_pw_ref_table

```
VecXT<NeighPofW> netdem::Wall::contact_pw_ref_table
```

7.122.3.5 dynamic_properties

```
MiniMap<std::string, double> netdem::Wall::dynamic_properties
```

customized properties

7.122.3.6 enable_bound_aabb

```
bool netdem::Wall::enable_bound_aabb {true}
```

7.122.3.7 enable_rotation

```
bool netdem::Wall::enable_rotation {false}
```

If not enabled rotation, we can use aabb with skin for linked-linked algorithm. Sometimes aabb can be expensive, enable_bound_aabb determines whether we would like to use aabb in the narrow-phase contact detection.

7.122.3.8 force

```
Vec3d netdem::Wall::force {0, 0, 0}
```

forces and moments in the global coordinate system

7.122.3.9 id

```
int netdem::Wall::id {0}
```

7.122.3.10 label

```
std::string netdem::Wall::label {"default"}
```

7.122.3.11 linked_cell_list

```
VecXT<std::pair<Cell *, int> > netdem::Wall::linked_cell_list
```

7.122.3.12 linked_particle_list

```
VecXT<NeighPofW> netdem::Wall::linked_particle_list
```

7.122.3.13 material_type

```
int netdem::Wall::material_type {0}
```

7.122.3.14 moment

```
Vec3d netdem::Wall::moment {0, 0, 0}
```

7.122.3.15 need_update_linked_list

```
bool netdem::Wall::need_update_linked_list {true}
```

linked-list algorithm, please refer to the description in particle.hpp

7.122.3.16 need_update_stl_model

```
bool netdem::Wall::need_update_stl_model {false}
```

this is only for trimesh intersection-based contact detection and resolution, disable if not the case for efficiency

7.122.3.17 pos

```
Vec3d netdem::Wall::pos {0, 0, 0}
```

pos and quaternion for translation and rotation

7.122.3.18 quaternion

```
Vec4d netdem::Wall::quaternion {1, 0, 0, 0}
```

7.122.3.19 shape

```
Shape* netdem::Wall::shape
```

7.122.3.20 spin

```
Vec3d netdem::Wall::spin {0, 0, 0}
```

7.122.3.21 stl_model

```
STLModel netdem::Wall::stl_model
```

7.122.3.22 vel

```
Vec3d netdem::Wall::vel {0, 0, 0}
```

translational rotational velocities

7.122.3.23 vel_spin

```
Vec3d netdem::Wall::vel_spin {0, 0, 0}
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/wall.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/wall.cpp

7.123 netdem::WallBoxPlane Class Reference

```
#include <gen_wall_box_plane.hpp>
```

Public Member Functions

- WallBoxPlane (double len_x, double len_y, double len_z, double center_x, double center_y, double center_z)
- VecXT< Shape * > GetShapes ()
- VecXT< Wall * > GetWalls ()
- void ImportToScene (Scene *scene)

Public Attributes

- Plane p_mx
- Plane p_px
- Plane p_my
- Plane p_py
- Plane p_mz
- Plane p_pz
- Wall w_mx
- Wall w_px
- Wall w_my
- Wall w_py
- Wall w_mz
- Wall w_pz

7.123.1 Detailed Description

generate box walls.

- p_mx, p_px: the planes with normal (-1, 0, 0) and (1, 0, 0), respectively.
- w_mx, w_px: the walls with normal (-1, 0, 0) and (1, 0, 0), respectively.

7.123.2 Constructor & Destructor Documentation

7.123.2.1 WallBoxPlane()

```
netdem::WallBoxPlane::WallBoxPlane (
            double len_x,
            double len_y,
            double len_z,
            double center_x,
            double center_y,
            double center_z )  [inline]
```

7.123.3 Member Function Documentation

7.123.3.1 GetShapes()

```
VecXT< Shape * > netdem::WallBoxPlane::GetShapes ( )  [inline]
```

7.123.3.2 GetWalls()

```
VecXT< Wall * > netdem::WallBoxPlane::GetWalls ( )  [inline]
```

7.123.3.3 ImportToScene()

```
void netdem::WallBoxPlane::ImportToScene (
            Scene * scene )  [inline]
```

7.123.4 Member Data Documentation

7.123.4.1 p_mx

```
Plane netdem::WallBoxPlane::p_mx
```

7.123.4.2 p_my

```
Plane netdem::WallBoxPlane::p_my
```

7.123.4.3 p_mz

```
Plane netdem::WallBoxPlane::p_mz
```

7.123.4.4 p_px

```
Plane netdem::WallBoxPlane::p_px
```

7.123.4.5 p_py

```
Plane netdem::WallBoxPlane::p_py
```

7.123.4.6 p_pz

```
Plane netdem::WallBoxPlane::p_pz
```

7.123.4.7 w_mx

```
Wall netdem::WallBoxPlane::w_mx
```

7.123.4.8 w_my

```
Wall netdem::WallBoxPlane::w_my
```

7.123.4.9 w_mz

```
Wall netdem::WallBoxPlane::w_mz
```

7.123.4.10 w_px

```
Wall netdem::WallBoxPlane::w_px
```

7.123.4.11 w_py

```
Wall netdem::WallBoxPlane::w_py
```

7.123.4.12 w_pz

```
Wall netdem::WallBoxPlane::w_pz
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/gen_wall_box_plane.hpp

7.124 netdem::WallBoxPlate Class Reference

`#include <gen_wall_box_plate.hpp>`

Public Member Functions

- WallBoxPlate (double len_x, double len_y, double len_z, double center_x, double center_y, double center_z)
- VecXT< Shape * > GetShapes ()
- VecXT< Wall * > GetWalls ()
- void ImportToScene (Scene *scene)

Public Attributes

- TriMesh p_x0
- TriMesh p_y0
- TriMesh p_z0
- Wall w_mx
- Wall w_px
- Wall w_my
- Wall w_py
- Wall w_mz
- Wall w_pz

Private Member Functions

- TriMesh GetBox (double len_x, double len_y, double len_z)

7.124.1 Constructor & Destructor Documentation

7.124.1.1 WallBoxPlate()

```
netdem::WallBoxPlate::WallBoxPlate (
            double len_x,
            double len_y,
            double len_z,
            double center_x,
            double center_y,
            double center_z ) [inline]
```

7.124.2 Member Function Documentation

7.124.2.1 GetBox()

```
TriMesh netdem::WallBoxPlate::GetBox (
            double len_x,
            double len_y,
            double len_z )  [inline], [private]
```

7.124.2.2 GetShapes()

```
VecXT< Shape * > netdem::WallBoxPlate::GetShapes ( )  [inline]
```

7.124.2.3 GetWalls()

```
VecXT< Wall * > netdem::WallBoxPlate::GetWalls ( )  [inline]
```

7.124.2.4 ImportToScene()

```
void netdem::WallBoxPlate::ImportToScene (
            Scene * scene )  [inline]
```

7.124.3 Member Data Documentation

7.124.3.1 p_x0

```
TriMesh netdem::WallBoxPlate::p_x0
```

7.124.3.2 p_y0

```
TriMesh netdem::WallBoxPlate::p_y0
```

7.124.3.3 p_z0

```
TriMesh netdem::WallBoxPlate::p_z0
```

7.124.3.4 w_mx

```
Wall netdem::WallBoxPlate::w_mx
```

7.124.3.5 w_my

```
Wall netdem::WallBoxPlate::w_my
```

7.124.3.6 w_mz

```
Wall netdem::WallBoxPlate::w_mz
```

7.124.3.7 w_px

```
Wall netdem::WallBoxPlate::w_px
```

7.124.3.8 w_py

```
Wall netdem::WallBoxPlate::w_py
```

7.124.3.9 w_pz

```
Wall netdem::WallBoxPlate::w_pz
```

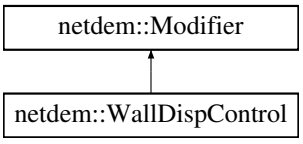
The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/gen_wall_box_plate.hpp

7.125 netdem::WallDispControl Class Reference

```
#include <wall_disp_control.hpp>
```

Inheritance diagram for netdem::WallDispControl:



Public Member Functions

- WallDispControl ()
- void SetVelocity (double v_x, double v_y, double v_z)
- void SetSpin (double spin_x, double spin_y, double spin_z)
- void SetWalls (const VecXT< int > &id_list)
- void SetWalls (const std::initializer_list< int > &id_list)
- Modifier * Clone () const override
- void Execute () override
- void Update () override

Public Attributes

- VecXT< int > wall_id_list
- VecXT< Wall * > wall_list
- Vec3d vel {0, 0, 0}
- Vec3d spin {0, 0, 0}

7.125.1 Constructor & Destructor Documentation

7.125.1.1 WallDispControl()

```
netdem::WallDispControl::WallDispControl ( )
```

7.125.2 Member Function Documentation

7.125.2.1 Clone()

```
Modifier * netdem::WallDispControl::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.125.2.2 Execute()

```
void netdem::WallDispControl::Execute ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.125.2.3 SetSpin()

```
void netdem::WallDispControl::SetSpin (
            double spin_x,
            double spin_y,
            double spin_z )
```

7.125.2.4 SetVelocity()

```
void netdem::WallDispControl::SetVelocity (
            double v_x,
            double v_y,
            double v_z )
```

7.125.2.5 SetWalls() [1/2]

```
void netdem::WallDispControl::SetWalls (
            const std::initializer_list< int > & id_list )
```

7.125.2.6 SetWalls() [2/2]

```
void netdem::WallDispControl::SetWalls (
            const VecXT< int > & id_list )
```

7.125.2.7 Update()

```
void netdem::WallDispControl::Update ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.125.3 Member Data Documentation

7.125.3.1 spin

```
Vec3d netdem::WallDispControl::spin {0, 0, 0}
```

7.125.3.2 vel

```
Vec3d netdem::WallDispControl::vel {0, 0, 0}
```

7.125.3.3 wall_id_list

```
VecXT<int> netdem::WallDispControl::wall_id_list
```

7.125.3.4 wall_list

```
VecXT<Wall *> netdem::WallDispControl::wall_list
```

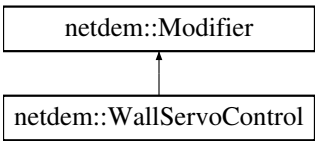
The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/wall_disp_control.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/wall_disp_control.cpp

7.126 netdem::WallServoControl Class Reference

```
#include <wall_servo_control.hpp>
```

Inheritance diagram for netdem::WallServoControl:



Public Member Functions

- WallServoControl (double kn)
- void SetWalls (const VecXT< int > &id_list)
- void SetWalls (const std::initializer_list< int > &id_list)
- void AddWall (int id)
- Modifier * Clone () const override
- void Execute () override
- void Update () override

Public Attributes

- VecXT< int > wall_id_list
- VecXT< Wall * > wall_list
- VecXT< double > pressure_list
- double kn {1.0e5}
- double target_pressure {0.0}
- double vel_max {0.5}
- double study_rate {0.5}
- double tol {0.05}
- bool achieved {true}
- bool enable_warning {false}

7.126.1 Constructor & Destructor Documentation

7.126.1.1 WallServoControl()

```
netdem::WallServoControl::WallServoControl (
            double kn )
```

7.126.2 Member Function Documentation

7.126.2.1 AddWall()

```
void netdem::WallServoControl::AddWall (
            int id )
```

7.126.2.2 Clone()

```
Modifier * netdem::WallServoControl::Clone ( ) const  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.126.2.3 Execute()

```
void netdem::WallServoControl::Execute ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.126.2.4 SetWalls() [1/2]

```
void netdem::WallServoControl::SetWalls (
            const std::initializer_list< int > & id_list )
```

7.126.2.5 SetWalls() [2/2]

```
void netdem::WallServoControl::SetWalls (
            const VecXT< int > & id_list )
```

7.126.2.6 Update()

```
void netdem::WallServoControl::Update ( )  [override], [virtual]
```

Reimplemented from netdem::Modifier.

7.126.3 Member Data Documentation

7.126.3.1 achieved

```
bool netdem::WallServoControl::achieved {true}
```

7.126.3.2 enable_warning

```
bool netdem::WallServoControl::enable_warning {false}
```

7.126.3.3 kn

```
double netdem::WallServoControl::kn {1.0e5}
```

7.126.3.4 pressure_list

```
VecXT<double> netdem::WallServoControl::pressure_list
```

7.126.3.5 study_rate

```
double netdem::WallServoControl::study_rate {0.5}
```

7.126.3.6 target_pressure

```
double netdem::WallServoControl::target_pressure {0.0}
```

7.126.3.7 tol

```
double netdem::WallServoControl::tol {0.05}
```

7.126.3.8 vel_max

```
double netdem::WallServoControl::vel_max {0.5}
```

7.126.3.9 wall_id_list

```
VecXT<int> netdem::WallServoControl::wall_id_list
```

7.126.3.10 wall_list

```
VecXT<Wall *> netdem::WallServoControl::wall_list
```

The documentation for this class was generated from the following files:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/wall_servo_control.hpp
- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/wall_servo_control.cpp

7.127 netdem::WSCVTSampler Class Reference

```
#include <wscvt_sampler.hpp>
```

Public Member Functions

- WSCVTSampler (const WSCVTSampler &)=delete
- WSCVTSampler & operator= (const WSCVTSampler &)=delete
- VecXT< Vec3d > Get (int num_samples, bool new_random=false)

Static Public Member Functions

- static WSCVTSampler & GetInstance ()

Public Attributes

- int max_iters {10000}
- double tol {1.0e-4}

Private Member Functions

- WSCVTSampler ()
- VecXT< Vec3d > Create (int num_samples)

Private Attributes

- MiniMap< int, VecXT< Vec3d > > samples_map

7.127.1 Constructor & Destructor Documentation

7.127.1.1 WSCVTSampler() [1/2]

```
netdem::WSCVTSampler::WSCVTSampler (
            const WSCVTSampler &  )  [delete]
```

7.127.1.2 WSCVTSampler() [2/2]

```
netdem::WSCVTSampler::WSCVTSampler ( )  [inline], [private]
```

7.127.2 Member Function Documentation

7.127.2.1 Create()

```
VecXT< Vec3d > netdem::WSCVTSampler::Create (
            int num_samples )  [inline], [private]
```

7.127.2.2 Get()

```
VecXT< Vec3d > netdem::WSCVTSampler::Get (
            int num_samples,
            bool new_random = false )  [inline]
```

7.127.2.3 GetInstance()

```
static WSCVTSampler & netdem::WSCVTSampler::GetInstance ( )  [inline], [static]
```

7.127.2.4 operator=()

```
WSCVTSampler & netdem::WSCVTSampler::operator= (
            const WSCVTSampler &  )  [delete]
```

7.127.3 Member Data Documentation

7.127.3.1 max_iters

```
int netdem::WSCVTSampler::max_iters {10000}
```

7.127.3.2 samples_map

```
MiniMap<int, VecXT<Vec3d> > netdem::WSCVTSampler::samples_map  [private]
```

7.127.3.3 tol

```
double netdem::WSCVTSampler::tol {1.0e-4}
```

The documentation for this class was generated from the following file:

- /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/wscvt_sampler.hpp

Chapter 8

File Documentation

8.1 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/readme.md File Reference

8.2 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_entry.cpp File Reference

```
#include "bond_entry.hpp"
#include "contact_pp.hpp"
#include "contact_pw.hpp"
```

8.3 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_entry.hpp File Reference

```
#include "bond_geometries.hpp"
#include "contact_forces.hpp"
#include "contact_model.hpp"
```

Classes

- class netdem::BondEntry

Namespaces

- namespace netdem

8.4 bond_entry.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_geometries.hpp"
4 #include "contact_forces.hpp"
5 #include "contact_model.hpp"
6
7 namespace netdem {
8
9 class ContactPP;
10 class ContactPW;
11
12 class BondEntry {
13 public:
14   BondGeometries cnt_geoms;
15   ContactForces cnt_forces;
16
17   // we maintein a contact model pointer in each individual contact entry for
18   // potential extersion
19   ContactModel *cnt_model{nullptr};
20
21   void UpdateForces(ContactPP *const cnt, double dt);
22   void UpdateForces(ContactPW *const cnt, double dt);
23
24   void UpdateLocalForces(ContactPP *const cnt, double dt);
25   void UpdateLocalForces(ContactPW *const cnt, double dt);
26
27   void UpdateGlobalForces();
28 };
29
30 } // namespace netdem
```

8.5 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_geometries.hpp File Reference

```
#include "utils_math.hpp"
```

Classes

- class netdem::BondGeometries

Namespaces

- namespace netdem

8.6 bond_geometries.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_math.hpp"
4
5 namespace netdem {
6
7 class BondGeometries {
8 public:
9   Vec3d pos{0, 0, 0};
10   Vec3d dir_n{1, 0, 0}, dir_s{0, 1, 0}, dir_t{0, 0, 1};
```

```
11   Vec3d branch_1{1, 0, 0}, branch_2{1, 0, 0};
12
13   Vec3d pos_ini{0, 0, 0};
14   Vec3d dir_n_ini{1, 0, 0}, dir_s_ini{0, 1, 0}, dir_t_ini{0, 0, 1};
15
16   Vec3d pos_1_ini{0, 0, 0}, pos_2_ini{0, 0, 0};
17   Vec4d quat_1_ini{1, 0, 0, 0}, quat_2_ini{1, 0, 0, 0};
18
19   double radius{0};
20
21   double len_n{0}, len_s{0}, len_t{0};
22   double theta_n{0}, theta_s{0}, theta_t{0};
23
24   bool active{false};
25 };
26
27 } // namespace netdem
```

8.7 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_solver_pp.cpp File Reference

```
#include "bond_solver_pp.hpp"
#include "utils_math.hpp"
#include <iostream>
#include <string>
```

8.8 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/bond_solver_pp.hpp File Reference

```
#include "contact_pp.hpp"
#include "particle.hpp"
```

Classes

- class netdem::BondSolverPP

Namespaces

- namespace netdem

8.9 bond_solver_pp.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "contact_pp.hpp"
4 #include "particle.hpp"
5
6 namespace netdem {
7
8 class BondSolverPP {
9 public:
10   Particle *particle_1{nullptr}, *particle_2{nullptr};
11
12   BondSolverPP();
13   BondSolverPP(Particle *const p1, Particle *const p2);
14
15   void Init(Particle *const p1, Particle *const p2);
16
17   void ResolveInit(ContactPP *const cnt, double timestep);
18   void ResolveUpdate(ContactPP *const cnt, double timestep);
19
20   void ResolveInit(BondGeometries *const cnt_geoms, Vec3d const &bond_pos,
21                    Vec3d const &bond_dir_n, double bound_radius);
22   void ResolveUpdate(BondGeometries *const cnt_geoms, double timestep);
23 };
24
25 } // namespace netdem
```

8.10 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/bond_solver_pw.cpp File Reference

```
#include "bond_solver_pw.hpp"
#include "utils_math.hpp"
#include <iostream>
#include <string>
```

8.11 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/bond_solver_pw.hpp File Reference

```
#include "contact_pw.hpp"
#include "particle.hpp"
#include "wall.hpp"
```

Classes

- class netdem::BondSolverPW

Namespaces

- namespace netdem

8.12 bond_solver_pw.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "contact_pw.hpp"
4 #include "particle.hpp"
5 #include "wall.hpp"
6
7 namespace netdem {
8
9 class BondSolverPW {
10 public:
11   Particle *particle{nullptr};
12   Wall *wall{nullptr};
13
14   BondSolverPW();
15   BondSolverPW(Particle *const p, Wall *const w);
16
17   void Init(Particle *const p, Wall *const w);
18
19   void ResolveInit(ContactPW *const cnt, double timestep);
20   void ResolveUpdate(ContactPW *const cnt, double timestep);
21
22   void ResolveInit(BondGeometries *const cnt_geoms, Vec3d const &bond_pos,
23                    Vec3d const &bond_dir_n, double bound_radius);
24   void ResolveUpdate(BondGeometries *const cnt_geoms, double timestep);
25 };
26
27 } // namespace netdem
```

8.13 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/collision_entry.cpp File Reference

```
#include "collision_entry.hpp"
#include "contact_pp.hpp"
#include "contact_pw.hpp"
```

8.14 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/collision_entry.hpp File Reference

```
#include "collision_geometries.hpp"
#include "contact_forces.hpp"
#include "contact_model.hpp"
```

Classes

- class netdem::CollisionEntry

Namespaces

- namespace netdem

8.15 collision_entry.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "contact_forces.hpp"
5 #include "contact_model.hpp"
6
7 namespace netdem {
8
9 class ContactPP;
10 class ContactPW;
11
12 class CollisionEntry {
13 public:
14   CollisionGeometries cnt_geoms;
15   ContactForces cnt_forces;
16
17   // we maintein a contact model pointer in each individual contact entry for
18   // potential extersion
19   ContactModel *cnt_model{nullptr};
20
21   void UpdateForces(ContactPP *const cnt, double dt);
22   void UpdateForces(ContactPW *const cnt, double dt);
23
24   void UpdateLocalForces(ContactPP *const cnt, double dt);
25   void UpdateLocalForces(ContactPW *const cnt, double dt);
26
27   void UpdateGlobalForces();
28 };
29
30 } // namespace netdem
```

8.16 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/collision_geometries.hpp File Reference

```
#include "utils_math.hpp"
```

Classes

- class netdem::CollisionGeometries

Namespaces

- namespace netdem

8.17 collision_geometries.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_math.hpp"
4
5 namespace netdem {
6
31 class CollisionGeometries {
32 public:
33   // for general contact
34   Vec3d pos{0, 0, 0};
```

```
35   Vec3d dir_n{1, 0, 0}, dir_s{0, 1, 0}, dir_t{0, 0, 1};
36   Vec3d branch_1{1, 0, 0}, branch_2{1, 0, 0};
37
38   double len_n{0}, dlen_n{0}, dlen_s{0}, dlen_t{0};
39   double dtheta_n{0}, dtheta_s{0}, dtheta_t{0};
40   double radius_1{1}, radius_2{1};
41
42   // for general purpose (track if the contact entry is still active)
43   bool active{true};
44
45   // for node-based contact solver
46   int node_id{0};
47   double node_dist{0};
48
49   // for contact volume based contact model
50   double vol{0}, sn{0};
51 };
52
53 } // namespace netdem
```

8.18 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/collision_solver_pp.hpp File Reference

```
#include "collision_geometries.hpp"
#include "contact_model.hpp"
#include "contact_pp.hpp"
#include "particle.hpp"
```

Classes

- class netdem::CollisionSolverPP

Namespaces

- namespace netdem

8.19 collision_solver_pp.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "contact_model.hpp"
5 #include "contact_pp.hpp"
6 #include "particle.hpp"
7
8 namespace netdem {
9
15 class CollisionSolverPP {
16 public:
17   Particle *particle_1{nullptr}, *particle_2{nullptr};
18
19   CollisionSolverPP() {}
20   CollisionSolverPP(Particle *const p1, Particle *const p2)
21       : particle_1(p1), particle_2(p2) {}
22
23   virtual CollisionSolverPP *Clone() const = 0;
24
25   virtual void Init(Particle *const p1, Particle *const p2) {
26     particle_1 = p1;
27     particle_2 = p2;
```

```
28   }
29
30   virtual bool Detect() = 0;
31   virtual bool Detect(ContactPP *const cnt) { return Detect(); }
32
33   virtual void ResolveInit(ContactPP *const cnt, double timestep) = 0;
34   virtual void ResolveUpdate(ContactPP *const cnt, double timestep) = 0;
35
36   virtual ~CollisionSolverPP() {}
37
38 protected:
39   // Provided: len_n, dir_n, pos. Compute: branch vectors, dir_s, dir_t, dlen_n,
40   // dlen_s, dlen_t, etc.
41   void InitBasicGeoms(CollisionGeometries *const cnt_geoms, double timestep) {
42     cnt_geoms->branch_1 = cnt_geoms->pos - particle_1->pos;
43     cnt_geoms->branch_2 = cnt_geoms->pos - particle_2->pos;
44
45     Vec3d x_axis = {1.0, 0.0, 0.0};
46     auto x_cross_dir_n = Math::Cross(x_axis, cnt_geoms->dir_n);
47     if (Math::NormLen(x_cross_dir_n) > 1.0e-7) {
48       double x_dot_dir_n = Math::Dot(x_axis, cnt_geoms->dir_n);
49       cnt_geoms->dir_s = x_axis - x_dot_dir_n * cnt_geoms->dir_n;
50       Math::Normalize(&cnt_geoms->dir_s);
51     } else {
52       Vec3d y_axis = {0.0, 1.0, 0.0};
53       double y_dot_dir_n = Math::Dot(y_axis, cnt_geoms->dir_n);
54       cnt_geoms->dir_s = y_axis - y_dot_dir_n * cnt_geoms->dir_n;
55       Math::Normalize(&cnt_geoms->dir_s);
56     }
57     cnt_geoms->dir_t = Math::Cross(cnt_geoms->dir_n, cnt_geoms->dir_s);
58
59     auto branch_cross_omega_1 =
60         Math::Cross(particle_1->spin, cnt_geoms->branch_1);
61     auto branch_cross_omega_2 =
62         Math::Cross(particle_2->spin, cnt_geoms->branch_2);
63     auto vel_relative = particle_1->vel - particle_2->vel +
64                         branch_cross_omega_1 - branch_cross_omega_2;
65
66     cnt_geoms->dlen_n = Math::Dot(vel_relative, cnt_geoms->dir_n) * timestep;
67     cnt_geoms->dlen_s = Math::Dot(vel_relative, cnt_geoms->dir_s) * timestep;
68     cnt_geoms->dlen_t = Math::Dot(vel_relative, cnt_geoms->dir_t) * timestep;
69   }
70
71   // Provided: len_n, dir_n, pos, dir_n_old. Compute: branch vectors, dir_s,
72   // dir_t, dlen_n, dlen_s, dlen_t, etc.
73   void UpdateBasicGeoms(CollisionGeometries *const cnt_geoms, double timestep,
74                         Vec3d const &dir_n_old) {
75     cnt_geoms->branch_1 = cnt_geoms->pos - particle_1->pos;
76     cnt_geoms->branch_2 = cnt_geoms->pos - particle_2->pos;
77
78     auto n_old_cross_n_new = Math::Cross(dir_n_old, cnt_geoms->dir_n);
79     auto dir_s_cross_nxn = Math::Cross(cnt_geoms->dir_s, n_old_cross_n_new);
80     auto dir_s_1 = cnt_geoms->dir_s - dir_s_cross_nxn;
81     auto omega_dot_dir_n = Math::Dot(particle_1->spin, cnt_geoms->dir_n);
82     auto dtheta = omega_dot_dir_n * cnt_geoms->dir_n * timestep;
83     auto dir_s_1_cross_dtheta = Math::Cross(dir_s_1, dtheta);
84     cnt_geoms->dir_s = dir_s_1 - dir_s_1_cross_dtheta;
85     Math::Normalize(&cnt_geoms->dir_s);
86     cnt_geoms->dir_t = Math::Cross(cnt_geoms->dir_n, cnt_geoms->dir_s);
87
88     auto vel_relative = particle_1->GetVelocity(cnt_geoms->pos) -
89                         particle_2->GetVelocity(cnt_geoms->pos);
90
91     cnt_geoms->dlen_n = Math::Dot(vel_relative, cnt_geoms->dir_n) * timestep;
92     cnt_geoms->dlen_s = Math::Dot(vel_relative, cnt_geoms->dir_s) * timestep;
93     cnt_geoms->dlen_t = Math::Dot(vel_relative, cnt_geoms->dir_t) * timestep;
94   }
95 };
96
97 } // namespace netdem
```

8.20 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/collision_solver_pw.hpp File Reference

```
#include "collision_geometries.hpp"
#include "contact_model.hpp"
#include "contact_pw.hpp"
```

```
#include "particle.hpp"
#include "wall.hpp"
```

Classes

- class netdem::CollisionSolverPW

Namespaces

- namespace netdem

8.21 collision_solver_pw.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "contact_model.hpp"
5 #include "contact_pw.hpp"
6 #include "particle.hpp"
7 #include "wall.hpp"
8
9 namespace netdem {
10
16 class CollisionSolverPW {
17 public:
18   Particle *particle{nullptr};
19   Wall *wall{nullptr};
20
21   CollisionSolverPW() {}
22   CollisionSolverPW(Particle *const p, Wall *const w) : particle(p), wall(w) {}
23
24   virtual CollisionSolverPW *Clone() const = 0;
25
26   virtual void Init(Particle *const p, Wall *const w) {
27     particle = p;
28     wall = w;
29   }
30
31   virtual bool Detect() = 0;
32   virtual bool Detect(ContactPW *const cnt) { return Detect(); }
33
34   virtual void ResolveInit(ContactPW *const cnt, double timestep) = 0;
35   virtual void ResolveUpdate(ContactPW *const cnt, double timestep) = 0;
36
37   virtual ~CollisionSolverPW() {}
38
39 protected:
40   // Provided: len_n, dir_n, pos. Compute: branch vectors, dir_s, dir_t, dlen_n,
41   // dlen_s, dlen_t, etc.
42   void InitBasicGeoms(CollisionGeometries *const cnt_geoms, double timestep) {
43     cnt_geoms->branch_1 = cnt_geoms->pos - particle->pos;
44     cnt_geoms->branch_2 = cnt_geoms->pos - wall->pos;
45
46     Vec3d x_axis = {1.0, 0.0, 0.0};
47     auto x_cross_dir_n = Math::Cross(x_axis, cnt_geoms->dir_n);
48     if (Math::NormLen(x_cross_dir_n) > 1.0e-7) {
49       double x_dot_dir_n = Math::Dot(x_axis, cnt_geoms->dir_n);
50       cnt_geoms->dir_s = x_axis - x_dot_dir_n * cnt_geoms->dir_n;
51       Math::Normalize(&cnt_geoms->dir_s);
52     } else {
53       Vec3d y_axis = {0.0, 1.0, 0.0};
54       double y_dot_dir_n = Math::Dot(y_axis, cnt_geoms->dir_n);
55       cnt_geoms->dir_s = y_axis - y_dot_dir_n * cnt_geoms->dir_n;
56       Math::Normalize(&cnt_geoms->dir_s);
57     }
58     cnt_geoms->dir_t = Math::Cross(cnt_geoms->dir_n, cnt_geoms->dir_s);
59
60     auto branch_cross_omega_1 =
61         Math::Cross(particle->spin, cnt_geoms->branch_1);
62     auto branch_cross_omega_2 = Math::Cross(wall->spin, cnt_geoms->branch_2);
```

```
63      auto vel_relative =
64          particle->vel - wall->vel + branch_cross_omega_1 - branch_cross_omega_2;
65
66      cnt_geoms->dlen_n = Math::Dot(vel_relative, cnt_geoms->dir_n) * timestep;
67      cnt_geoms->dlen_s = Math::Dot(vel_relative, cnt_geoms->dir_s) * timestep;
68      cnt_geoms->dlen_t = Math::Dot(vel_relative, cnt_geoms->dir_t) * timestep;
69    }
70
71    // Provided: len_n, dir_n, pos, dir_n_old. Compute: branch vectors, dir_s,
72    // dir_t, dlen_n, dlen_s, dlen_t, etc.
73    void UpdateBasicGeoms(CollisionGeometries *const cnt_geoms, double timestep,
74                          Vec3d const &dir_n_old) {
75      cnt_geoms->branch_1 = cnt_geoms->pos - particle->pos;
76      cnt_geoms->branch_2 = cnt_geoms->pos - wall->pos;
77
78      auto n_old_cross_n_new = Math::Cross(dir_n_old, cnt_geoms->dir_n);
79      auto dir_s_cross_nxn = Math::Cross(cnt_geoms->dir_s, n_old_cross_n_new);
80      auto dir_s_1 = cnt_geoms->dir_s - dir_s_cross_nxn;
81      auto omega_dot_dir_n = Math::Dot(particle->spin, cnt_geoms->dir_n);
82      auto dtheta = omega_dot_dir_n * cnt_geoms->dir_n * timestep;
83      auto dir_s_1_cross_dtheta = Math::Cross(dir_s_1, dtheta);
84      cnt_geoms->dir_s = dir_s_1 - dir_s_1_cross_dtheta;
85      Math::Normalize(&cnt_geoms->dir_s);
86      cnt_geoms->dir_t = Math::Cross(cnt_geoms->dir_n, cnt_geoms->dir_s);
87
88      auto vel_relative = particle->GetVelocity(cnt_geoms->pos) -
89                          wall->GetVelocity(cnt_geoms->pos);
90
91      cnt_geoms->dlen_n = Math::Dot(vel_relative, cnt_geoms->dir_n) * timestep;
92      cnt_geoms->dlen_s = Math::Dot(vel_relative, cnt_geoms->dir_s) * timestep;
93      cnt_geoms->dlen_t = Math::Dot(vel_relative, cnt_geoms->dir_t) * timestep;
94    }
95  };
96
97  } // namespace netdem
```

8.22 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/contact_forces.hpp File Reference

Classes

- class netdem::ContactForces

Namespaces

- namespace netdem

8.23 contact_forces.hpp

Go to the documentation of this file.

```
1  #pragma once
2
3  namespace netdem {
4
15 class ContactForces {
16 public:
17   // local
18   double fc_n{0}, fc_s{0}, fc_t{0};
19   double mc_n{0}, mc_s{0}, mc_t{0};
20
21   double fd_n{0}, fd_s{0}, fd_t{0};
22   double md_n{0}, md_s{0}, md_t{0};
23
24   // global
25   Vec3d force{0, 0, 0};
26   Vec3d moment{0, 0, 0};
```

```
27
28   Vec3d force_n{0, 0, 0}, force_t{0, 0, 0};   // normal and tangential
29   Vec3d moment_n{0, 0, 0}, moment_t{0, 0, 0}; // normal and tangential
30
31   void Clear() {
32     fc_n = 0;
33     fc_s = 0;
34     fc_t = 0;
35
36     mc_n = 0;
37     mc_s = 0;
38     mc_t = 0;
39
40     fd_n = 0;
41     fd_s = 0;
42     fd_t = 0;
43
44     md_n = 0;
45     md_s = 0;
46     md_t = 0;
47
48     force[0] = 0;
49     force[1] = 0;
50     force[2] = 0;
51
52     moment[0] = 0;
53     moment[1] = 0;
54     moment[2] = 0;
55
56     force_n[0] = 0;
57     force_n[1] = 0;
58     force_n[2] = 0;
59
60     force_t[0] = 0;
61     force_t[1] = 0;
62     force_t[2] = 0;
63
64     moment_n[0] = 0;
65     moment_n[1] = 0;
66     moment_n[2] = 0;
67
68     moment_t[0] = 0;
69     moment_t[1] = 0;
70     moment_t[2] = 0;
71   }
72 };
73
74 } // namespace netdem
```

8.24 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/contact_model.hpp File Reference

```
#include "bond_geometries.hpp"
#include "collision_geometries.hpp"
#include "contact_forces.hpp"
#include "utils_io.hpp"
#include <iostream>
#include <nlohmann/json.hpp>
#include <string>
```

Classes

- class netdem::ContactModel

Namespaces

- namespace netdem

8.25 contact_model.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_geometries.hpp"
4 #include "collision_geometries.hpp"
5 #include "contact_forces.hpp"
6 #include "utils_io.hpp"
7 #include <iostream>
8 #include <nlohmann/json.hpp>
9 #include <string>
10
11 namespace netdem {
12
13 class ContactPP;
14 class ContactPW;
15
20 class ContactModel {
21 public:
22   enum Type { none, linear_spring, hertz_mindlin, volume_based, parallel_bond };
23
24   int id{0};
25   std::string label{"default"};
26
27   int model_type{0};
28   std::string model_name{"contact_model"};
29
30   virtual nlohmann::json PackJson() {
31     PrintWarning("in ContactModel::PackJson, method not implemented for: " +
32                  model_name);
33     return nlohmann::json();
34   }
35
36   virtual void InitFromJson(nlohmann::json const &js) {
37     PrintWarning("in ContactModel::InitFromJson, method not implemented for: " +
38                  model_name);
39   }
40
41   virtual void SetProperty(nlohmann::json const &js) {
42     PrintWarning("in ContactModel::SetProperty, method not implemented for: " +
43                  model_name);
44   }
45
46   virtual ContactModel *Clone() const { return new ContactModel(*this); }
47
48   virtual void EvaluateForceMoment(ContactForces *const cnt_forces,
49                                    CollisionGeometries &cnt_geoms,
50                                    ContactPP *const cnt, double dt) const {
51     PrintWarning(
52         "in ContactModel::EvaluateForceMoment, method not implemented for: " +
53         model_name);
54   }
55
56   virtual void EvaluateForceMoment(ContactForces *const cnt_forces,
57                                    BondGeometries &cnt_geoms,
58                                    ContactPP *const cnt, double dt) const {
59     PrintWarning(
60         "in ContactModel::EvaluateForceMoment, method not implemented for: " +
61         model_name);
62   }
63
64   virtual void EvaluateForceMoment(ContactForces *const cnt_forces,
65                                    CollisionGeometries &cnt_geoms,
66                                    ContactPW *const cnt, double dt) const {
67     PrintWarning(
68         "in ContactModel::EvaluateForceMoment, method not implemented for: " +
69         model_name);
70   }
71
72   virtual void EvaluateForceMoment(ContactForces *const cnt_forces,
73                                    BondGeometries &cnt_geoms,
74                                    ContactPW *const cnt, double dt) const {
75     PrintWarning(
76         "in ContactModel::EvaluateForceMoment, method not implemented for: " +
77         model_name);
78   }
79
80   virtual void Print() const {
81     PrintWarning("in ContactModel::Print, method not implemented for: " +
82                  model_name);
83   }
84
85   virtual ~ContactModel() {}
86 };
```

```
87
88 } // namespace netdem
```

8.26 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/contact_model_↩ factory.cpp File Reference

```
#include "contact_model_factory.hpp"
#include "model_hertz_mindlin.hpp"
#include "model_linear_spring.hpp"
#include "model_parallel_bond.hpp"
#include "model_volume_based.hpp"
#include <iostream>
#include <string>
```

8.27 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/contact_model_↩ factory.hpp File Reference

```
#include "contact_model.hpp"
#include <string>
#include <unordered_map>
```

Classes

- class netdem::ContactModelFactory

Namespaces

- namespace netdem

8.28 contact_model_factory.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "contact_model.hpp"
4 #include <string>
5 #include <unordered_map>
6
7 namespace netdem {
8
9 class ContactModelFactory {
10 public:
11   static std::unordered_map<std::string, ContactModel::Type> model_map;
12
13   static ContactModel *NewContactModel(std::string const &model_name,
14                                        nlohmann::json const &js);
15 };
16
17 } // namespace netdem
```

8.29 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/contact_solver_↩ factory.cpp File Reference

```
#include "contact_solver_factory.hpp"
#include "bond_solver_pp.hpp"
#include "bond_solver_pw.hpp"
#include "particle.hpp"
#include "solver_ann_pp.hpp"
#include "solver_boolean_pp.hpp"
#include "solver_boolean_pw.hpp"
#include "solver_gjk_pp.hpp"
#include "solver_gjk_pw.hpp"
#include "solver_sdf_pp.hpp"
#include "solver_sdf_pw.hpp"
#include "solver_sphere_plane.hpp"
#include "solver_sphere_sphere.hpp"
#include "solver_sphere_triangle.hpp"
#include "wall.hpp"
#include <iostream>
```

8.30 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/contact_solver_↩ factory.hpp File Reference

```
#include "bond_solver_pp.hpp"
#include "bond_solver_pw.hpp"
#include "collision_solver_pp.hpp"
#include "collision_solver_pw.hpp"
#include "utils_math.hpp"
#include <memory>
#include <unordered_map>
```

Classes

- class netdem::ContactSolverSettings
- class netdem::ContactSolverFactory

Namespaces

- namespace netdem

8.31 contact_solver_factory.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_solver_pp.hpp"
4 #include "bond_solver_pw.hpp"
5 #include "collision_solver_pp.hpp"
6 #include "collision_solver_pw.hpp"
7 #include "utils_math.hpp"
8 #include <memory>
9 #include <unordered_map>
10
11 namespace netdem {
12
13 class ContactSolverSettings {
14 public:
15   enum SolverType { gjk, sdf, automatic };
16
17   SolverType solver_type{SolverType::automatic};
18
19   bool gjk_use_erosion{false};
20   double gjk_erosion_ratio_initial{0.01}, gjk_erosion_ratio_increment{0.01};
21
22   int sdf_potential_type{0};
23   bool sdf_solve_two_sides{false};
24 };
25
36 class ContactSolverFactory {
37 public:
38   ContactSolverSettings settings;
39
40   // bond solvers
41   BondSolverPP bond_solver_pp;
42   BondSolverPW bond_solver_pw;
43
46   VecNT<VecNT<int, Shape::Type::num_shapes>, Shape::Type::num_shapes>
47       solver_id_pp_list, solver_id_pw_list;
48
49   std::unordered_map<std::pair<int, int>, int, pair_hash> solver_id_customized;
50
51   VecXT<CollisionSolverPP *> solver_pp_pool;
52   VecXT<CollisionSolverPW *> solver_pw_pool;
53
54   ContactSolverFactory();
55
56   ContactSolverFactory(ContactSolverFactory &tmp_factory);
57   ContactSolverFactory(ContactSolverFactory &&tmp_factory);
58   ContactSolverFactory &operator=(ContactSolverFactory &tmp_factory);
59   ContactSolverFactory &operator=(ContactSolverFactory &&tmp_factory);
60
61   BondSolverPP *GetBondSolver(Particle *const p1, Particle *const p2);
62   BondSolverPW *GetBondSolver(Particle *const p, Wall *const w);
63
64   CollisionSolverPP *GetCollisionSolver(Particle *const p1, Particle *const p2);
65   CollisionSolverPW *GetCollisionSolver(Particle *const p, Wall *const w);
66
67   int InsertSolver(CollisionSolverPP *const cnt_solver);
68   int InsertSolver(CollisionSolverPW *const cnt_solver);
69
70   void CustomizeSolverPP(int shape_id1, int shape_id2, int solver_id);
71   void CustomizeSolverPW(int shape_id1, int shape_id2, int solver_id);
72
73   ~ContactSolverFactory();
74
75 private:
76   CollisionSolverPP *NewCollisionSolver(Particle *const p1, Particle *const p2);
77   CollisionSolverPW *NewCollisionSolver(Particle *const p, Wall *const w);
78 };
79
80 } // namespace netdem
```

8.32 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/dem_profiler.cpp File Reference

```
#include "dem_profiler.hpp"
#include <iostream>
```

8.33 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/dem_profiler.hpp File Reference

`#include <sys/time.h>`

Classes

- class netdem::DEMProfiler

Namespaces

- namespace netdem

Typedefs

- typedef long long netdem::int64t

Enumerations

- enum netdem::TimerType {
  netdem::linked_list , netdem::contacts , netdem::particles , netdem::walls ,
  netdem::pre_modifiers , netdem::mid_modifiers , netdem::post_modifiers , netdem::mpi_communication ,
  netdem::custom , netdem::num_timers }

8.34 dem_profiler.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifdef _WIN32
4 #include <windows.h>
5 #else
6 #include <sys/time.h>
7 #endif
8
9 namespace netdem {
10
11 #if defined(_WIN32) && !defined(CYGWIN)
12 typedef __int64 int64t;
13 #else
14 typedef long long int64t;
15 #endif // _WIN32
16
17 enum TimerType {
18   linked_list,
19   contacts,
20   particles,
21   walls,
22   pre_modifiers,
23   mid_modifiers,
24   post_modifiers,
```

```
25   mpi_communication,
26   custom,
27   num_timers
28 };
29
30 class DEMProfiler {
31 public:
32   int64t timer_list[TimerType::num_timers];
33
34   int num_particles{0}, num_walls{0}, num_neighs{0}, num_neigh_builds{0};
35   double num_neighs_per_p{0};
36
37   DEMProfiler();
38
39   static int64t GetTimeMicros();
40
41   void StartTimer(TimerType t_type);
42   void EndTimer(TimerType t_type);
43
44   void Clear();
45   void Print();
46
47 private:
48   int64t t_start[TimerType::num_timers];
49   bool timer_started[TimerType::num_timers];
50
51   inline int64t GetTotalTime();
52 };
53
54 } // namespace netdem
```

8.35 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/dem_solver.cpp File Reference

```
#include "dem_solver.hpp"
#include "dem_object_pool.hpp"
#include "modifier_manager.hpp"
#include "mpi_manager.hpp"
#include "scene.hpp"
#include "simulation.hpp"
#include <cmath>
#include <iostream>
#include <omp.h>
```

8.36 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/dem_solver.hpp File Reference

```
#include "contact_solver_factory.hpp"
#include "dem_profiler.hpp"
```

Classes

- class netdem::DEMSolver

Namespaces

- namespace netdem

8.37 dem_solver.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "contact_solver_factory.hpp"
4 #include "dem_profiler.hpp"
5
6 namespace netdem {
7
8 class Simulation;
9 class MPIManager;
10 class ModifierManager;
11 class Scene;
12
13 class DEMSolver {
14 public:
16   enum CyclePoint { pre, mid_0, mid_1, mid_2, post, num_cycle_points };
17
18   double timestep{1.0e-4};
19
21   DEMProfiler dem_profiler;
22
24   ContactSolverFactory contact_solver_factory;
25
26   DEMSolver();
27
28   void Init(Simulation *sim);
29
30   void UpdatePreModifiers();
31   void UpdateLinkedList();
32   void UpdateContacts();
33   void UpdateParticles();
34   void UpdateWalls();
35   void UpdatePostModifiers();
36
37   void Cycle(int num_cycles);
38   void Solve(double time);
39
40 private:
41   Simulation *sim{nullptr};
42
43   void UpdateMidModifiers(CyclePoint cyc_point);
44
45   void Cycle();
46
47   // dry cycle just updates the pre-modifiers and does no particle interaction
48   // and motion updating
49   void DryCycle();
50
51   void SolveContactPP(Particle *const p_ii, NeighPofP *const neigh_tuple,
52                       ContactSolverFactory *const solver_factory);
53   void SolveContactPW(Particle *const p_ii, NeighWofP *const neigh_tuple,
54                       ContactSolverFactory *const solver_factory);
55 };
56
57 } // namespace netdem
```

8.38 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/gjk_simplex.hpp File Reference

```
#include "utils_math.hpp"
```

Classes

- class netdem::Simplex

Namespaces

- namespace netdem

8.39 gjk_simplex.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_math.hpp"
4
5 namespace netdem {
6
7 class Simplex {
8 public:
9   VecNT<Vec3d, 4> points;
10   int size{0};
11
12   Simplex() {}
13
14   Simplex(Vec3d const &a) {
15     size = 1;
16
17     points[0] = a;
18   }
19
20   Simplex(Vec3d const &a, Vec3d const &b) {
21     size = 2;
22
23     points[0] = a;
24     points[1] = b;
25   }
26
27   Simplex(Vec3d const &a, Vec3d const &b, Vec3d const &c) {
28     size = 3;
29
30     points[0] = a;
31     points[1] = b;
32     points[2] = c;
33   }
34
35   Simplex(Vec3d const &a, Vec3d const &b, Vec3d const &c, Vec3d const &d) {
36     size = 4;
37
38     points[0] = a;
39     points[1] = b;
40     points[2] = c;
41     points[3] = d;
42   }
43
44   void PushBack(Vec3d const &p) {
45     points[size] = p;
46     size += 1;
47   }
48
49   void PushFront(Vec3d const &p) {
50     for (int i = size; i > 0; i--) {
51       points[size] = points[size - 1];
52     }
53     points[0] = p;
54     size += 1;
55   }
56 };
57
58 } // namespace netdem
```

8.40 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_hertz_mindlin.cpp File Reference

```
#include "model_hertz_mindlin.hpp"
#include "contact_model_factory.hpp"
#include "contact_pp.hpp"
#include "contact_pw.hpp"
#include "utils_math.hpp"
```

Namespaces

- namespace netdem

8.41 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_hertz_mindlin.hpp File Reference

```
#include "contact_model.hpp"
```

Classes

- class netdem::HertzMindlin

Namespaces

- namespace netdem

8.42 model_hertz_mindlin.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "contact_model.hpp"
4
5 namespace netdem {
6
7 class HertzMindlin : public ContactModel {
8 public:
9   double kn{2.0e7}, kt{1.0e6}, beta{0.7}, mu{0.5};
10
11   HertzMindlin();
12   HertzMindlin(double kn, double kt, double beta, double mu);
13
14   nlohmann::json PackJson() override;
15
16   void InitFromJson(nlohmann::json const &js) override;
17
18   void SetProperty(nlohmann::json const &js) override;
19
20   ContactModel *Clone() const override;
```

```
21
22   void EvaluateForceMoment(ContactForces *const cnt_forces,
23                            CollisionGeometries &cnt_geoms, ContactPP *const cnt,
24                            double dt) const override;
25
26   void EvaluateForceMoment(ContactForces *const cnt_forces,
27                            CollisionGeometries &cnt_geoms, ContactPW *const cnt,
28                            double dt) const override;
29
30   void Print() const override;
31 };
32
33 } // namespace netdem
```

8.43 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/model_linear_spring.cpp File Reference

```
#include "model_linear_spring.hpp"
#include "contact_model_factory.hpp"
#include "contact_pp.hpp"
#include "contact_pw.hpp"
#include "utils_math.hpp"
```

Namespaces

- namespace netdem

8.44 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/model_linear_spring.hpp File Reference

```
#include "contact_model.hpp"
```

Classes

- class netdem::LinearSpring

Namespaces

- namespace netdem

8.45 model_linear_spring.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "contact_model.hpp"
4
5 namespace netdem {
6
17 class LinearSpring : public ContactModel {
18 public:
19   double kn{2e6}, kt{1e6}, beta{0.7}, mu{0.5};
20   bool use_viscous_damping{false};
21
22   LinearSpring();
23   LinearSpring(double kn, double kt, double beta, double mu);
24
25   nlohmann::json PackJson() override;
26
27   void InitFromJson(nlohmann::json const &js) override;
28
29   void SetProperty(nlohmann::json const &js) override;
30
31   ContactModel *Clone() const override;
32
33   void EvaluateForceMoment(ContactForces *const cnt_forces,
34                            CollisionGeometries &cnt_geoms, ContactPP *const cnt,
35                            double dt) const override;
36
37   void EvaluateForceMoment(ContactForces *const cnt_forces,
38                            CollisionGeometries &cnt_geoms, ContactPW *const cnt,
39                            double dt) const override;
40
41   void Print() const override;
42 };
43
44 } // namespace netdem
```

8.46 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/model_parallel_bond.cpp File Reference

```
#include "model_parallel_bond.hpp"
#include "contact_model_factory.hpp"
#include "contact_pp.hpp"
#include "contact_pw.hpp"
#include "utils_math.hpp"
```

Namespaces

- namespace netdem

8.47 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/model_parallel_bond.hpp File Reference

```
#include "contact_model.hpp"
#include "utils_math.hpp"
```

Classes

- class netdem::ParallelBond

Namespaces

- namespace netdem

8.48 model_parallel_bond.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "contact_model.hpp"
4 #include "utils_math.hpp"
5
6 namespace netdem {
7
8 class ContactPP;
9 class ContactPW;
10
11 class ParallelBond : public ContactModel {
12 public:
13   double kn{2e6}, kt{1e6};
14   double max_sig_n{1.0e6}, max_sig_t{1.0e6};
15
16   ParallelBond();
17   ParallelBond(double kn, double kt, double sig_n, double sig_t);
18
19   nlohmann::json PackJson() override;
20
21   void InitFromJson(nlohmann::json const &js) override;
22
23   void SetProperty(nlohmann::json const &js) override;
24
25   ContactModel *Clone() const override;
26
27   void SetRadius(double r);
28
29   void EvaluateForceMoment(ContactForces *const cnt_forces,
30                            BondGeometries &cnt_geoms, ContactPP *const cnt,
31                            double dt) const override;
32
33   void EvaluateForceMoment(ContactForces *const cnt_forces,
34                            BondGeometries &cnt_geoms, ContactPW *const cnt,
35                            double dt) const override;
36
37   void Print() const override;
38 };
39
40 } // namespace netdem
```

8.49 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/model_volume_based.cpp File Reference

```
#include "model_volume_based.hpp"
#include "contact_model_factory.hpp"
#include "contact_pp.hpp"
#include "contact_pw.hpp"
#include "utils_math.hpp"
```

Namespaces

- namespace netdem

8.50 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/model_volume_based.hpp File Reference

```
#include "collision_geometries.hpp"
#include "contact_model.hpp"
#include <unordered_map>
```

Classes

- class netdem::VolumeBased

Namespaces

- namespace netdem

8.51 model_volume_based.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "contact_model.hpp"
5 #include <unordered_map>
6
7 namespace netdem {
8
9 class VolumeBased : public ContactModel {
10 public:
11   int order{2};
12   double kn{2e6}, kt{1e6}, beta{0.7}, mu{0.5};
13
14   VolumeBased();
15   VolumeBased(double kn, double kt, double beta, double mu);
16
17   nlohmann::json PackJson() override;
18
19   void InitFromJson(nlohmann::json const &js) override;
20
21   void SetProperty(nlohmann::json const &js) override;
22
23   ContactModel *Clone() const override;
24
25   void EvaluateForceMoment(ContactForces *const cnt_forces,
26                            CollisionGeometries &cnt_geoms, ContactPP *const cnt,
27                            double dt) const override;
28
29   void EvaluateForceMoment(ContactForces *const cnt_forces,
30                            CollisionGeometries &cnt_geoms, ContactPW *const cnt,
31                            double dt) const override;
32
33   void Print() const override;
34 };
35
36 } // namespace netdem
```

8.52 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_boolean_pp.cpp File Reference

```
#include "solver_boolean_pp.hpp"
#include "cork_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "utils_math.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.53 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_boolean_pp.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pp.hpp"
#include "contact_forces.hpp"
#include "particle.hpp"
#include "shape.hpp"
#include "shape_trimesh.hpp"
#include "utils_math.hpp"
#include "wall.hpp"
```

Classes

- class netdem::SolverBooleanPP

Namespaces

- namespace netdem

8.54 solver_boolean_pp.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pp.hpp"
5 #include "contact_forces.hpp"
6 #include "particle.hpp"
7 #include "shape.hpp"
8 #include "shape_trimesh.hpp"
9 #include "utils_math.hpp"
```

```
10 #include "wall.hpp"
11
12 namespace netdem {
13
17 class SolverBooleanPP : public CollisionSolverPP {
18 public:
19   SolverBooleanPP();
20   SolverBooleanPP(Particle *const p1, Particle *const p2);
21
22   CollisionSolverPP *Clone() const override;
23
24   void Init(Particle *const p1, Particle *const p2) override;
25
26   bool Detect() override;
27
28   void ResolveInit(ContactPP *const cnt, double timestep) override;
29   void ResolveUpdate(ContactPP *const cnt, double timestep) override;
30
31   void ResolveInit(CollisionGeometries *const cnt_geoms, double timestep,
32                    const VecXT<Vec3d> &vertices, const VecXT<Vec3i> &facets,
33                    const VecXT<int> &facets_of_1or2);
34   void ResolveUpdate(CollisionGeometries *const cnt_geoms, double timestep,
35                      const VecXT<Vec3d> &vertices, const VecXT<Vec3i> &facets,
36                      const VecXT<int> &facets_of_1or2);
37
38   void ResolveInit_Equivalent(CollisionGeometries *const cnt_geoms,
39                               double timestep);
40   void ResolveUpdate_Equivalent(CollisionGeometries *const cnt_geoms,
41                                 double timestep);
42
43   STLModel GetContactTriMesh(int id);
44
45 protected:
46   double bound_sphere_radius_1, bound_sphere_radius_2;
47   Vec3d dpos_12;
48
49   STLModel *stl_model_1, *stl_model_2;
50
51   VecXT<Vec3d> vertices_isct;
52   VecXT<Vec3i> facets_isct;
53
54   VecXT<int> facets_birth_ids;
55
56   VecXT<VecXT<Vec3i>> comp_facets_list;
57   VecXT<VecXT<int>> comp_facets_of_1or2_list;
58
59   void ClearIntersectInfo();
60   void SeperateComponents();
61 };
62
63 } // namespace netdem
```

8.55 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_boolean_pw.cpp File Reference

```
#include "solver_boolean_pw.hpp"
#include "cork_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "utils_math.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.56 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_boolean_pw.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pw.hpp"
#include "contact_forces.hpp"
#include "particle.hpp"
#include "shape.hpp"
#include "shape_trimesh.hpp"
#include "wall.hpp"
```

Classes

- class netdem::SolverBooleanPW

Namespaces

- namespace netdem

8.57 solver_boolean_pw.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pw.hpp"
5 #include "contact_forces.hpp"
6 #include "particle.hpp"
7 #include "shape.hpp"
8 #include "shape_trimesh.hpp"
9 #include "wall.hpp"
10
11 namespace netdem {
12
16 class SolverBooleanPW : public CollisionSolverPW {
17 public:
18   SolverBooleanPW();
19   SolverBooleanPW(Particle *const p, Wall *const w);
20
21   CollisionSolverPW *Clone() const override;
22
23   void Init(Particle *const p, Wall *const w) override;
24
25   bool Detect() override;
26
27   void ResolveInit(ContactPW *const cnt, double timestep) override;
28   void ResolveUpdate(ContactPW *const cnt, double timestep) override;
29
30   void ResolveInit(CollisionGeometries *const cnt_geoms, double timestep,
31                    const VecXT<Vec3d> &vertices, const VecXT<Vec3i> &facets,
32                    const VecXT<int> &facets_of_1or2);
33   void ResolveUpdate(CollisionGeometries *const cnt_geoms, double timestep,
34                      const VecXT<Vec3d> &vertices, const VecXT<Vec3i> &facets,
35                      const VecXT<int> &facets_of_1or2);
36
37   void ResolveInit_Equivalent(CollisionGeometries *const cnt_geoms,
38                               double timestep);
39   void ResolveUpdate_Equivalent(CollisionGeometries *const cnt_geoms,
40                                 double timestep);
41
42   STLModel GetContactTriMesh(int id);
43
44 protected:
```

```
45   double bound_sphere_radius_1, bound_sphere_radius_2;
46   Vec3d dpos_12;
47
48   STLModel *stl_model_1, *stl_model_2;
49
50   VecXT<Vec3d> vertices_isct;
51   VecXT<Vec3i> facets_isct;
52
53   VecXT<int> facets_birth_ids;
54
55   VecXT<VecXT<Vec3i>> comp_facets_list;
56   VecXT<VecXT<int>> comp_facets_of_1or2_list;
57
58   void ClearIntersectInfo();
59   void SeperateComponents();
60 };
61
62 } // namespace netdem
```

8.58 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_gjk_pp.cpp File Reference

```
#include "solver_gjk_pp.hpp"
#include "stl_model.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
#include <iostream>
#include <string>
```

Namespaces

- namespace netdem

8.59 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_gjk_pp.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pp.hpp"
#include "gjk_simplex.hpp"
#include "particle.hpp"
#include "shape.hpp"
#include "wall.hpp"
```

Classes

- class netdem::SolverGJKPP

Namespaces

- namespace netdem

8.60 solver_gjk_pp.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pp.hpp"
5 #include "gjk_simplex.hpp"
6 #include "particle.hpp"
7 #include "shape.hpp"
8 #include "wall.hpp"
9
10 namespace netdem {
11
15 class SolverGJKPP : public CollisionSolverPP {
16 public:
17   double erosion_ratio_initial{0.01}, erosion_ratio_increment{0.01};
18   bool use_erosion{false};
19
20   SolverGJKPP();
21   SolverGJKPP(Particle *const p1, Particle *const p2);
22
23   CollisionSolverPP *Clone() const override;
24
25   void Init(Particle *const p1, Particle *const p2) override;
26
27   bool Detect() override;
28
29   void ResolveInit(ContactPP *const cnt, double timestep) override;
30   void ResolveUpdate(ContactPP *const cnt, double timestep) override;
31
32   void ResolveInit(CollisionGeometries *const cnt_geoms, double timestep);
33   void ResolveUpdate(CollisionGeometries *const cnt_geoms, double timestep);
34
35 protected:
36   Shape *shape_1{nullptr}, *shape_2{nullptr};
37   double bound_sphere_radius_1, bound_sphere_radius_2;
38
39   Vec3d dpos_12, dpos_12_ref;
40   Vec4d dquat_12, dquat_12_conj;
41
42   Simplex simplex_after_gjk;
43
44   bool GJK();
45
46   // return: cnt_len_n, cnt_dir_n, cnt_pos
47   std::tuple<double, Vec3d, Vec3d> GJK_EROSION();
48
49   // return: cnt_len_n, cnt_dir_n, cnt_pos
50   std::tuple<double, Vec3d, Vec3d> EPA();
51
52   // return: cnt_pos, and true if success
53   std::tuple<Vec3d, bool> GetContactPoint(Vec3d const &dir);
54
55   // return: cnt_pos, and true if success
56   std::tuple<Vec3d, bool>
57   GetContactPoint_PlaneCase(Vec3d const &dir, const VecXT<Vec3d> &pos_vec_1,
58                             const VecXT<Vec3d> &pos_vec_2);
59
60   inline Vec3d MinkowskiDiff(Vec3d const &dir, double erosion_ratio = 0);
61
62   void UpdateSimplex(Simplex *const s, Vec3d *const dir, double *const min_dist,
63                      bool *const cnt_flag);
64   void UpdateSimplexLine(Simplex *const s, Vec3d *const dir,
65                          double *const min_dist, bool *const cnt_flag);
66   void UpdateSimplexTriangle(Simplex *const s, Vec3d *const dir,
67                              double *const min_dist, bool *const cnt_flag);
68   void UpdateSimplexTetrahedron(Simplex *const s, Vec3d *const dir,
69                                 double *const min_dist, bool *const cnt_flag);
70
71   // return: normal, dist_from_origin
72   inline std::tuple<Vec3d, double>
73   GetFacetNormal(Vec3d const &a, Vec3d const &b, Vec3d const &c);
74
75   void GetLooseEdges(VecXT<Vec2i> *const edges, Vec3i const &facet);
76
77   void GetIntersections(VecXT<Vec3d> *const intersects, Vec3d const &dir_n,
78                         Vec3d const &l1_p1, Vec3d const &l1_p2,
79                         Vec3d const &l2_p1, Vec3d const &l2_p2);
80
81   void GetIntersectionsAggresive(VecXT<Vec3d> *const intersects,
82                                  Vec3d const &dir_n, Vec3d const &l1_p1,
83                                  Vec3d const &l1_p2, Vec3d const &l2_p1,
84                                  Vec3d const &l2_p2);
85
```

```
86   void SortVertices(VecXT<Vec3d> *const pos_vec, Vec3d const &dir_n);
87
88   bool IsInsidePolygon(VecXT<Vec3d> const &pos_vec, Vec3d const &dir_n,
89                        Vec3d const &pos);
90
91   Vec3d GetPolygonCentroid(VecXT<Vec3d> const &pos_vec, Vec3d const &dir_n);
92 };
93
94 } // namespace netdem
```

8.61 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_gjk_pw.cpp File Reference

```
#include "solver_gjk_pw.hpp"
#include "stl_model.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
#include <iostream>
#include <string>
```

Namespaces

- namespace netdem

8.62 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_gjk_pw.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pw.hpp"
#include "gjk_simplex.hpp"
#include "particle.hpp"
#include "shape.hpp"
#include "wall.hpp"
```

Classes

- class netdem::SolverGJKPW

Namespaces

- namespace netdem

8.63 solver_gjk_pw.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pw.hpp"
5 #include "gjk_simplex.hpp"
6 #include "particle.hpp"
7 #include "shape.hpp"
8 #include "wall.hpp"
9
10 namespace netdem {
11
15 class SolverGJKPW : public CollisionSolverPW {
16 public:
17   double erosion_ratio_initial{0.01}, erosion_ratio_increment{0.01};
18   bool use_erosion{false};
19
20   SolverGJKPW();
21   SolverGJKPW(Particle *const p, Wall *const w);
22
23   CollisionSolverPW *Clone() const override;
24
25   void Init(Particle *const p, Wall *const w) override;
26
27   bool Detect() override;
28
29   void ResolveInit(ContactPW *const cnt, double timestep) override;
30   void ResolveUpdate(ContactPW *const cnt, double timestep) override;
31
32   void ResolveInit(CollisionGeometries *const cnt_geoms, double timestep);
33   void ResolveUpdate(CollisionGeometries *const cnt_geoms, double timestep);
34
35 protected:
36   Shape *shape_1{nullptr}, *shape_2{nullptr};
37   double bound_sphere_radius_1, bound_sphere_radius_2;
38
39   Vec3d dpos_12, dpos_12_ref;
40   Vec4d dquat_12, dquat_12_conj;
41
42   Simplex simplex_after_gjk;
43
44   bool GJK();
45
46   // return: cnt_len_n, cnt_dir_n, cnt_pos
47   std::tuple<double, Vec3d, Vec3d> GJK_EROSION();
48
49   // return: cnt_len_n, cnt_dir_n, cnt_pos
50   std::tuple<double, Vec3d, Vec3d> EPA();
51
52   // return: cnt_pos, and true if success
53   std::tuple<Vec3d, bool> GetContactPoint(Vec3d const &dir);
54
55   // return: cnt_pos, and true if success
56   std::tuple<Vec3d, bool>
57   GetContactPoint_PlaneCase(Vec3d const &dir, const VecXT<Vec3d> &pos_vec_1,
58                             const VecXT<Vec3d> &pos_vec_2);
59
60   inline Vec3d MinkowskiDiff(Vec3d const &dir, double erosion_ratio = 0);
61
62   void UpdateSimplex(Simplex *const s, Vec3d *const dir, double *const min_dist,
63                      bool *const cnt_flag);
64   void UpdateSimplexLine(Simplex *const s, Vec3d *const dir,
65                          double *const min_dist, bool *const cnt_flag);
66   void UpdateSimplexTriangle(Simplex *const s, Vec3d *const dir,
67                              double *const min_dist, bool *const cnt_flag);
68   void UpdateSimplexTetrahedron(Simplex *const s, Vec3d *const dir,
69                                 double *const min_dist, bool *const cnt_flag);
70
71   // return: normal, dist_from_origin
72   inline std::tuple<Vec3d, double>
73   GetFacetNormal(Vec3d const &a, Vec3d const &b, Vec3d const &c);
74
75   void GetLooseEdges(VecXT<Vec2i> *const edges, Vec3i const &facet);
76
77   void GetIntersections(VecXT<Vec3d> *const intersects, Vec3d const &dir_n,
78                         Vec3d const &l1_p, Vec3d const &l1_w, Vec3d const &l2_p,
79                         Vec3d const &l2_w);
80
81   void GetIntersectionsAggresive(VecXT<Vec3d> *const intersects,
82                                  Vec3d const &dir_n, Vec3d const &l1_p,
83                                  Vec3d const &l1_w, Vec3d const &l2_p,
84                                  Vec3d const &l2_w);
85
```

```
86   void SortVertices(VecXT<Vec3d> *const pos_vec, Vec3d const &dir_n);
87
88   bool IsInsidePolygon(VecXT<Vec3d> const &pos_vec, Vec3d const &dir_n,
89                        Vec3d const &pos);
90
91   Vec3d GetPolygonCentroid(VecXT<Vec3d> const &pos_vec, Vec3d const &dir_n);
92 };
93
94 } // namespace netdem
```

8.64 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_sdf_pp.cpp File Reference

```
#include "solver_sdf_pp.hpp"
#include "utils_math.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.65 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_sdf_pp.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pp.hpp"
#include "particle.hpp"
#include "shape.hpp"
```

Classes

- class netdem::SolverSDFPP

Namespaces

- namespace netdem

8.66 solver_sdf_pp.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pp.hpp"
5 #include "particle.hpp"
6 #include "shape.hpp"
7
8 namespace netdem {
9
13 class SolverSDFPP : public CollisionSolverPP {
14 public:
15   enum PotentialType { linear, hertz };
16   int potential_type{PotentialType::linear};
17   bool solve_two_sides{false};
18
19   SolverSDFPP();
20   SolverSDFPP(Particle *const p1, Particle *const p2);
21
22   CollisionSolverPP *Clone() const override;
23
24   void Init(Particle *const p1, Particle *const p2) override;
25
26   bool Detect() override;
27
28   void ResolveInit(ContactPP *const cnt, double timestep) override;
29   void ResolveUpdate(ContactPP *const cnt, double timestep) override;
30
31   void ResolveInit(CollisionGeometries *const cnt_geoms, double timestep);
32   void ResolveUpdate(CollisionGeometries *const cnt_geoms, double timestep);
33
34   void ResolveInitP2ToP1(CollisionGeometries *const cnt_geoms, double timestep);
35   void ResolveUpdateP2ToP1(CollisionGeometries *const cnt_geoms,
36                            double timestep);
37
38   void ResolveInitP1ToP2(CollisionGeometries *const cnt_geoms, double timestep);
39   void ResolveUpdateP1ToP2(CollisionGeometries *const cnt_geoms,
40                            double timestep);
41
42   double Potential(double dist, Shape *shape);
43
44 private:
45   bool solve_p1_to_p2{false}, solve_p2_to_p1{false};
46
47   double bound_sphere_radius_1, bound_sphere_radius_2;
48
49   // particle 1 as reference particle
50   Vec3d pos_1, dpos_12, dpos_12_ref;
51   Vec4d quat_1, dquat_12;
52
53   // particle 2 as reference particle
54   Vec3d pos_2, dpos_21, dpos_21_ref;
55   Vec4d quat_2, dquat_21;
56
57   Shape *shape_1{nullptr}, *shape_2{nullptr};
58
59   VecXT<int> node_id_list;
60   VecXT<double> node_dist_list;
61 };
62
63 } // namespace netdem
```

8.67 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sdf_pw.cpp File Reference

```
#include "solver_sdf_pw.hpp"
#include "utils_math.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.68 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_sdf_pw.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pw.hpp"
#include "particle.hpp"
#include "shape.hpp"
```

Classes

- class netdem::SolverSDFPW

Namespaces

- namespace netdem

8.69 solver_sdf_pw.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pw.hpp"
5 #include "particle.hpp"
6 #include "shape.hpp"
7
8 namespace netdem {
9
13 class SolverSDFPW : public CollisionSolverPW {
14 public:
15   enum PotentialType { linear, hertz };
16   int potential_type{PotentialType::linear};
17   bool solve_two_sides{false};
18
19   SolverSDFPW();
20   SolverSDFPW(Particle *const p, Wall *const w);
21
22   CollisionSolverPW *Clone() const override;
23
24   void Init(Particle *const p, Wall *const w) override;
25
26   bool Detect() override;
27
28   void ResolveInit(ContactPW *const cnt, double timestep) override;
29   void ResolveUpdate(ContactPW *const cnt, double timestep) override;
30
31   void ResolveInit(CollisionGeometries *const cnt_geoms, double timestep);
32   void ResolveUpdate(CollisionGeometries *const cnt_geoms, double timestep);
33
34   void ResolveInitWToP(CollisionGeometries *const cnt_geoms, double timestep);
35   void ResolveUpdateWToP(CollisionGeometries *const cnt_geoms, double timestep);
36
37   void ResolveInitPToW(CollisionGeometries *const cnt_geoms, double timestep);
38   void ResolveUpdatePToW(CollisionGeometries *const cnt_geoms, double timestep);
39
40   double Potential(double dist, Shape *shape);
```

```
41
42 private:
43   bool solve_p_to_w{false}, solve_w_to_p{false};
44
45   double bound_sphere_radius_1, bound_sphere_radius_2;
46
47   // particle as reference
48   Vec3d pos_1, dpos_12, dpos_12_ref;
49   Vec4d quat_1, dquat_12;
50
51   // wall as reference
52   Vec3d pos_2, dpos_21, dpos_21_ref;
53   Vec4d quat_2, dquat_21;
54
55   Shape *shape_1{nullptr}, *shape_2{nullptr};
56
57   VecXT<int> node_id_list;
58   VecXT<double> node_dist_list;
59 };
60
61 } // namespace netdem
```

8.70 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_sphere_plane.cpp File Reference

```
#include "solver_sphere_plane.hpp"
#include "utils_math.hpp"
#include "wall.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.71 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_sphere_plane.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pw.hpp"
#include "particle.hpp"
#include "shape.hpp"
#include "shape_plane.hpp"
#include "shape_sphere.hpp"
#include "wall.hpp"
```

Classes

- class netdem::SolverSpherePlane

Namespaces

- namespace netdem

8.72 solver_sphere_plane.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pw.hpp"
5 #include "particle.hpp"
6 #include "shape.hpp"
7 #include "shape_plane.hpp"
8 #include "shape_sphere.hpp"
9 #include "wall.hpp"
10
11 namespace netdem {
12
20 class SolverSpherePlane : public CollisionSolverPW {
21 public:
22   SolverSpherePlane();
23   SolverSpherePlane(Particle *const p, Wall *const w);
24
25   CollisionSolverPW *Clone() const override;
26
27   void Init(Particle *const p, Wall *const w) override;
28
29   bool Detect() override;
30
31   void ResolveInit(ContactPW *const cnt, double timestep) override;
32   void ResolveUpdate(ContactPW *const cnt, double timestep) override;
33
34   void ResolveInit(CollisionGeometries *const cnt_geoms, double timestep);
35   void ResolveUpdate(CollisionGeometries *const cnt_geoms, double timestep);
36
37 private:
38   Vec3d dpos_12;
39
40   double radius_1, dist_pc_to_plane;
41   Vec3d dir_n, cnt_pos;
42
43   Plane *plane;
44 };
45
46 } // namespace netdem
```

8.73 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_sphere_sphere.cpp File Reference

```
#include "solver_sphere_sphere.hpp"
#include "utils_math.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.74 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/dem/solver_sphere_sphere.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pp.hpp"
#include "contact_forces.hpp"
#include "particle.hpp"
#include "shape.hpp"
#include "shape_sphere.hpp"
#include "wall.hpp"
```

Classes

- class netdem::SolverSphereSphere

Namespaces

- namespace netdem

8.75 solver_sphere_sphere.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pp.hpp"
5 #include "contact_forces.hpp"
6 #include "particle.hpp"
7 #include "shape.hpp"
8 #include "shape_sphere.hpp"
9 #include "wall.hpp"
10
11 namespace netdem {
12
16 class SolverSphereSphere : public CollisionSolverPP {
17 public:
18   SolverSphereSphere();
19   SolverSphereSphere(Particle *const p1, Particle *const p2);
20
21   CollisionSolverPP *Clone() const override;
22
23   void Init(Particle *const p1, Particle *const p2) override;
24
25   bool Detect() override;
26
27   void ResolveInit(ContactPP *const cnt, double timestep) override;
28   void ResolveUpdate(ContactPP *const cnt, double timestep) override;
29
30   void ResolveInit(CollisionGeometries *const cnt_geoms, double timestep);
31   void ResolveUpdate(CollisionGeometries *const cnt_geoms, double timestep);
32
33 private:
34   double radius_1, radius_2;
35   Vec3d dpos_12;
36 };
37
38 } // namespace netdem
```

8.76 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sphere_triangle.cpp File Reference

```
#include "solver_sphere_triangle.hpp"
#include "utils_math.hpp"
#include <iostream>
```

8.77 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/dem/solver_sphere_triangle.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pw.hpp"
#include "contact_forces.hpp"
#include "particle.hpp"
#include "shape.hpp"
#include "shape_sphere.hpp"
#include "shape_triangle.hpp"
#include "wall.hpp"
```

Classes

- class netdem::SolverSphereTriangle

Namespaces

- namespace netdem

8.78 solver_sphere_triangle.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pw.hpp"
5 #include "contact_forces.hpp"
6 #include "particle.hpp"
7 #include "shape.hpp"
8 #include "shape_sphere.hpp"
9 #include "shape_triangle.hpp"
10 #include "wall.hpp"
11
12 namespace netdem {
13
14 class SolverSphereTriangle : public CollisionSolverPW {
15 public:
16   SolverSphereTriangle();
```

```
17   SolverSphereTriangle(Particle *const p, Wall *const w);
18
19   CollisionSolverPW *Clone() const override;
20
21   void Init(Particle *const p, Wall *const w) override;
22
23   bool Detect() override;
24
25   void ResolveInit(ContactPW *const cnt, double timestep) override;
26   void ResolveUpdate(ContactPW *const cnt, double timestep) override;
27
28   void ResolveInit(CollisionGeometries *const cnt_geoms, double timestep);
29   void ResolveUpdate(CollisionGeometries *const cnt_geoms, double timestep);
30
31 private:
32   double radius_1;
33   double dist_pc_to_tri;
34
35   Vec3d cnt_pos, cnt_dir_n;
36   double cnt_len_n, cnt_weight;
37
38   Triangle triangle;
39
40   void UpdateLocalTriangle();
41   void ResolvePotentialContact();
42
43   inline Vec3d GetLineCircleIntersection(double cr, Vec3d const &cc,
44                                          double dist_to_line,
45                                          Vec3d const &dir_n_cross_line,
46                                          Vec3d const &v0, Vec3d const &v1);
47   inline double GetTriangleArea(Vec3d const &v0, Vec3d const &v1,
48                                 Vec3d const &v2);
49   inline double GetCircleSegmentArea(double cr, double signed_d);
50 };
51
52 } // namespace netdem
```

8.79 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/domain/cell.cpp File Reference

```
#include "cell.hpp"
#include <iostream>
```

8.80 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/domain/cell.hpp File Reference

```
#include "particle.hpp"
#include "wall.hpp"
```

Classes

- class netdem::Cell

Namespaces

- namespace netdem

8.81 cell.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "particle.hpp"
4 #include "wall.hpp"
5
6 namespace netdem {
7
17 class Cell {
18 public:
19   Vec3d bound_min{-0.5, -0.5, -0.5}, bound_max{0.5, 0.5, 0.5};
20
22   VecXT<std::pair<Particle *, int>> linked_particle_list;
23
25   VecXT<std::pair<Wall *, int>> linked_wall_list;
26
27   Cell();
28   Cell(Vec3d const &bmin, Vec3d const &bmax);
29
30   bool IsJudgeCell(Particle const &p, Particle const &q);
31   bool IsJudgeCell(Particle const &p, Wall const &w);
32
33   void ClearLinkedLists();
34
36   STLModel GetSTLModel();
37
38   ~Cell();
39
40   void Print();
41 };
42
43 } // namespace netdem
```

8.82 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/domain/cell_manager.cpp File Reference

```
#include "cell_manager.hpp"
#include "utils_math.hpp"
#include <cmath>
#include <iostream>
```

8.83 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/domain/cell_manager.hpp File Reference

```
#include "cell.hpp"
```

Classes

- class netdem::CellManager

Namespaces

- namespace netdem

8.84 cell_manager.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "cell.hpp"
4
5 namespace netdem {
6
7 class CellManager {
8 public:
9   Vec3d bound_min{-0.5, -0.5, -0.5};
10   Vec3d bound_max{0.5, 0.5, 0.5};
11
12   Vec3d spacing{1.0, 1.0, 1.0};
13   Vec3i cell_size{3, 3, 3};
14
15   VecXT<VecXT<VecXT<Cell>» cell_list;
16
17   CellManager();
18
19   void Init();
20
21   void SetBound(double bmin_x, double bmin_y, double bmin_z, double bmax_x,
22                 double bmax_y, double bmax_z);
23   void SetSpacing(double s_x, double s_y, double s_z);
24
25   std::tuple<Vec3i, Vec3i> GetOverlappedCells(Vec3d const &bmin,
26                                               Vec3d const &bmax);
27
29   STLModel GetSTLModel();
30 };
31
32 } // namespace netdem
```

8.85 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/domain.cpp File Reference

```
#include "domain.hpp"
#include <iostream>
```

8.86 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/domain.hpp File Reference

```
#include "cell_manager.hpp"
#include "particle.hpp"
#include "particle_data.hpp"
#include "wall.hpp"
```

Classes

- class netdem::Domain

Namespaces

- namespace netdem

8.87 domain.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "cell_manager.hpp"
4 #include "particle.hpp"
5 #include "particle_data.hpp"
6 #include "wall.hpp"
7
8 namespace netdem {
9
10 class Scene;
11
12 class Domain {
13 public:
15   int my_rank{0}, num_procs{0};
16
19   Vec3d bound_min{-0.5, -0.5, -0.5}, bound_max{0.5, 0.5, 0.5};
20
22   CellManager cell_manager;
23
29   VecXT<std::pair<Particle *, int>> outer_particle_list;
30
31   Domain();
32   Domain(Vec3d const &bmin, Vec3d const &bmax);
33
34   void Init();
35
37   void SetBound(double bmin_x, double bmin_y, double bmin_z, double bmax_x,
38                 double bmax_y, double bmax_z);
39
42   bool IsJudgeDomain(Particle const &p, Particle const &q);
43   bool IsJudgeDomain(Particle const &p, Wall const &w);
44
47   bool IsBelongToDomain(Particle const &p);
48   bool IsBelongToDomain(ParticleData const &p);
49
52   bool IsParticleProxyToSend(Particle const &p);
53
57   bool IsParticleProxyToRecv(Particle const &p);
58   bool IsParticleProxyToRecv(ParticleData const &p);
59
61   void Print();
62
63   void ClearLinkedLists();
64
66   STLModel GetSTLModel();
67
68   ~Domain();
69 };
70
71 }; // namespace netdem
```

8.88 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/domain/domain_manager.cpp File Reference

```
#include "domain_manager.hpp"
#include "simulation.hpp"
```

8.89 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/domain/domain_manager.hpp File Reference

```
#include "domain.hpp"
```

Classes

- class netdem::DomainManager

Namespaces

- namespace netdem

8.90 domain_manager.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "domain.hpp"
4
5 namespace netdem {
6
7 class Simulation;
8
9 class DomainManager {
10 public:
13   Vec3d bound_min{-0.5, -0.5, -0.5}, bound_max{0.5, 0.5, 0.5};
14
18   Vec3i num_div{1, 1, 1};
19
21   VecXT<Domain> domain_list;
22
23   DomainManager();
24
26   void Init(Simulation *s);
27   void Init();
28
29   void SetBound(double bmin_x, double bmin_y, double bmin_z, double bmax_x,
30                 double bmax_y, double bmax_z);
31   void SetDecomposition(int num_div_x, int num_div_y, int num_div_z);
32
33   Domain *GetSelfDomain();
34
35 private:
36   Simulation *sim{nullptr};
37 };
38
39 }; // namespace netdem
```

8.91 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/deformable_particle.cpp File Reference

```
#include "deformable_particle.hpp"
#include "contact_pp.hpp"
#include "contact_pw.hpp"
```

Namespaces

- namespace netdem

8.92 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/fem/deformable_particle.hpp File Reference

```
#include "fem_simulator.hpp"
#include "particle.hpp"
#include "shape_trimesh.hpp"
```

Classes

- class netdem::DeformableParticle

Namespaces

- namespace netdem

8.93 deformable_particle.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "fem_simulator.hpp"
4 #include "particle.hpp"
5 #include "shape_trimesh.hpp"
6
7 namespace netdem {
8
9 class DeformableParticle : public Particle {
10 public:
11   // shape
12   TriMesh *trimesh{nullptr};
13
14   // fem
15   TetMesh tetmesh;
16   FEMSimulator fem_simulator;
17
18   int mesh_res{20};
19
20   // constructor
21   DeformableParticle();
22
23   Particle *Clone() const override;
24
25   void SetShape(Shape *s) override;
26   void SetDensity(double dens) override;
27
28   void SetPosition(double x, double y, double z) override;
29   void SetRodrigues(double angle, double axis_x, double axis_y,
30                     double axis_z) override;
31   void SetQuaternion(double q_0, double q_1, double q_2, double q_3) override;
32
33   void SetVelocity(double v_x, double v_y, double v_z) override;
34
35   Vec3d GetVelocity(Vec3d const &pos) override;
36
37   void AddForce(int node_id, Vec3d const &f);
38   void ClearForce() override;
```

```
39
40   void ApplyContactForce(ContactPP const *cnt) override;
41   void ApplyContactForce(ContactPW const *cnt) override;
42
43   void UpdateMotion(double dt) override;
44
45   void UpdateShape();
46   void UpdateBound() override;
47
48   void SaveSurfaceAsVTK(std::string const &filename);
49   void SaveAsVTK(std::string const &filename) override;
50
51   ~DeformableParticle() override;
52
53 private:
54   void InitFEMSimulator();
55 };
56
57 } // namespace netdem
```

8.94 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/fem_simulator.cpp File Reference

```
#include "fem_simulator.hpp"
#include "igl_wrapper.hpp"
#include "stl_model.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
```

8.95 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/fem_simulator.hpp File Reference

```
#include "tetmesh.hpp"
```

Classes

- class netdem::FEMSimulator

Namespaces

- namespace netdem

8.96 fem_simulator.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "tetmesh.hpp"
4
5 namespace netdem {
6
7 // use tetradron element with 4 nodes
8 class FEMSimulator {
9 public:
10   // default as latex material. ref: Qu et al. CG, 2019.
11   double neo_k{6.94e5}, neo_mu{5.21e5}, density{500.0};
12
13   double damp_coef{0.7};
14   Vec3d gravity_coef{0.0, 0.0, 0.0};
15
16   double timestep{1.0e-4};
17
18   // tet mesh
19   VecXT<Vec3d> nodes;
20   VecXT<Vec4i> elements;
21
22   VecXT<Vec3d> nodes_ref;
23
24   VecXT<Vec3i> bound_facets;
25   VecXT<int> bound_nodes;
26
27   // elemental stress
28   VecXT<double> elemental_vol;
29   VecXT<VecNT<double, 6>> elemental_stress;
30
31   // use dynamic lumped mass to update node positions
32   VecXT<double> nodal_vols;
33   VecXT<Vec3d> nodal_vels;
34
35   // element boundary conditions, prescribed forces
36   VecXT<Vec3d> bc_facet_forces;
37
38   // node boundary conditions, prescribed velocities
39   // the vector is of the same length of the nodes VecXT
40   // the last three element indicate if uses prescribed velocities in this dim
41   VecXT<VecNT<double, 6>> bc_nodal_vels;
42
43 public:
44   FEMSimulator();
45
46   void SetMesh(const TetMesh &tetmesh);
47
48   void Init();
49
50   void SetBCNodalVelocity(int nid, double vx, double vy, double vz,
51                           bool use_prescribed_vx, bool use_prescribed_vy,
52                           bool use_prescribed_vz);
53
54   void AddBCFacetForce(int bc_fid, double fx, double fy, double fz);
55
56   void SetNodalVels(double v_x, double v_y, double v_z);
57
58   void ClearBoundaryCondition();
59
60   void Solve(double dt);
61
62   VecXT<Vec3d> GetNodalPositions(VecXT<int> nids);
63   VecXT<Vec3d> GetNodalDisps(VecXT<int> nids);
64   VecXT<Vec3d> GetNodalVels(VecXT<int> nids);
65
66   void SaveAsVTK(std::string const &file_name);
67
68 protected:
69   VecXT<Vec3d> nodal_forces_int;
70   VecXT<Vec3d> nodal_forces_ext;
71
72   VecXT<Vec3d> nodal_vels_ave;
73
74   void Advance(double dt);
75
76   void InitInitialCondition();
77
78   double GetElementVolume(Vec3d const &v0, Vec3d const &v1, Vec3d const &v2,
79                           Vec3d const &v3);
80
81   Mat3d GetDeformationGradient(Vec3d const &v0_new, Vec3d const &v1_new,
82                                Vec3d const &v2_new, Vec3d const &v3_new,
```

```
83                                Vec3d const &v0_ref, Vec3d const &v1_ref,
84                                Vec3d const &v2_ref, Vec3d const &v3_ref);
85
86   Mat3d GetCauchyStress(Mat3d const &def_grad);
87
88   // return: internal_force, 4 nodes by 3 dims
89   MatNd<4, 3> GetInternalForces(Mat3d const &cauchy_stress, Vec3d const &v0,
90                                 Vec3d const &v1, Vec3d const &v2,
91                                 Vec3d const &v3);
92 };
93
94 } // namespace netdem
```

8.97 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/fem/membrane.cpp File Reference

```
#include "membrane.hpp"
#include "igl_wrapper.hpp"
#include "stl_model.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
```

8.98 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/fem/membrane.hpp File Reference

```
#include "utils_math.hpp"
#include <string>
```

Classes

- class netdem::Membrane

Namespaces

- namespace netdem

8.99 membrane.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_math.hpp"
4 #include <string>
5
6 namespace netdem {
7
8 class Membrane {
9 public:
10   double radius{0.25}, height{1.0}, mesh_size{0.1};
```

```
11   Vec3d center{0, 0, 0};
12
13   // ref: Qu et al. CG, 2019.
14   double neo_k{6.94e5}, neo_mu{5.21e5}, density{500.0}, thickness{0.3e-3};
15
16   double damp_coef{0.7};
17
18   double timestep{1.0e-4};
19
20   // triangle mesh
21   VecXT<Vec3d> nodes;
22   VecXT<Vec3i> elements;
23
24   VecXT<Vec3d> elemental_stress;
25
26   // nodal properties, for updating motion using lumped mass approach
27   VecXT<double> nodal_vols;
28   VecXT<Vec3d> nodal_vels;
29
30   // boundary condition
31   VecXT<double> bc_facet_pressure;
32   VecXT<Vec3d> bc_facet_forces;
33
34   // the last three element indicate if uses prescribed velocities in this dim
35   VecXT<VecNT<double, 6>> bc_nodal_vels;
36
37   Membrane(double radius, double height);
38
39   Membrane(double radius, double height, double mesh_size);
40
41   Membrane(double radius, double height, double mesh_size, double center_x,
42            double center_y, double center_z);
43
44   void Remesh(double ele_size);
45
46   void Init();
47
48   void SetBCNodalVelocity(int nid, double vx, double vy, double vz,
49                           bool use_prescribed_vx, bool use_prescribed_vy,
50                           bool use_prescribed_vz);
51
52   void Solve(double dt);
53
54   void SaveAsVTK(std::string const &file_name);
55
56   ~Membrane();
57
58 protected:
59   double ref_ele_width{0.1}, ref_ele_height{0.1}, ref_ele_area{0.05};
60
61   VecXT<Vec3d> nodal_forces_int;
62   VecXT<Vec3d> nodal_forces_ext;
63
64   void Advance(double dt);
65
66   void InitMesh();
67   void InitInitialCondition();
68
69   // the functions below assume plane stress condition and elements with the
70   // same ref width and height. Node 0 is at origin, edge 01 is of
71   // ref_ele_width, node 2 is of ref_ele_height distance to edge 01. Projection
72   // of node 2 onto edge 01 is at the middle of edge 01.
73   //    2
74   //   / \
75   //  0---1
76
77   Mat2d GetDeformationGradient(Vec3d const &v0, Vec3d const &v1,
78                                Vec3d const &v2);
79
80   Mat2d GetCauchyStress(Mat2d const &def_grad);
81
82   // return: internal_force and external force, 3 nodes by 3 dims
83   std::tuple<Mat3d, Mat3d> GetGlobalForces(Mat2d const &cauchy_stress,
84                                            double pressure, Vec3d const &v0,
85                                            Vec3d const &v1, Vec3d const &v2);
86 };
87
88 } // namespace netdem
```

8.100 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/tetmesh.cpp File Reference

```
#include "tetmesh.hpp"
#include "cgal_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "utils_io.hpp"
#include <cstring>
#include <fstream>
#include <iostream>
#include <memory>
#include <sstream>
#include <unordered_set>
```

8.101 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/fem/tetmesh.hpp File Reference

```
#include "stl_model.hpp"
#include "utils_macros.hpp"
#include <string>
```

Classes

- class netdem::TetMesh

Namespaces

- namespace netdem

8.102 tetmesh.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "stl_model.hpp"
4 #include "utils_macros.hpp"
5 #include <string>
6
7 namespace netdem {
8
9 class TetMesh {
10 public:
11   VecXT<Vec3d> nodes;
12   VecXT<Vec4i> elements;
13
14   // connectivity
15   VecXT<Vec3i> bound_facets;
16   VecXT<Vec2i> bound_edges;
17   VecXT<int> bound_nodes;
```

```
18
19   // surface triangle mesh, for dem to apply contact forces
20   VecXT<Vec3d> surface_nodes;
21   VecXT<Vec3i> surface_facets;
22
23   // ids of the bound_facets that contains the node
24   VecXT<VecXT<int>> surface_node_linked_bounaries;
25   VecXT<double> bound_facet_areas;
26
27 public:
28   TetMesh();
29
30   TetMesh(const VecXT<Vec3d> &tv, const VecXT<Vec4i> &tt);
31
32   TetMesh(STLModel const &stl_model, double mesh_size);
33
34   TetMesh(const VecXT<Vec3d> &vv, const VecXT<Vec3i> &ff, double mesh_size);
35
36   void Init();
37
38   STLModel GetSurfaceSTL();
39
40   void SaveAsVTK(std::string const &file);
41
42 private:
43   void InitBoundary();
44 };
45
46 } // namespace netdem
```

8.103 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/input/command.hpp File Reference

```
#include "simulation.hpp"
#include <nlohmann/json.hpp>
```

Classes

- class netdem::Command

Namespaces

- namespace netdem

8.104 command.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "simulation.hpp"
4 #include <nlohmann/json.hpp>
5
6 namespace netdem {
7
12 class Command {
13 public:
14   nlohmann::json info;
15   Simulation *sim;
16
17   Command(nlohmann::json const &info, Simulation *sim) : info(info), sim(sim) {}
18
19   virtual void Execute() = 0;
20   virtual ~Command() {}
21 };
22
23 } // namespace netdem
```

8.105 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/input/command_create.cpp File Reference

```
#include "command_create.hpp"
#include <string>
```

Namespaces

- namespace netdem

8.106 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/input/command_create.hpp File Reference

```
#include "command.hpp"
#include "shape_sphere.hpp"
```

Classes

- class netdem::CommandCreate

Namespaces

- namespace netdem

8.107 command_create.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "command.hpp"
4 #include "shape_sphere.hpp"
5
6 namespace netdem {
7
12 class CommandCreate : public Command {
13 public:
14   CommandCreate(nlohmann::json const &info, Simulation *sim);
15
16   void Execute() override;
17 };
18
19 } // namespace netdem
```

8.108 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/input/input_processor.cpp File Reference

```
#include "input_processor.hpp"
#include "command_create.hpp"
#include <fstream>
#include <iostream>
#include <string>
```

Namespaces

- namespace netdem

Typedefs

- using json = nlohmann::json

8.108.1 Typedef Documentation

8.108.1.1 json

```
using json = nlohmann::json
```

8.109 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/input/input_processor.hpp File Reference

```
#include <nlohmann/json.hpp>
```

Classes

- class netdem::InputProcessor

Namespaces

- namespace netdem

8.110 input_processor.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include <nlohmann/json.hpp>
4
5 namespace netdem {
6
7 class Simulation;
8
12 class InputProcessor {
13 public:
14   InputProcessor();
15
16   void Init(Simulation *s);
17
18   void ProcessJsonFile(std::string const &filename);
19   void ProcessJson(nlohmann::json const &js);
20
21 private:
22   Simulation *sim{nullptr};
23 };
24
25 } // namespace netdem
```

8.111 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/main.cpp File Reference

```
#include "netdem.hpp"
#include <fstream>
#include <iostream>
```

Functions

- int main (int argc, char *argv[ ])

8.111.1 Function Documentation

8.111.1.1 main()

```
int main (
            int argc,
            char * argv[ ] )
```

main entry of the netdem app, (prospective) usage format:

- serial run: netdem [input_file.json]
- or parallel run: mpirun -np 2 netdem [input_file.json] to do:
  1. currently only work for serial run, will need to implement and invoke MPI
  2. will need to implement the serialization and de-serialization based on json

8.112 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/general_net.cpp File Reference

```
#include "general_net.hpp"
#include <ensmallen.hpp>
#include <mlpack/methods/ann/layer/layer.hpp>
```

Namespaces

- namespace netdem

8.113 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/general_net.hpp File Reference

```
#include "mlpack_utils.hpp"
#include <mlpack/methods/ann/ffn.hpp>
#include <cstdarg>
#include <string>
```

Classes

- class netdem::GeneralNet

Namespaces

- namespace netdem

8.114 general_net.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 // netdem
4 #include "mlpack_utils.hpp"
5
6 // mlpack
7 #include <mlpack/methods/ann/ffn.hpp>
8
9 // std
10 #include <cstdarg>
11 #include <string>
12
13 namespace netdem {
14
15 class GeneralNet {
16 public:
17   mlpack::ann::FFN<> model;
18
```

```
19   double step_size{0.01};
20   int batch_size{32};
21   double decay_rate_moment{0.9};
22   double decay_rate_norm{0.9};
23   double gradient_init_param{1e-8};
24   int epochs{100};
25   double stop_tol{1e-8};
26
27   void ResetModel();
28
29   void AddLayer(LayerName layer_name, ...);
30
31   void Train(const arma::mat &data_x, const arma::mat &data_y);
32   arma::mat Predict(const arma::mat &data_x);
33
34   arma::mat Classify(const arma::mat &data_x);
35   arma::mat Regress(const arma::mat &data_x);
36
37   void Load(std::string const &filename, std::string const &label);
38   void Save(std::string const &filename, std::string const &label);
39 };
40
41 } // namespace netdem
```

8.115 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mlpack/mlpack_utils.cpp File Reference

```
#include "mlpack_utils.hpp"
#include <cmath>
```

Namespaces

- namespace netdem

Functions

- double netdem::GetMSE (const arma::mat &pred, const arma::mat &Y)
- double netdem::GetMAE (const arma::mat &pred, const arma::mat &Y)
- arma::mat netdem::GetLabels (const arma::mat &ann_outputs)

8.116 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mlpack/mlpack_utils.hpp File Reference

```
#include <mlpack/core.hpp>
```

Namespaces

- namespace netdem

Enumerations

- enum class netdem::LayerName {
  netdem::IdentityLayer , netdem::LayerNorm , netdem::Linear , netdem::ReLU ,
  netdem::LeakyReLU , netdem::FlexibleReLU , netdem::ELU , netdem::Softmax ,
  netdem::LogSoftMax , netdem::LSTM }

Functions

- double netdem::GetMSE (const arma::mat &pred, const arma::mat &Y)
- double netdem::GetMAE (const arma::mat &pred, const arma::mat &Y)
- arma::mat netdem::GetLabels (const arma::mat &ann_outputs)

8.117 mlpack_utils.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include <mlpack/core.hpp>
4
5 namespace netdem {
6
7 enum class LayerName {
8   IdentityLayer,
9   LayerNorm,
10   Linear,
11   ReLU,
12   LeakyReLU,
13   FlexibleReLU,
14   ELU,
15   Softmax,
16   LogSoftMax,
17   LSTM
18 };
19
20 double GetMSE(const arma::mat &pred, const arma::mat &Y);
21
22 double GetMAE(const arma::mat &pred, const arma::mat &Y);
23
24 arma::mat GetLabels(const arma::mat &ann_outputs);
25
26 } // namespace netdem
```

8.118 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mlpack/regression_net.cpp File Reference

```
#include "regression_net.hpp"
#include <ensmallen.hpp>
#include <mlpack/core/data/scaler_methods/min_max_scaler.hpp>
#include <mlpack/methods/ann/ffn.hpp>
#include <mlpack/methods/ann/layer/layer.hpp>
```

Namespaces

- namespace netdem

8.119 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mlpack/regression_net.hpp File Reference

```
#include "mlpack_utils.hpp"
#include <mlpack/methods/ann/ffn.hpp>
#include <mlpack/methods/ann/init_rules/he_init.hpp>
#include <mlpack/methods/ann/loss_functions/mean_squared_error.hpp>
#include <cstdarg>
#include <string>
```

Classes

- class netdem::RegressionNet

Namespaces

- namespace netdem

8.120 regression_net.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 // netdem
4 #include "mlpack_utils.hpp"
5
6 // mlpack
7 #include <mlpack/methods/ann/ffn.hpp>
8 #include <mlpack/methods/ann/init_rules/he_init.hpp>
9 #include <mlpack/methods/ann/loss_functions/mean_squared_error.hpp>
10
11 // std
12 #include <cstdarg>
13 #include <string>
14
15 namespace netdem {
16
17 class RegressionNet {
18 public:
19   mlpack::ann::FFN<mlpack::ann::MeanSquaredError<>,
20                    mlpack::ann::HeInitialization>
21       model;
22
23   double step_size{0.01};
24   int batch_size{32};
25   double decay_rate_moment{0.9};
26   double decay_rate_norm{0.9};
27   double gradient_init_param{1e-8};
28   int epochs{100};
29   double stop_tol{1e-8};
30
31   void ResetModel();
32
33   void AddLayer(LayerName layer_name, ...);
34
35   void Train(const arma::mat &data_x, const arma::mat &data_y);
36   arma::mat Predict(const arma::mat &data_x);
37
38   void Load(std::string const &filename, std::string const &label);
39   void Save(std::string const &filename, std::string const &label);
40 };
41
42 } // namespace netdem
```

8.121 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/solver_ann_pp.cpp File Reference

```
#include "solver_ann_pp.hpp"
#include "utils_math.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.122 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/solver_ann_pp.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pp.hpp"
#include "contact_model_factory.hpp"
#include "general_net.hpp"
#include "particle.hpp"
#include "regression_net.hpp"
#include "shape.hpp"
#include "shape_trimesh.hpp"
#include "wall.hpp"
#include <string>
```

Classes

- class netdem::SolverANNPP

Namespaces

- namespace netdem

8.123 solver_ann_pp.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pp.hpp"
5 #include "contact_model_factory.hpp"
6 #include "general_net.hpp"
7 #include "particle.hpp"
8 #include "regression_net.hpp"
9 #include "shape.hpp"
```

```
10 #include "shape_trimesh.hpp"
11 #include "wall.hpp"
12 #include <string>
13
14 namespace netdem {
15
19 class SolverANNPP : public CollisionSolverPP {
20 public:
21   netdem::GeneralNet classifier;
22   netdem::RegressionNet regressor;
23   bool is_initialized{false};
24
25   SolverANNPP();
26   SolverANNPP(Particle *const p1, Particle *const p2);
27
28   CollisionSolverPP *Clone() const override;
29
30   void Init(std::string const &classifier_file,
31             std::string const &regressor_file);
32
33   void Init(Particle *const p1, Particle *const p2) override;
34
35   bool Detect() override;
36
37   void ResolveInit(ContactPP *const cnt, double timestep) override;
38   void ResolveUpdate(ContactPP *const cnt, double timestep) override;
39
40   void ResolveInit_LinearSpring(CollisionGeometries *const cnt_geoms,
41                                 double timestep);
42   void ResolveUpdate_LinearSpring(CollisionGeometries *const cnt_geoms,
43                                   double timestep);
44
45   void ResolveInit_VolumeBased(CollisionGeometries *const cnt_geoms,
46                                double timestep);
47   void ResolveUpdate_VolumeBased(CollisionGeometries *const cnt_geoms,
48                                  double timestep);
49
50   void ResolveInit_PotentialBased(CollisionGeometries *const cnt_geoms,
51                                   double timestep);
52   void ResolveUpdate_PotentialBased(CollisionGeometries *const cnt_geoms,
53                                     double timestep);
54
55   // return: potential and cnt_pos
56   std::tuple<double, Vec3d> Potential(Vec3d const &pos, Vec4d const &quat);
57
58   // return: force, moment and pos
59   std::tuple<Vec3d, Vec3d, Vec3d> EvaluateContactForces();
60
61 private:
62   Shape *shape_1{nullptr}, *shape_2{nullptr};
63
64   double bound_sphere_radius_1, bound_sphere_radius_2, scale;
65
66   Vec3d dpos_12, dpos_12_ref;
67   Vec4d dquat_12;
68 };
69
70 } // namespace netdem
```

8.124 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mlpack/solver_ann_↩ pplane.cpp File Reference

```
#include "solver_ann_pplane.hpp"
#include "utils_math.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.125 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mlpack/solver_ann_pplane.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pw.hpp"
#include "contact_model_factory.hpp"
#include "general_net.hpp"
#include "particle.hpp"
#include "regression_net.hpp"
#include "shape.hpp"
#include "shape_plane.hpp"
#include "wall.hpp"
#include <string>
```

Classes

- class netdem::SolverANNPPlane

Namespaces

- namespace netdem

8.126 solver_ann_pplane.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pw.hpp"
5 #include "contact_model_factory.hpp"
6 #include "general_net.hpp"
7 #include "particle.hpp"
8 #include "regression_net.hpp"
9 #include "shape.hpp"
10 #include "shape_plane.hpp"
11 #include "wall.hpp"
12 #include <string>
13
14 namespace netdem {
15
19 class SolverANNPPlane : public CollisionSolverPW {
20 public:
21   netdem::GeneralNet classifier;
22   netdem::RegressionNet regressor;
23   bool is_initialized{false};
24
25   SolverANNPPlane();
26   SolverANNPPlane(Particle *const p, Wall *const w);
27
28   CollisionSolverPW *Clone() const override;
29
30   void Init(std::string const &classifier_file,
31             std::string const &regressor_file);
32
33   void Init(Particle *const p, Wall *const w) override;
34
35   bool Detect() override;
36
37   void ResolveInit(ContactPW *const cnt, double timestep) override;
```

```
38   void ResolveUpdate(ContactPW *const cnt, double timestep) override;
39
40   void ResolveInit_LinearSpring(CollisionGeometries *const cnt_geoms,
41                                 double timestep);
42   void ResolveUpdate_LinearSpring(CollisionGeometries *const cnt_geoms,
43                                   double timestep);
44
45   void ResolveInit_VolumeBased(CollisionGeometries *const cnt_geoms,
46                                double timestep);
47   void ResolveUpdate_VolumeBased(CollisionGeometries *const cnt_geoms,
48                                  double timestep);
49
50   void ResolveInit_PotentialBased(CollisionGeometries *const cnt_geoms,
51                                   double timestep);
52   void ResolveUpdate_PotentialBased(CollisionGeometries *const cnt_geoms,
53                                     double timestep);
54
55   // return: potential and cnt_pos
56   std::tuple<double, Vec3d> Potential(double dist, Vec3d const &nn);
57
58   // return: force, moment and pos
59   std::tuple<double, Vec3d, Vec3d> EvaluateContactForces();
60
61 private:
62   double bound_sphere_radius_1, dist_pc_to_plane, scale;
63   Vec3d dir_n, dir_n_ref;
64 };
65
66 } // namespace netdem
```

8.127 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mlpack/solver_ann_pw.cpp File Reference

```
#include "solver_ann_pw.hpp"
#include "utils_math.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.128 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mlpack/solver_ann_pw.hpp File Reference

```
#include "collision_geometries.hpp"
#include "collision_solver_pw.hpp"
#include "contact_model_factory.hpp"
#include "general_net.hpp"
#include "particle.hpp"
#include "regression_net.hpp"
#include "shape.hpp"
#include "wall.hpp"
#include <string>
```

Classes

- class netdem::SolverANNPW

Namespaces

- namespace netdem

8.129 solver_ann_pw.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_geometries.hpp"
4 #include "collision_solver_pw.hpp"
5 #include "contact_model_factory.hpp"
6 #include "general_net.hpp"
7 #include "particle.hpp"
8 #include "regression_net.hpp"
9 #include "shape.hpp"
10 #include "wall.hpp"
11 #include <string>
12
13 namespace netdem {
14
18 class SolverANNPW : public CollisionSolverPW {
19 public:
20   netdem::GeneralNet classifier;
21   netdem::RegressionNet regressor;
22   bool is_initialized{false};
23
24   SolverANNPW();
25   SolverANNPW(Particle *const p, Wall *const w);
26
27   CollisionSolverPW *Clone() const override;
28
29   void Init(std::string const &classifier_file,
30             std::string const &regressor_file);
31
32   void Init(Particle *const p, Wall *const w) override;
33
34   bool Detect() override;
35
36   void ResolveInit(ContactPW *const cnt, double timestep) override;
37   void ResolveUpdate(ContactPW *const cnt, double timestep) override;
38
39   void ResolveInit_LinearSpring(CollisionGeometries *const cnt_geoms,
40                                 double timestep);
41   void ResolveUpdate_LinearSpring(CollisionGeometries *const cnt_geoms,
42                                   double timestep);
43
44   void ResolveInit_VolumeBased(CollisionGeometries *const cnt_geoms,
45                                double timestep);
46   void ResolveUpdate_VolumeBased(CollisionGeometries *const cnt_geoms,
47                                  double timestep);
48
49   void ResolveInit_PotentialBased(CollisionGeometries *const cnt_geoms,
50                                   double timestep);
51   void ResolveUpdate_PotentialBased(CollisionGeometries *const cnt_geoms,
52                                     double timestep);
53
54   // return: potential and cnt_pos
55   std::tuple<double, Vec3d> Potential(Vec3d const &pos, Vec4d const &quat);
56
57   // return: force, moment and pos
58   std::tuple<Vec3d, Vec3d, Vec3d> EvaluateContactForces();
59
60 private:
61   Shape *shape_1{nullptr}, *shape_2{nullptr};
62
63   double bound_sphere_radius_1, bound_sphere_radius_2, scale;
64
65   Vec3d dpos_12, dpos_12_ref;
66   Vec4d dquat_12;
67 };
68
69 } // namespace netdem
```

8.130 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/body_force.cpp File Reference

```
#include "body_force.hpp"
#include "simulation.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.131 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/body_force.hpp File Reference

```
#include "modifier.hpp"
#include "particle.hpp"
#include <cstdarg>
```

Classes

- class netdem::BodyForce

Namespaces

- namespace netdem

8.132 body_force.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "modifier.hpp"
4 #include "particle.hpp"
5 #include <cstdarg>
6
7 namespace netdem {
8
18 class BodyForce : public Modifier {
19 public:
20   VecXT<int> particle_id_list;
21   VecXT<Particle *> particle_list;
22   Vec3d unit_force{0, 0, 0};
23
24   bool use_particles_in_scene{false};
25
26   BodyForce(Vec3d const &b);
27   BodyForce(double b_x, double b_y, double b_z);
28
29   void SetParticlesFromScene();
```

```
30
31   void SetParticles(const VecXT<int> &id_list);
32   void SetParticles(int num_ids, ...);
33
34   Modifier *Clone() const override;
35
36   void Execute() override;
37   void Update() override;
38 };
39
40 } // namespace netdem
```

8.133 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/modifier/breakage_analysis_↩ pd.cpp File Reference

```
#include "breakage_analysis_pd.hpp"
#include "eigen_wrapper.hpp"
#include "peridigm_dem_coupler.hpp"
#include "simulation.hpp"
```

Namespaces

- namespace netdem

8.134 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/modifier/breakage_analysis_↩ pd.hpp File Reference

```
#include "modifier.hpp"
#include "peridigm_dem_coupler.hpp"
```

Classes

- class netdem::BreakageAnalysisPD

Namespaces

- namespace netdem

8.135 breakage_analysis_pd.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "modifier.hpp"
4 #include "peridigm_dem_coupler.hpp"
5
6 namespace netdem {
7
8 class BreakageAnalysisPD : public Modifier {
9 public:
10   VecXT<int> particle_id_list;
11   VecXT<Particle *> particle_list;
12
13   bool use_particles_in_scene{false};
14
15   PeriDigmDEMCoupler pd_dem_coupler;
16
17   BreakageAnalysisPD();
18
19   void SetRootPath(std::string const &root_path);
20   void SetFrequency(bool save_by_cycles, double interval);
21
22   void SetParticlesFromScene();
23
24   void SetParticles(const VecXT<int> &id_list);
25   void SetParticles(const std::initializer_list<int> &id_list);
26
27   Modifier *Clone() const override;
28
29   void Init(Simulation *sim) override;
30
31   void Execute() override;
32
33   void Update();
34
35 private:
36   std::string root_path{"tmp/out/"};
37   bool excute_by_cycles{true};
38   int cycle_interval{0}, cycle_previous{0};
39   double time_interval{0}, time_previous{0};
40
41   bool CheckIfToExecute();
42 };
43
44 } // namespace netdem
```

8.136 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/data_dumper.cpp File Reference

```
#include "data_dumper.hpp"
#include "simulation.hpp"
#include <filesystem>
#include <fstream>
#include <iostream>
#include <nlohmann/json.hpp>
#include <sstream>
#include <string>
```

Namespaces

- namespace netdem

Typedefs

- using json = nlohmann::json

8.136.1 Typedef Documentation

8.136.1.1 json

```
using json = nlohmann::json
```

8.137 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/data_dumper.hpp File Reference

```
#include "modifier.hpp"
#include "scene.hpp"
```

Classes

- class netdem::DataDumper

Namespaces

- namespace netdem

8.138 data_dumper.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "modifier.hpp"
4 #include "scene.hpp"
5
6 namespace netdem {
7
28 class DataDumper : public Modifier {
29 public:
30   bool dump_particle_info{true}, dump_wall_info{false},
31       dump_contact_info{false}, dump_shape_info{false};
32
33   DataDumper();
34
35   void Init(Simulation *sim) override;
36
37   void SetRootPath(std::string const &root_path);
38   void SetDataType(std::string const &data_type);
39   void SetFrequency(bool save_by_cycles, double interval);
40
41   void SaveParticleInfoAsVTK();
42   void SaveParticleInfoAsDump();
```

```
43
44   void SaveParticleInfoAsVTKWithProxy(); // for mpi debug use
45
46   void SaveWallInfoAsVTK();
47   void SaveWallInfoAsDump();
48
49   void SaveCollisionInfoAsVTK();
50   void SaveCollisionInfoAsDump();
51
52   void SaveBondInfoAsVTK();
53   void SaveBondInfoAsDump();
54
55   void SaveShapeInfoAsSTL();
56   void SaveShapeInfoAsVTK();
57   void SaveShapeInfoAsJson(bool all_in_one = false);
58
59   Modifier *Clone() const override;
60
61   void Execute() override;
62
63 private:
64   std::string root_path{"tmp/out/"}, data_type{"vtk"};
65   bool save_by_cycles{true};
66   int cycle_interval{0}, cycle_previous{0};
67   double time_interval{0}, time_previous{0};
68
69   void GetCollisionContacts(VecXT<ContactPP *> *const cnt_pp_list,
70                             VecXT<ContactPW *> *const cnt_pw_list);
71   void GetBondContacts(VecXT<ContactPP *> *const cnt_pp_list,
72                        VecXT<ContactPW *> *const cnt_pw_list);
73
74   inline std::string GetParticleInfoFilename();
75   inline std::string GetWallInfoFilename();
76
77   inline std::string GetCollisionInfoFilename();
78   inline std::string GetBondInfoFilename();
79
80   inline std::string GetShapeInfoFilename();
81
82   inline bool CheckIfToSave();
83 };
84
85 } // namespace netdem
```

8.139 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/modifier/deformation_↩ analysis.cpp File Reference

```
#include "deformation_analysis.hpp"
#include "simulation.hpp"
```

Namespaces

- namespace netdem

8.140 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/modifier/deformation_↩ analysis.hpp File Reference

```
#include "deformable_particle.hpp"
#include "modifier.hpp"
#include <unordered_map>
```

Classes

- class netdem::DeformationAnalysis
- class netdem::DeformationAnalysis::Settings

Namespaces

- namespace netdem

8.141 deformation_analysis.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "deformable_particle.hpp"
4 #include "modifier.hpp"
5 #include <unordered_map>
6
7 namespace netdem {
8
9 class DeformationAnalysis : public Modifier {
10 public:
11   class Settings {
12   public:
13     enum SolveBy { cycles, time };
14
15     double neo_k{6.94e5}, neo_mu{5.21e5}, density{500.0};
16     double damp_coef{0.7};
17
18     Vec3d gravity_coef{0.0, 0.0, -9.81};
19     double timestep{1.0e-4};
20     int mesh_res{20};
21
22     bool save_by_cycles{false};
23     bool save_by_time{false};
24
25     int save_cycle_interval{0};
26     double save_time_interval{0};
27
28     std::string root_path{"tmp/out/deformation_analysis/"};
29
30     // 0: cycles, 1: time
31     SolveBy solve_by{SolveBy::cycles};
32
33     int solve_cycle_interval{0};
34     double solve_time_interval{0};
35   };
36
37   Settings settings;
38
39   VecXT<int> particle_id_list;
40
41   // reference of the particles in the dem simulation
42   std::unordered_map<Particle *, std::pair<bool, DeformableParticle»
43       particle_map;
44
45   DeformationAnalysis();
46
47   void SetParticlesFromScene();
48
49   void SetParticles(const VecXT<int> &id_list);
50   void SetParticles(const std::initializer_list<int> &id_list);
51
52   Modifier *Clone() const override;
53
54   void Init(Simulation *sim) override;
55
56   void Execute() override;
57
58   void Update();
59
60 private:
61   void SolveDeformation();
62   void SaveFEMAsVTK();
63
64   void EvaluateBCForce(DeformableParticle *const p_deformable_ptr,
```

```
65                       ContactPP *const cnt);
66   void EvaluateBCForce(DeformableParticle *const p_deformable_ptr,
67                       ContactPW *const cnt);
68
69   void SetSettings(FEMSimulator *const fem_sim);
70
71   bool use_particles_in_scene{false};
72
73   int solve_cycle_previous{0};
74   double solve_time_previous{0};
75
76   int save_cycle_previous{0};
77   double save_time_previous{0};
78
79   std::string GetFEMResultFileName(Particle *const p_ptr);
80
81   bool CheckIfToExecute();
82 };
83
84 } // namespace netdem
```

8.142 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/modifier/gravity.cpp File Reference

```
#include "gravity.hpp"
#include "simulation.hpp"
```

Namespaces

- namespace netdem

8.143 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/modifier/gravity.hpp File Reference

```
#include "modifier.hpp"
```

Classes

- class netdem::Gravity

Namespaces

- namespace netdem

8.144 gravity.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "modifier.hpp"
4
5 namespace netdem {
6
7 class Gravity : public Modifier {
8 public:
9   Gravity();
10
11   Modifier *Clone() const override;
12
13   void Init(Simulation *sim) override;
14
15   void Execute() override;
16 };
17
18 } // namespace netdem
```

8.145 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/membrane_wall.cpp File Reference

```
#include "membrane_wall.hpp"
#include "shape_triangle.hpp"
#include "simulation.hpp"
```

8.146 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/membrane_wall.hpp File Reference

```
#include "membrane.hpp"
#include "modifier.hpp"
#include "wall.hpp"
```

Classes

- class netdem::MembraneWall

Namespaces

- namespace netdem

8.147 membrane_wall.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "membrane.hpp"
4 #include "modifier.hpp"
5 #include "wall.hpp"
6
7 namespace netdem {
8
9 class Simulation;
10
11 class MembraneWall : public Modifier, public Membrane {
12 public:
13   bool enable_deformation{false}, dump_info{true};
14
15   VecXT<Wall *> wall_list;
16
17   MembraneWall();
18
19   MembraneWall(double radius, double height);
20
21   MembraneWall(double radius, double height, double mesh_size);
22
23   MembraneWall(double radius, double height, double mesh_size, double center_x,
24                double center_y, double center_z);
25
26   void SetRootPath(std::string const &root_path);
27   void SetDataType(std::string const &data_type);
28   void SetFrequency(bool save_by_cycles, double interval);
29
30   void SetDimensions(double r, double h);
31
32   Modifier *Clone() const override;
33
34   void Init(Simulation *sim) override;
35
36   void SetPressure(double pressure);
37
38   void Execute() override;
39
40 private:
41   std::string root_path{"tmp/out/"}, data_type{"vtk"};
42   bool save_by_cycles{true};
43   int cycle_interval{0}, cycle_previous{0};
44   double time_interval{0}, time_previous{0};
45
46   void UpdateBCForceFromDEM();
47
48   std::string GetFilename();
49   bool CheckIfToSave();
50 };
51
52 } // namespace netdem
```

8.148 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/modifier/modifier.cpp File Reference

```
#include "modifier.hpp"
#include "modifier_manager.hpp"
#include "simulation.hpp"
```

Namespaces

- namespace netdem

8.149 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/modifier.hpp File Reference

```
#include "dem_solver.hpp"
#include <string>
```

Classes

- class netdem::Modifier

Namespaces

- namespace netdem

8.150 modifier.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "dem_solver.hpp"
4 #include <string>
5
6 namespace netdem {
7
8 class Simulation;
9 class Scene;
10
17 class Modifier {
18 public:
19   std::string label{"default"};
20   DEMSolver::CyclePoint cycle_point{DEMSolver::CyclePoint::pre};
21
22   Simulation *sim{nullptr};
23   Scene *scene{nullptr};
24
25   bool update_with_scene{false};
26
27   Modifier();
28
29   virtual Modifier *Clone() const;
30
31   virtual void Init(Simulation *sim);
32
33   virtual void Enable();
34
35   virtual void Execute();
36   virtual void Update();
37
38   virtual ~Modifier();
39 };
40
41 } // namespace netdem
```

8.151 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/modifier_manager.cpp File Reference

```
#include "modifier_manager.hpp"
#include "simulation.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.152 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/modifier/modifier_↩ manager.hpp File Reference

```
#include "modifier.hpp"
#include <memory>
#include <string>
#include <unordered_map>
#include <unordered_set>
```

Classes

- class netdem::ModifierManager

Namespaces

- namespace netdem

8.153 modifier_manager.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "modifier.hpp"
4 #include <memory>
5 #include <string>
6 #include <unordered_map>
7 #include <unordered_set>
8
9 namespace netdem {
10
11 class Simulation;
12
28 class ModifierManager {
29 public:
30   // object maps
31   std::unordered_map<std::string, Modifier *> modifier_lib;
32
33   // active modifiers
34   VecXT<std::unordered_set<Modifier *>> modifier_list;
35
36   // subscribers
37   std::unordered_set<Modifier *> scene_state_subscribers;
38
39   ModifierManager();
40
41   void Init(Simulation *s);
42
44   Modifier *Insert(Modifier *e);
45   void RemoveModifier(std::string const &label);
46
48   void Enable(std::string const &label);
49   void Disable(std::string const &label);
50
```

```
51   void Enable(Modifier *const e);
52   void Disable(Modifier *const e);
53
55   Modifier *FindModifier(std::string const &label);
56   bool FindModifier(Modifier *const e);
57
59   void UpdateModifiers();
60
62   void ExecuteModifiers(DEMSolver::CyclePoint cycle_point);
63
64   ~ModifierManager();
65
66 private:
67   Simulation *sim{nullptr};
68 };
69
70 } // namespace netdem
```

8.154 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/particle_energy_cal.cpp File Reference

```
#include "particle_energy_cal.hpp"
#include "simulation.hpp"
#include <cstdarg>
#include <iostream>
```

Namespaces

- namespace netdem

8.155 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/particle_energy_cal.hpp File Reference

```
#include "modifier.hpp"
#include "particle.hpp"
#include <cstdarg>
```

Classes

- struct netdem::ParticleEnergy
- class netdem::ParticleEnergyCalculator

Namespaces

- namespace netdem

8.156 particle_energy_cal.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "modifier.hpp"
4 #include "particle.hpp"
5 #include <cstdarg>
6
7 namespace netdem {
8
9 struct ParticleEnergy {
10   double total{0}, kinetic{0}, gravitational{0}, translational{0},
11       rotational{0};
12 };
13
17 class ParticleEnergyCalculator : public Modifier {
18 public:
19   ParticleEnergyCalculator();
20
21   VecXT<int> particle_id_list;
22   VecXT<Particle *> particle_list;
23   VecXT<ParticleEnergy> particle_energy_list;
24
25   bool use_particles_in_scene{false};
26
27   void SetParticlesFromScene();
28   void SetParticles(const VecXT<int> &id_list);
29   void SetParticles(int num_ids, ...);
30
31   ParticleEnergy GetEnergy();
32   ParticleEnergy GetEnergy(Particle *const p);
33
34   Modifier *Clone() const override;
35
36   void Execute() override;
37
38   void Execute(const VecXT<Particle *> &p_list);
39
40   void Update() override;
41 };
42
43 } // namespace netdem
```

8.157 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/wall_disp_control.cpp File Reference

```
#include "wall_disp_control.hpp"
#include "simulation.hpp"
#include <cstdarg>
#include <iostream>
```

Namespaces

- namespace netdem

8.158 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/wall_disp_control.hpp File Reference

```
#include "modifier.hpp"
#include "wall.hpp"
#include <cstdarg>
```

Classes

- class netdem::WallDispControl

Namespaces

- namespace netdem

8.159 wall_disp_control.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "modifier.hpp"
4 #include "wall.hpp"
5 #include <cstdarg>
6
7 namespace netdem {
8
9 class Simulation;
10
11 class WallDispControl : public Modifier {
12 public:
13   VecXT<int> wall_id_list;
14   VecXT<Wall *> wall_list;
15   Vec3d vel{0, 0, 0}, spin{0, 0, 0};
16
17   WallDispControl();
18
19   void SetVelocity(double v_x, double v_y, double v_z);
20   void SetSpin(double spin_x, double spin_y, double spin_z);
21
22   void SetWalls(const VecXT<int> &id_list);
23   void SetWalls(const std::initializer_list<int> &id_list);
24
25   Modifier *Clone() const override;
26
27   void Execute() override;
28   void Update() override;
29 };
30
31 } // namespace netdem
```

8.160 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/wall_servo_control.cpp File Reference

```
#include "wall_servo_control.hpp"
#include "igl_wrapper.hpp"
#include "shape_plane.hpp"
#include "simulation.hpp"
#include <iostream>
```

Namespaces

- namespace netdem

8.161 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/modifier/wall_servo_control.hpp File Reference

```
#include "modifier.hpp"
#include "wall.hpp"
#include <cstdarg>
```

Classes

- class netdem::WallServoControl

Namespaces

- namespace netdem

8.162 wall_servo_control.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "modifier.hpp"
4 #include "wall.hpp"
5 #include <cstdarg>
6
7 namespace netdem {
8
9 class Simulation;
10
11 class WallServoControl : public Modifier {
12 public:
```

```
13   VecXT<int> wall_id_list;
14   VecXT<Wall *> wall_list;
15   VecXT<double> pressure_list;
16   double kn{1.0e5}, target_pressure{0.0}, vel_max{0.5}, study_rate{0.5};
17
18   double tol{0.05};
19   bool achieved{true}, enable_warning{false};
20
21   WallServoControl(double kn);
22
23   void SetWalls(const VecXT<int> &id_list);
24   void SetWalls(const std::initializer_list<int> &id_list);
25
26   void AddWall(int id);
27
28   Modifier *Clone() const override;
29
30   void Execute() override;
31   void Update() override;
32 };
33
34 } // namespace netdem
```

8.163 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mpi/bond_entry_data.hpp File Reference

```
#include <string>
```

Classes

- struct netdem::BondEntryData

Namespaces

- namespace netdem

8.164 bond_entry_data.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include <string>
4
5 namespace netdem {
6
7 struct BondEntryData {
8   double pos[3]{0, 0, 0};
9   double dir_n[3]{1, 0, 0}, dir_s[3]{0, 1, 0}, dir_t[3]{0, 0, 1};
10   double branch_1[3]{1, 0, 0}, branch_2[3]{1, 0, 0};
11
12   double pos_ini[3]{0, 0, 0};
13   double dir_n_ini[3]{1, 0, 0}, dir_s_ini[3]{0, 1, 0}, dir_t_ini[3]{0, 0, 1};
14
15   double pos_1_ini[3]{0, 0, 0}, pos_2_ini[3]{0, 0, 0};
16   double quat_1_ini[4]{1, 0, 0, 0}, quat_2_ini[4]{1, 0, 0, 0};
17
18   double radius{0};
19
20   double fc_n{0}, fc_s{0}, fc_t{0};
21   double mc_n{0}, mc_s{0}, mc_t{0};
22
23   double fd_n{0}, fd_s{0}, fd_t{0};
24   double md_n{0}, md_s{0}, md_t{0};
25
26   int cnt_model_id{-1};
27 };
28
29 } // namespace netdem
```

8.165 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/bond_entry_parser.cpp File Reference

```
#include "bond_entry_parser.hpp"
#include "utils_io.hpp"
```

8.166 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/bond_entry_parser.hpp File Reference

```
#include "bond_entry.hpp"
#include "bond_entry_data.hpp"
#include "mini_map.hpp"
#include <mpi.h>
```

Classes

- class netdem::BondEntryParser

Namespaces

- namespace netdem

8.167 bond_entry_parser.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_entry.hpp"
4 #include "bond_entry_data.hpp"
5 #include "mini_map.hpp"
6 #include <mpi.h>
7
8 namespace netdem {
9
10 class BondEntryParser {
11 public:
12   static void ClassToStruct(const BondEntry *const entry_class,
13                             BondEntryData *const entry_struct);
14
15   static void
16   StructToClass(BondEntry *const entry_class,
17                 const BondEntryData *const entry_struct,
18                 const MiniMap<int, ContactModel *> &contact_model_map);
19
20   static void DefineMPIDataType(MPI_Datatype *const datatype);
21 };
22
23 } // namespace netdem
```

8.168 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/collision_entry_data.hpp File Reference

```
#include <string>
```

Classes

- struct netdem::CollisionEntryData

Namespaces

- namespace netdem

8.169 collision_entry_data.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include <string>
4
5 namespace netdem {
6
7 struct CollisionEntryData {
8   double pos[3]{0, 0, 0};
9   double dir_n[3]{1, 0, 0}, dir_s[3]{0, 1, 0}, dir_t[3]{0, 0, 1};
10   double branch_1[3]{1, 0, 0}, branch_2[3]{-1, 0, 0};
11
12   int node_id{0};
13
14   double fc_n{0}, fc_s{0}, fc_t{0};
15   double mc_n{0}, mc_s{0}, mc_t{0};
16
17   double fd_n{0}, fd_s{0}, fd_t{0};
18   double md_n{0}, md_s{0}, md_t{0};
19
20   int cnt_model_id{-1};
21 };
22
23 } // namespace netdem
```

8.170 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/collision_entry_parser.cpp File Reference

```
#include "collision_entry_parser.hpp"
#include "utils_io.hpp"
```

8.171 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/collision_entry_parser.hpp File Reference

```
#include "collision_entry.hpp"
#include "collision_entry_data.hpp"
#include "mini_map.hpp"
#include <mpi.h>
```

Classes

- class netdem::CollisionEntryParser

Namespaces

- namespace netdem

8.172 collision_entry_parser.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "collision_entry.hpp"
4 #include "collision_entry_data.hpp"
5 #include "mini_map.hpp"
6 #include <mpi.h>
7
8 namespace netdem {
9
10 class CollisionEntryParser {
11 public:
12   static void ClassToStruct(const CollisionEntry *const entry_class,
13                             CollisionEntryData *const entry_struct);
14
15   static void
16   StructToClass(CollisionEntry *const entry_class,
17                 const CollisionEntryData *const entry_struct,
18                 const MiniMap<int, ContactModel *> &contact_model_map);
19
20   static void DefineMPIDataType(MPI_Datatype *const datatype);
21 };
22
23 } // namespace netdem
```

8.173 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pp_data.hpp File Reference

Classes

- struct netdem::ContactPPData

Namespaces

- namespace netdem

8.174 contact_pp_data.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 namespace netdem {
4
8 struct ContactPPData {
9   int particle_1_id{0}, particle_2_id{0};
10
11   int bond_model_id{-1}, collision_model_id{-1};
12
13   int num_bond_entries{0}, num_collision_entries{0};
14 };
15
16 } // namespace netdem
```

8.175 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mpi/contact_pp_parser.cpp File Reference

```
#include "contact_pp_parser.hpp"
#include "utils_io.hpp"
```

8.176 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mpi/contact_pp_parser.hpp File Reference

```
#include "bond_entry_data.hpp"
#include "bond_entry_parser.hpp"
#include "collision_entry_data.hpp"
#include "collision_entry_parser.hpp"
#include "contact_pp.hpp"
#include "contact_pp_data.hpp"
```

Classes

- class netdem::ContactPPParser

Namespaces

- namespace netdem

8.177 contact_pp_parser.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_entry_data.hpp"
4 #include "bond_entry_parser.hpp"
5 #include "collision_entry_data.hpp"
6 #include "collision_entry_parser.hpp"
7 #include "contact_pp.hpp"
8 #include "contact_pp_data.hpp"
9
10 namespace netdem {
11
15 class ContactPPParser {
16 public:
17   static void ClassToStruct(const ContactPP *const cnt_class,
18                             ContactPPData *const cnt_struct);
19
20   static void
21   StructToClass(ContactPP *const cnt_class,
22                 const ContactPPData *const cnt_struct,
23                 const BondEntryData *const bond_entries_data,
24                 const CollisionEntryData *const collision_entries_data,
25                 const MiniMap<int, ContactModel *> &contact_model_map);
26
27   static void DefineMPIDataType(MPI_Datatype *const datatype);
28 };
29
30 } // namespace netdem
```

8.178 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pw_data.hpp File Reference

Classes

- struct netdem::ContactPWData

Namespaces

- namespace netdem

8.179 contact_pw_data.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 namespace netdem {
4
8 struct ContactPWData {
9   int particle_id{0}, wall_id{0};
10
11   int bond_model_id{-1}, collision_model_id{-1};
12
13   int num_bond_entries{0}, num_collision_entries{0};
14 };
15
16 } // namespace netdem
```

8.180 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pw_parser.cpp File Reference

```
#include "contact_pw_parser.hpp"
#include "utils_io.hpp"
```

8.181 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/contact_pw_parser.hpp File Reference

```
#include "bond_entry_data.hpp"
#include "bond_entry_parser.hpp"
#include "collision_entry_data.hpp"
#include "collision_entry_parser.hpp"
#include "contact_pw.hpp"
#include "contact_pw_data.hpp"
```

Classes

- class netdem::ContactPWParser

Namespaces

- namespace netdem

8.182 contact_pw_parser.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_entry_data.hpp"
4 #include "bond_entry_parser.hpp"
5 #include "collision_entry_data.hpp"
6 #include "collision_entry_parser.hpp"
7 #include "contact_pw.hpp"
8 #include "contact_pw_data.hpp"
9
10 namespace netdem {
11
15 class ContactPWParser {
16 public:
17   static void ClassToStruct(const ContactPW *const cnt_class,
18                             ContactPWData *const cnt_struct);
19
20   static void
21   StructToClass(ContactPW *const cnt_class,
22                 const ContactPWData *const cnt_struct,
23                 const BondEntryData *const bond_entries_data,
24                 const CollisionEntryData *const collision_entries_data,
25                 const MiniMap<int, ContactModel *> &contact_model_map);
26
27   static void DefineMPIDataType(MPI_Datatype *const datatype);
28 };
29
30 } // namespace netdem
```

8.183 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mpi/mpi_data_def.hpp File Reference

```
#include "bond_entry_parser.hpp"
#include "collision_entry_parser.hpp"
#include "contact_pp_parser.hpp"
#include "contact_pw_parser.hpp"
#include "particle_parser.hpp"
#include <iostream>
#include <mpi.h>
```

Classes

- class netdem::MPIDataDefine

Namespaces

- namespace netdem

8.184 mpi_data_def.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_entry_parser.hpp"
4 #include "collision_entry_parser.hpp"
5 #include "contact_pp_parser.hpp"
6 #include "contact_pw_parser.hpp"
7 #include "particle_parser.hpp"
8 #include <iostream>
9 #include <mpi.h>
10
11 namespace netdem {
12
13 class MPIDataDefine {
14 public:
15   MPI_Datatype particle_datatype, bond_entry_datatype, collision_entry_datatype,
16       contact_pp_datatype, contact_pw_datatype;
17
18   void Init() {
19     ParticleParser::DefineMPIDataType(&particle_datatype);
20     BondEntryParser::DefineMPIDataType(&bond_entry_datatype);
21     CollisionEntryParser::DefineMPIDataType(&collision_entry_datatype);
22     ContactPPParser::DefineMPIDataType(&contact_pp_datatype);
23     ContactPWParser::DefineMPIDataType(&contact_pw_datatype);
24   }
25 };
26
27 } // namespace netdem
```

8.185 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/mpi_manager.cpp File Reference

```
#include "mpi_manager.hpp"
#include "dem_object_pool.hpp"
#include "domain_manager.hpp"
#include "shape_factory.hpp"
#include "simulation.hpp"
#include <unordered_set>
```

Namespaces

- namespace netdem

Typedefs

- using json = nlohmann::json

8.185.1 Typedef Documentation

8.185.1.1 json

```
using json = nlohmann::json
```

8.186 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/mpi_manager.hpp File Reference

```
#include "mpi_data_def.hpp"
#include <list>
#include <mpi.h>
```

Classes

- class netdem::MPIManager

Namespaces

- namespace netdem

8.187 mpi_manager.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "mpi_data_def.hpp"
4 #include <list>
5 #include <mpi.h>
6
7 namespace netdem {
8
9 class Simulation;
10
38 class MPIManager {
39 public:
41   MPIDataDefine mpi_data_def;
42
44   int my_rank, num_procs;
45
48   VecXT<Shape *> shape_transfer_out_list;
49
51   VecXT<VecXT<Particle *>> particle_proxy_out_list;
52
54   VecXT<VecXT<ContactPP *>> contact_pp_proxy_out_list;
55   VecXT<VecXT<BondEntry *>> bond_entry_pp_proxy_out_list;
56   VecXT<VecXT<CollisionEntry *>> collision_entry_pp_proxy_out_list;
57
59   VecXT<VecXT<ContactPW *>> contact_pw_proxy_out_list;
60   VecXT<VecXT<BondEntry *>> bond_entry_pw_proxy_out_list;
61   VecXT<VecXT<CollisionEntry *>> collision_entry_pw_proxy_out_list;
62
64   VecXT<VecXT<Particle *>> particle_proxy_in_list;
65
67   VecXT<VecXT<ContactPP *>> contact_pp_back_out_list;
68   VecXT<VecXT<BondEntry *>> bond_entry_pp_back_out_list;
69   VecXT<VecXT<CollisionEntry *>> collision_entry_pp_back_out_list;
70
72   VecXT<VecXT<ContactPW *>> contact_pw_back_out_list;
73   VecXT<VecXT<BondEntry *>> bond_entry_pw_back_out_list;
74   VecXT<VecXT<CollisionEntry *>> collision_entry_pw_back_out_list;
75
77   VecXT<VecXT<Particle *>> particle_transfer_out_list;
78
80   VecXT<VecXT<ContactPP *>> contact_pp_transfer_out_list;
81   VecXT<VecXT<BondEntry *>> bond_entry_pp_transfer_out_list;
82   VecXT<VecXT<CollisionEntry *>> collision_entry_pp_transfer_out_list;
83
85   VecXT<VecXT<ContactPW *>> contact_pw_transfer_out_list;
86   VecXT<VecXT<BondEntry *>> bond_entry_pw_transfer_out_list;
87   VecXT<VecXT<CollisionEntry *>> collision_entry_pw_transfer_out_list;
88
89   MPIManager();
90
91   void Init(Simulation *sim);
92   void CommitMPIDataType();
93
94   void BuildContactRef();
95   void CleanUpParticleProxy();
96   void ExchangeDataTransfer();
97   void ExchangeDataProxy();
98   void ExchangeDataBack();
99   void CleanUpParticleGhost();
100   void ClearContactRef();
101
102   void GatherDataProxy();
103   void GatherDataBack();
104   void GatherDataTransfer();
105
106   void SendDataProxy();
107   void SendDataBack();
108   void SendDataTransfer();
109
110   void RecvDataProxy();
111   void RecvDataBack();
112   void RecvDataTransfer();
113
114   void MergeParticleProxy(int source_rank);
115   void MergeContactPPProxy(int source_rank);
116   void MergeContactPWProxy(int source_rank);
117
118   void MergeContactPPBack(int source_rank);
119   void MergeContactPWBack(int source_rank);
120
121   void MergeShapeTransfer(int source_rank);
122
```

```
123   void MergeParticleTransfer(int source_rank);
124   void MergeContactPPTransfer(int source_rank);
125   void MergeContactPWTransfer(int source_rank);
126
127 private:
128   Simulation *sim{nullptr};
129
130   VecXT<ParticleData *> particle_data_list_send;
131   VecXT<ContactPPData *> contact_pp_data_list_send;
132   VecXT<BondEntryData *> bond_entry_pp_data_list_send;
133   VecXT<CollisionEntryData *> collision_entry_pp_data_list_send;
134   VecXT<ContactPWData *> contact_pw_data_list_send;
135   VecXT<BondEntryData *> bond_entry_pw_data_list_send;
136   VecXT<CollisionEntryData *> collision_entry_pw_data_list_send;
137
138   VecXT<int> particle_num_list_send;
139   VecXT<int> contact_pp_num_list_send;
140   VecXT<int> bond_entry_pp_num_list_send;
141   VecXT<int> collision_entry_pp_num_list_send;
142   VecXT<int> contact_pw_num_list_send;
143   VecXT<int> bond_entry_pw_num_list_send;
144   VecXT<int> collision_entry_pw_num_list_send;
145
146   VecXT<MPI_Request> particle_req_list_send;
147   VecXT<MPI_Request> contact_pp_req_list_send;
148   VecXT<MPI_Request> bond_entry_pp_req_list_send;
149   VecXT<MPI_Request> collision_entry_pp_req_list_send;
150   VecXT<MPI_Request> contact_pw_req_list_send;
151   VecXT<MPI_Request> bond_entry_pw_req_list_send;
152   VecXT<MPI_Request> collision_entry_pw_req_list_send;
153
154   std::string *shape_data_send{nullptr};
155   VecXT<MPI_Request> shape_req_list_send;
156   VecXT<std::string> shape_data_list_recv;
157   VecXT<MPI_Request> shape_req_list_recv;
158   VecXT<bool> shape_probed_list;
159
160   VecXT<ParticleData *> particle_data_list_recv;
161   VecXT<ContactPPData *> contact_pp_data_list_recv;
162   VecXT<BondEntryData *> bond_entry_pp_data_list_recv;
163   VecXT<CollisionEntryData *> collision_entry_pp_data_list_recv;
164   VecXT<ContactPWData *> contact_pw_data_list_recv;
165   VecXT<BondEntryData *> bond_entry_pw_data_list_recv;
166   VecXT<CollisionEntryData *> collision_entry_pw_data_list_recv;
167
168   VecXT<int> particle_num_list_recv;
169   VecXT<int> contact_pp_num_list_recv;
170   VecXT<int> bond_entry_pp_num_list_recv;
171   VecXT<int> collision_entry_pp_num_list_recv;
172   VecXT<int> contact_pw_num_list_recv;
173   VecXT<int> bond_entry_pw_num_list_recv;
174   VecXT<int> collision_entry_pw_num_list_recv;
175
176   VecXT<MPI_Request> particle_req_list_recv;
177   VecXT<MPI_Request> contact_pp_req_list_recv;
178   VecXT<MPI_Request> bond_entry_pp_req_list_recv;
179   VecXT<MPI_Request> collision_entry_pp_req_list_recv;
180   VecXT<MPI_Request> contact_pw_req_list_recv;
181   VecXT<MPI_Request> bond_entry_pw_req_list_recv;
182   VecXT<MPI_Request> collision_entry_pw_req_list_recv;
183
184   VecXT<bool> particle_probed_list;
185   VecXT<bool> contact_pp_probed_list;
186   VecXT<bool> bond_entry_pp_probed_list;
187   VecXT<bool> collision_entry_pp_probed_list;
188   VecXT<bool> contact_pw_probed_list;
189   VecXT<bool> bond_entry_pw_probed_list;
190   VecXT<bool> collision_entry_pw_probed_list;
191
192   std::list<int> GetRankList();
193   void RemoveParticle(int id, VecXT<Particle *> *p_list);
194   void ClearBuffer();
195 };
196
197 } // namespace netdem
```

8.188 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/particle_data.hpp File Reference

```
#include <string>
```

Classes

- struct netdem::ParticleData

Namespaces

- namespace netdem

8.189 particle_data.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include <string>
4
5 namespace netdem {
6
19 struct ParticleData {
20   int id{0};
21
22   int shape_id{0};
23   double bound_min[3]{0, 0, 0}, bound_max[3]{0, 0, 0}, margin{0};
24   double bound_disp[3]{0, 0, 0};
25
26   int material_type{0};
27
28   double density{2650};
29   double damp_global{0};
30
31   double pos[3]{0, 0, 0}, quaternion[4]{1, 0, 0, 0};
32   double vel[3]{0, 0, 0}, spin[3]{0, 0, 0};
33   double vel_m0p5[3]{0, 0, 0}, spin_principal[3]{0, 0, 0};
34   double force[3]{0, 0, 0}, moment[3]{0, 0, 0};
35
36   bool enable_rotation{true}, enable_bound_aabb{false};
37
38   bool need_update_linked_list{true};
39 };
40
41 } // namespace netdem
```

8.190 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/mpi/particle_parser.cpp File Reference

```
#include "particle_parser.hpp"
#include "utils_io.hpp"
#include <mpi.h>
```

8.191 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/mpi/particle_parser.hpp File Reference

```
#include "particle.hpp"
#include "particle_data.hpp"
#include <mpi.h>
```

Classes

- class netdem::ParticleParser

Namespaces

- namespace netdem

8.192 particle_parser.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "particle.hpp"
4 #include "particle_data.hpp"
5 #include <mpi.h>
6
7 namespace netdem {
8
12 class ParticleParser {
13 public:
14   static void ClassToStruct(const Particle *const p_class,
15                             ParticleData *const p_struct);
16
17   static void StructToClass(Particle *const p_class,
18                             const ParticleData *const p_struct,
19                             const std::unordered_map<int, Shape *> &shape_map);
20
21   static void DefineMPIDataType(MPI_Datatype *const datatype);
22 };
23
24 } // namespace netdem
```

8.193 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/netdem.hpp File Reference

```
#include "bonded_spheres.hpp"
#include "bonded_voronois.hpp"
#include "breakage_analysis_pd.hpp"
#include "cell_manager.hpp"
#include "contact_solver_factory.hpp"
#include "cork_wrapper.hpp"
#include "data_dumper.hpp"
#include "dem_object_pool.hpp"
```

```
#include "dem_solver.hpp"
#include "distribution.hpp"
#include "distribution_uniform.hpp"
#include "domain_manager.hpp"
#include "domain_splittor.hpp"
#include "gen_pack.hpp"
#include "gen_wall_box_plane.hpp"
#include "gen_wall_box_plate.hpp"
#include "gravity.hpp"
#include "level_set_function.hpp"
#include "level_set_splittor.hpp"
#include "membrane.hpp"
#include "membrane_wall.hpp"
#include "model_hertz_mindlin.hpp"
#include "model_linear_spring.hpp"
#include "model_parallel_bond.hpp"
#include "modifier_manager.hpp"
#include "mpi_manager.hpp"
#include "particle_strength_parameters.hpp"
#include "peridigm_block.hpp"
#include "peridigm_boundary_condition.hpp"
#include "peridigm_damage_model.hpp"
#include "peridigm_dem_coupler.hpp"
#include "peridigm_discretization.hpp"
#include "peridigm_material.hpp"
#include "peridigm_settings.hpp"
#include "peridigm_simulator.hpp"
#include "shape_cylinder.hpp"
#include "shape_level_set.hpp"
#include "shape_poly_super_ellipsoid.hpp"
#include "shape_poly_super_quadrics.hpp"
#include "shape_polybezier.hpp"
#include "shape_sphere.hpp"
#include "shape_spherical_harmonics.hpp"
#include "simulation.hpp"
#include "spherical_voronoi.hpp"
#include "tetmesh.hpp"
#include "tetmesh_splittor.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
#include "wall_disp_control.hpp"
#include "wall_servo_control.hpp"
#include "wscvt_sampler.hpp"
```

8.194 netdem.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bonded_spheres.hpp"
4 #include "bonded_voronois.hpp"
5 #include "breakage_analysis_pd.hpp"
6 #include "cell_manager.hpp"
7 #include "contact_solver_factory.hpp"
8 #include "cork_wrapper.hpp"
9 #include "data_dumper.hpp"
10 #include "dem_object_pool.hpp"
11 #include "dem_solver.hpp"
12 #include "distribution.hpp"
```

```
13 #include "distribution_uniform.hpp"
14 #include "domain_manager.hpp"
15 #include "domain_splittor.hpp"
16 #include "gen_pack.hpp"
17 #include "gen_wall_box_plane.hpp"
18 #include "gen_wall_box_plate.hpp"
19 #include "gravity.hpp"
20 #include "level_set_function.hpp"
21 #include "level_set_splittor.hpp"
22 #include "membrane.hpp"
23 #include "membrane_wall.hpp"
24 #include "model_hertz_mindlin.hpp"
25 #include "model_linear_spring.hpp"
26 #include "model_parallel_bond.hpp"
27 #include "modifier_manager.hpp"
28 #include "mpi_manager.hpp"
29 #include "particle_strength_parameters.hpp"
30 #include "peridigm_block.hpp"
31 #include "peridigm_boundary_condition.hpp"
32 #include "peridigm_damage_model.hpp"
33 #include "peridigm_dem_coupler.hpp"
34 #include "peridigm_discretization.hpp"
35 #include "peridigm_material.hpp"
36 #include "peridigm_settings.hpp"
37 #include "peridigm_simulator.hpp"
38 #include "shape_cylinder.hpp"
39 #include "shape_level_set.hpp"
40 #include "shape_poly_super_ellipsoid.hpp"
41 #include "shape_poly_super_quadrics.hpp"
42 #include "shape_polybezier.hpp"
43 #include "shape_sphere.hpp"
44 #include "shape_spherical_harmonics.hpp"
45 #include "simulation.hpp"
46 #include "spherical_voronoi.hpp"
47 #include "tetmesh.hpp"
48 #include "tetmesh_splittor.hpp"
49 #include "utils_io.hpp"
50 #include "utils_math.hpp"
51 #include "wall_disp_control.hpp"
52 #include "wall_servo_control.hpp"
53 #include "wscvt_sampler.hpp"
```

8.195 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/dem_fragment.cpp File Reference

```
#include "dem_fragment.hpp"
#include "igl_wrapper.hpp"
#include "utils_io.hpp"
#include <cmath>
```

8.196 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/dem_fragment.hpp File Reference

```
#include "level_set_function.hpp"
#include "shape.hpp"
#include "stl_model.hpp"
```

Classes

- class netdem::DEMFragment

Namespaces

- namespace netdem

8.197 dem_fragment.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "level_set_function.hpp"
4 #include "shape.hpp"
5 #include "stl_model.hpp"
6
7 namespace netdem {
8
9 class DEMFragment {
10 public:
11   Shape::Type shape_type{Shape::Type::trimesh};
12
13   double sphere_size{1.0};
14   Vec3d pos{0, 0, 0};
15
16   STLModel stl_model;
17
18   Vec3d vel{0, 0, 0}, spin{0, 0, 0};
19
20   void InitLevelSet(double corner_x, double corner_y, double corner_z,
21                     double sp, int dim_x, int dim_y, int dim_z);
22
23   void ResolverOverlap(DEMFragment *const frag_q);
24
25   void ReInitSTLModel();
26
27 public:
28   LevelSetFunction level_set;
29 };
30
31 } // namespace netdem
```

8.198 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/domain_splittor.hpp File Reference

```
#include "stl_model.hpp"
#include "stl_reader.hpp"
```

Classes

- class netdem::DomainSplittor

Namespaces

- namespace netdem

8.199 domain_splittor.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "stl_model.hpp"
4 #include "stl_reader.hpp"
5
6 namespace netdem {
7
8 class DomainSplittor {
9 public:
10   DomainSplittor() {}
11
12   virtual void InitFromSTL(std::string const &stl_file, int res) {
13     STLModel stl_model;
14     stl_model.InitFromSTL(stl_file);
15     InitFromSTL(stl_model, res);
16   }
17
18   virtual void InitFromSTL(STLModel const &stl_model, int res) = 0;
19
20   virtual void GetPeriDigmNodes(VecXT<Vec3d> *const nodes,
21                                 VecXT<double> *const node_vols) = 0;
22
23   virtual void MakePorosity(double porosity) = 0;
24
25   virtual STLModel GetSTLModel() = 0;
26   virtual STLModel GetSTLModel(const VecXT<int> &indices) = 0;
27 };
28
29 } // namespace netdem
```

8.200 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/level_set_splittor.cpp File Reference

```
#include "level_set_splittor.hpp"
#include "distribution_uniform.hpp"
#include "igl_wrapper.hpp"
#include "tetmesh.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
```

8.201 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/level_set_splittor.hpp File Reference

```
#include "domain_splittor.hpp"
#include "level_set_function.hpp"
#include <iostream>
#include <string>
```

Classes

- class netdem::LevelSetSplittor

Namespaces

- namespace netdem

8.202 level_set_splittor.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "domain_splittor.hpp"
4 #include "level_set_function.hpp"
5 #include <iostream>
6 #include <string>
7
8 namespace netdem {
9
10 class LevelSetSplittor : public DomainSplittor, public LevelSetFunction {
11 public:
12   LevelSetSplittor();
13
14   void InitFromSTL(STLModel const &stl_model, int res) override;
15
16   // data format: corner_x, corner_y, corner_z, spacing, dim_x, dim_y, dim_z,
17   // dist[i][j][k]
18   void InitFromDistanceMap(std::string const &file_name);
19   void InitFromDistanceMap(double corner_x, double corner_y, double corner_z,
20                            double sp, int dim_x, int dim_y, int dim_z,
21                            const VecXT<double> &dist_list);
22
23   void GetPeriDigmNodes(VecXT<Vec3d> *const nodes,
24                         VecXT<double> *const node_vols) override;
25
26   void MakePorosity(double porosity) override;
27
28   STLModel GetSTLModel() override;
29   STLModel GetSTLModel(const VecXT<int> &node_indices) override;
30
31 private:
32   VecXT<Vec3i> node_grid_indices;
33 };
34
35 } // namespace netdem
```

8.203 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/particle_strength_parameters.hpp File Reference

```
#include "utils_math.hpp"
```

Classes

- class netdem::ParticleStrengthParameters

Namespaces

- namespace netdem

8.204 particle_strength_parameters.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_math.hpp"
4
5 namespace netdem {
6
7 class ParticleStrengthParameters {
8 public:
9   // particle breakage strength
10   double ref_size = 1.5e-3;
11   double ref_energy_release_rate = 60.0;
12
13   double weibull_modulus = 3.1;  // McDowell 2002
14   double weibull_coef_a = -0.76; // Hanley et al 2015
15   double weibull_coef_b = 1.13;  // Hanley et al 2015
16
17   double min_breakable_size = 0.02;
18
19   // note that the critical energy release rate is calculated with pre-assuming
20   // that the pd horizon is a same prescribed times of particle size divided by
21   // a same prescribed mesh resolution. Please refer to Zhu & Zhao, 2019 CMAME
22   // for  detials
23   double GetEnergyReleaseRate(double size) {
24     double strength_ratio =
25         (double(rand()) / double(RAND_MAX) - weibull_coef_b) / weibull_coef_a;
26
27     return ref_energy_release_rate * strength_ratio * strength_ratio *
28            pow(size / ref_size, -6.0 / weibull_modulus + 1.0);
29   }
30
31   double GetEnergyReleaseRate(double size, double percentile) {
32     double strength_ratio = (percentile - weibull_coef_b) / weibull_coef_a;
33
34     return ref_energy_release_rate * strength_ratio * strength_ratio *
35            pow(size / ref_size, -6.0 / weibull_modulus + 1.0);
36   }
37 };
38
39 } // namespace netdem
```

8.205 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_block.hpp File Reference

```
#include "utils_io.hpp"
#include <fstream>
#include <sstream>
```

Classes

- class netdem::PeriDigmBlock

Namespaces

- namespace netdem

8.206 peridigm_block.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_io.hpp"
4 #include <fstream>
5 #include <sstream>
6
7 namespace netdem {
8
9 class PeriDigmBlock {
10 public:
11   VecXT<int> node_indices;
12
13   int material_id;
14   int damage_model_id;
15
16   double horizon;
17
18   // to do: change horizon to be node-wise
19   void WriteInputFile(std::ostream &os, int block_id) {
20     os « "    My Block " + std::to_string(block_id + 1) + ": " « std::endl;
21     os « "      Block Names: \"block_" + std::to_string(block_id + 1) + "\""
22        « std::endl;
23     os « "      Material: \"My Material " + std::to_string(material_id + 1) +
24           "\""
25        « std::endl;
26     os « "      Damage Model: \"My Damage Model " +
27           std::to_string(damage_model_id + 1) + "\""
28        « std::endl;
29     os « "      Horizon: " + my_to_string(horizon) « std::endl;
30     os « std::endl;
31   }
32 };
33
34 } // namespace netdem
```

8.207 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_boundary_condition.hpp File Reference

```
#include "utils_io.hpp"
#include "utils_math.hpp"
#include <fstream>
#include <sstream>
#include <string>
```

Classes

- class netdem::PeriDigmBoundaryCondition

Namespaces

- namespace netdem

8.208 peridigm_boundary_condition.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_io.hpp"
4 #include "utils_math.hpp"
5 #include <fstream>
6 #include <sstream>
7 #include <string>
8
9 namespace netdem {
10
11 class PeriDigmBoundaryCondition {
12 public:
13   enum Type { Prescribed_Displacement, Body_Force };
14
15   Type type{Type::Prescribed_Displacement};
16
17   VecXT<int> node_indices;
18
19   VecNT<bool, 3> dim_activated{true, true, true};
20
21   bool time_depedent{true};
22
23   Vec3d disp_rate{0, 0, 0}, loading_rate{0, 0, 0};
24
25   Vec3d disp{0, 0, 0}, loading{0, 0, 0};
26   double mech_time{0};
27
28   void InsertNode(int node_set_id) { node_indices.emplace_back(node_set_id); }
29
30   void SetActivatedDimensions(bool x, bool y, bool z) {
31     dim_activated[0] = x;
32     dim_activated[1] = y;
33     dim_activated[2] = z;
34   }
35
36   void SetByDisplacementRate(double x, double y, double z) {
37     time_depedent = true;
38
39     disp_rate[0] = x;
40     disp_rate[1] = y;
41     disp_rate[2] = z;
42   }
43
44   void SetByUltimateDisplacement(double x, double y, double z, double t) {
45     time_depedent = false;
46
47     disp[0] = x;
48     disp[1] = y;
49     disp[2] = z;
50
51     mech_time = t;
52   }
53
54   void SetByLoadingRate(double x, double y, double z) {
55     time_depedent = true;
56
57     loading_rate[0] = x;
58     loading_rate[1] = y;
59     loading_rate[2] = z;
60   }
61
62   void SetByUltimateLoading(double x, double y, double z, double t) {
63     time_depedent = false;
64
65     loading[0] = x;
66     loading[1] = y;
67     loading[2] = z;
68
69     mech_time = t;
70   }
71
72   void WriteInputFile(std::ostream &os, int node_set_id) {
73     switch (type) {
74     case Type::Prescribed_Displacement:
75       os « "    Node_Set_" + my_to_string(node_set_id + 1) + ": \"" +
76             GetNodeSetFileName(node_set_id) + "\""
77          « std::endl;
78
79       if (dim_activated[0]) {
80         os « "    Displacement_NS_" + my_to_string(node_set_id + 1) + "_X: "
81            « std::endl;
82         os « "      Type: \"Prescribed Displacement\"" « std::endl;
```

```
83          os « "      Node Set: \"Node_Set_" + my_to_string(node_set_id + 1) +
84                    "\""
85             « std::endl;
86          os « "      Coordinate: \"x\"" « std::endl;
87          os « "      Value: " + GetDisplacementString(0) « std::endl;
88        }
89
90        if (dim_activated[1]) {
91          os « "    Displacement_NS_" + my_to_string(node_set_id + 1) + "_Y: "
92             « std::endl;
93          os « "      Type: \"Prescribed Displacement\"" « std::endl;
94          os « "      Node Set: \"Node_Set_" + my_to_string(node_set_id + 1) +
95                    "\""
96             « std::endl;
97          os « "      Coordinate: \"y\"" « std::endl;
98          os « "      Value: " + GetDisplacementString(1) « std::endl;
99        }
100
101       if (dim_activated[2]) {
102         os « "    Displacement_NS_" + my_to_string(node_set_id + 1) + "_Z: "
103            « std::endl;
104         os « "      Type: \"Prescribed Displacement\"" « std::endl;
105         os « "      Node Set: \"Node_Set_" + my_to_string(node_set_id + 1) +
106                   "\""
107            « std::endl;
108         os « "      Coordinate: \"z\"" « std::endl;
109         os « "      Value: " + GetDisplacementString(2) « std::endl;
110       }
111       break;
112
113     case Type::Body_Force:
114       os « "    Node_Set_" + my_to_string(node_set_id + 1) + ": \"" +
115             GetNodeSetFileName(node_set_id) + "\""
116          « std::endl;
117
118       if (dim_activated[0]) {
119         os « "    Loading_NS_" + my_to_string(node_set_id + 1) + "_X: "
120            « std::endl;
121         os « "      Type: \"Body Force\"" « std::endl;
122         os « "      Node Set: \"Node_Set_" + my_to_string(node_set_id + 1) +
123                   "\""
124            « std::endl;
125         os « "      Coordinate: \"x\"" « std::endl;
126         os « "      Value: " + GetLoadingString(0) « std::endl;
127       }
128
129       if (dim_activated[1]) {
130         os « "    Loading_NS_" + my_to_string(node_set_id + 1) + "_Y: "
131            « std::endl;
132         os « "      Type: \"Body Force\"" « std::endl;
133         os « "      Node Set: \"Node_Set_" + my_to_string(node_set_id + 1) +
134                   "\""
135            « std::endl;
136         os « "      Coordinate: \"y\"" « std::endl;
137         os « "      Value: " + GetLoadingString(1) « std::endl;
138       }
139
140       if (dim_activated[2]) {
141         os « "    Loading_NS_" + my_to_string(node_set_id + 1) + "_Z: "
142            « std::endl;
143         os « "      Type: \"Body Force\"" « std::endl;
144         os « "      Node Set: \"Node_Set_" + my_to_string(node_set_id + 1) +
145                   "\""
146            « std::endl;
147         os « "      Coordinate: \"z\"" « std::endl;
148         os « "      Value: " + GetLoadingString(2) « std::endl;
149       }
150       break;
151
152     default:
153       PrintError("in PeriDigmBoundaryCondition::WriteToFile, boundary "
154                  "condition type not defined");
155       break;
156     }
157   }
158
159   void WriteNodeSetFile(std::string const &result_dir, int node_set_id) {
160     std::stringbuf buf;
161     std::ostream os(&buf);
162
163     for (auto nid : node_indices) {
164       // 1-based node numbering in Exodus II, 0-based node numbering in Epetra
165       // and all the rest of Peridigm
166       os « nid + 1 « std::endl;
167     }
168
169     std::ofstream outfile;
```

```
170     outfile.open(result_dir + GetNodeSetFileName(node_set_id));
171     if (!outfile.is_open()) {
172       PrintError(
173           "in PeriDigmBoundaryCondition::WriteNodeFile, cannot open file: " +
174           result_dir + GetNodeSetFileName(node_set_id));
175     } else {
176       outfile << buf.str();
177       outfile.close();
178     }
179   }
180
181   std::string GetNodeSetFileName(int node_set_id) {
182     char filename[128];
183     std::sprintf(filename, "node_set/node_set_%08d.txt", node_set_id + 1);
184     return filename;
185   }
186
187   std::string GetDisplacementString(int dim) {
188     std::string disp_string;
189
190     if (time_depedent) {
191       disp_string = "\"" + my_to_string(disp_rate[dim]) + "*t\"";
192     } else {
193       disp_string = "\"" + my_to_string(disp[dim]) + "\"";
194     }
195
196     return disp_string;
197   }
198
199   std::string GetLoadingString(int dim) {
200     std::string loading_string;
201
202     if (time_depedent) {
203       loading_string = "\"" + my_to_string(loading_rate[dim]) + "*t\"";
204     } else {
205       // one-step loading
206       // loading_string = "\"" + my_to_string(loading[dim]) + "\"";
207
208       // sigmoid-increasing loading to improve stability
209       loading_string = "\"" + my_to_string(loading[dim] / mech_time) +
210                        "*t/sqrt(0.01+t*t/" +
211                        my_to_string(mech_time * mech_time) + ")\"";
212     }
213
214     return loading_string;
215   }
216 };
217
218 } // namespace netdem
```

8.209 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/peridigm_damage_↩ model.hpp File Reference

```
#include "utils_io.hpp"
#include "utils_math.hpp"
#include <fstream>
#include <sstream>
#include <string>
```

Classes

- class netdem::PeriDigmDamageModel

Namespaces

- namespace netdem

8.210 peridigm_damage_model.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_io.hpp"
4 #include "utils_math.hpp"
5 #include <fstream>
6 #include <sstream>
7 #include <string>
8
9 namespace netdem {
10
11 class PeriDigmDamageModel {
12 public:
13   enum Type { Critical_Stretch };
14
15   Type type{Type::Critical_Stretch};
16
17   double critical_stretch{1.0e-2};
18
19   void InitFromEnergyReleaseRate(double youngs_modulus, double poissons_ratio,
20                                  double horizon, double energy_release_rate) {
21     critical_stretch = GetStretchFromEnergyReleaseRate(
22         youngs_modulus, poissons_ratio, horizon, energy_release_rate);
23   }
24
25   static double GetStretchFromEnergyReleaseRate(double youngs_modulus,
26                                                 double poissons_ratio,
27                                                 double horizon,
28                                                 double energy_release_rate) {
29     double shear_modulus = youngs_modulus / 2.0 / (1.0 + poissons_ratio);
30     double bulk_modulus = youngs_modulus / 3.0 / (1.0 - 2.0 * poissons_ratio);
31
32     return sqrt(energy_release_rate /
33                 (3.0 * shear_modulus +
34                  81.0 / 256.0 * (bulk_modulus - 5.0 / 3.0 * shear_modulus)) /
35                 horizon);
36   }
37
38   void WriteInputFile(std::ostream &os, int damage_model_id) {
39     os « "    My Damage Model " + std::to_string(damage_model_id + 1) + ": "
40        « std::endl;
41     switch (type) {
42     case Type::Critical_Stretch:
43       os « "      Damage Model : \"Critical Stretch\"" « std::endl;
44       os « "      Critical Stretch: " + my_to_string(critical_stretch)
45          « std::endl;
46       break;
47
48     default:
49       PrintError(
50           "in PeriDigmDamageModel::WriteToFile, damage model type not defined");
51       break;
52     }
53   }
54 };
55
56 } // namespace netdem
```

8.211 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/peridigm_dem_↩ coupler.cpp File Reference

```
#include "peridigm_dem_coupler.hpp"
#include "cgal_wrapper.hpp"
#include "eigen_wrapper.hpp"
#include "utils_io.hpp"
#include <filesystem>
#include <fstream>
#include <sstream>
```

Namespaces

- namespace netdem

8.212 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/peridigm_dem_↩ coupler.hpp File Reference

```
#include "dem_fragment.hpp"
#include "particle.hpp"
#include "particle_strength_parameters.hpp"
#include "peridigm_simulator.hpp"
#include "simulation.hpp"
#include <string>
```

Classes

- class netdem::PeriDigmDEMCoupler

Namespaces

- namespace netdem

8.213 peridigm_dem_coupler.hpp

Go to the documentation of this file.

```
1 // Originates from Fan Zhu. Modified by Zhengshou Lai to be coupled with NetDEM
2 // by Sep 12, 2021
3
4 #pragma once
5
6 #include "dem_fragment.hpp"
7 #include "particle.hpp"
8 #include "particle_strength_parameters.hpp"
9 #include "peridigm_simulator.hpp"
10 #include "simulation.hpp"
11 #include <string>
12
13 namespace netdem {
14
15 class PeriDigmDEMCoupler {
16 public:
18   std::string base_dir{"tmp/out/"};
19   int sub_dir_index{0};
20
21   // particle and discretization info
22   Particle *particle{nullptr};
23   PeriDigmSimulator pd_sim;
24
25   STLModel surface_stl;
26   int mesh_res{20};
27   double node_size_ave{0.0};
28
29   // for boundary conditions in pd, note that all boundary forces are actually
30   // force density in pd
31   VecXT<int> fixed_nodes;
32
```

```
33   VecXT<int> boundary_force_nodes;
34   VecXT<double> boundary_force_node_vols;
35   VecXT<Vec3d> boundary_force_values;
36
37   // unbalanced force is introduced as intertia force such that the toal force
38   // is zero
39   VecXT<int> unbalanced_force_nodes;
40   Vec3d unbalanced_force_values{0, 0, 0};
41
42   // for estimate the loading rate
43   double contact_force_max{0.0};
44   VecXT<double> contact_force_list;
45
46   // for pd loading and time settings
47   bool use_customized_loading_rate{false};
48
49   double loading_rate{1.0e5};
50   int loading_steps{1000};
51
52   double mech_time{0.0};
53
55   bool is_broken{false};
56
57   // if fraction of damaged bond is greater than the limit, the fragments
58   // reconstructin routine is invoked
59   double damage_fraction_limit{0.05};
60
61   // fragment is reconstructed at trimesh only if volume is greater than the
62   // limit; otherwise, it will be considered as spherical fines
63   double fragment_vol_limit{0.001};
64   bool ignore_fines{true};
65
66   bool use_alpha_shape{true};
67   double fragment_alpha{0.0};
68
69   // particle breakage strength parameters
70   ParticleStrengthParameters strength_params;
71   PeriDigmMaterial material_params;
72
73   PeriDigmDEMCoupler();
74
75   void Init(Particle *p);
76
77   void Solve();
78
79   void ApplyBoundaryForce(Vec3d const &pos, Vec3d const &force);
80
81   bool CheckBreakage();
82
83   VecXT<DEMFragment> GetFragments();
84
85 private:
86   // in peridigm, bond has only two states, i.e., 0 and 1. The damage limit is
87   // kept there for pential improvements in future
88   double damage_limit{0.5};
89
90   // data structure: node id i, node id j, damage, ...
91   VecXT<VecXT<double>> damage_data;
92
93   void UpdateMaterials();
94   void UpdateMechTime();
95   void UpdateCriticalStretch();
96
97   std::string GetResultDirectory();
98
99   VecXT<int> SeperateFragments(const VecXT<Vec2i> &bond_list);
100
101   VecXT<int> GetFragmentNodeIndices(const VecXT<Vec2i> &bond_list,
102                                     const VecXT<int> &frag_id_list,
103                                     int frag_id);
104
105   VecXT<Vec3d> GetFragmentNodes(const VecXT<Vec3d> &node_list,
106                                 const VecXT<int> &node_ids);
107
108   VecXT<Vec3d> GetFragmentNodeVelocities(const VecXT<Vec3d> &velocity_list,
109                                          const VecXT<int> &node_ids);
110
111   VecXT<double> GetFragmentNodevolumes(const VecXT<double> &node_vol_list,
112                                        const VecXT<int> &node_ids);
113
114   STLModel GetAlphaShape(const VecXT<Vec3d> &point_list, double alpha);
115
116   void ResolveFragmentOverlap(VecXT<DEMFragment> *const frag_list);
117
118   DEMFragment GetFragmentCombined(const VecXT<DEMFragment> &frag_list);
119
120   void WriteLogFileDEM();
```

```
121 };
122
123 } // namespace netdem
```

8.214 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/peridigm_↩ discretization.cpp File Reference

```
#include "peridigm_discretization.hpp"
#include "stl_reader.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
#include <fstream>
#include <sstream>
```

8.215 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/peridigm_↩ discretization.hpp File Reference

```
#include "level_set_splittor.hpp"
#include "tetmesh_splittor.hpp"
#include <fstream>
#include <sstream>
#include <string>
```

Classes

- class netdem::PeriDigmDiscretization

Namespaces

- namespace netdem

8.216 peridigm_discretization.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "level_set_splittor.hpp"
4 #include "tetmesh_splittor.hpp"
5 #include <fstream>
6 #include <sstream>
7 #include <string>
8
```

```
9 namespace netdem {
10
11 class PeriDigmDiscretization {
12 public:
13   enum Type { level_set, tetmesh };
14   Type type{Type::level_set};
15
16   DomainSplittor *domain_splittor{nullptr};
17
18   VecXT<Vec3d> nodes;
19   VecXT<int> node_block_indices;
20   VecXT<double> node_vols;
21
22   PeriDigmDiscretization();
23
24   void InitFromSTL(std::string const &stl_file, int res);
25
26   void InitFromSTL(STLModel const &stl_model, int res);
27
28   // data format: corner_x, corner_y, corner_z, spacing, dim_x, dim_y, dim_z,
29   // dist[i][j][k]
30   void InitFromDistanceMap(std::string const &txt_file);
31
32   void InitFromGrid(double corner_x, double corner_y, double corner_z,
33                     double len_x, double len_y, double len_z, int res);
34
35   void MakePorosity(double porosity);
36
37   void WriteNodeFile(std::string const &result_dir);
38
39   double GetNodeSize();
40
41   ~PeriDigmDiscretization();
42
43 private:
44   void InitDefaultBlockIndices();
45 };
46
47 } // namespace netdem
```

8.217 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/peridigm_↩ material.hpp File Reference

```
#include "utils_io.hpp"
#include <cmath>
```

Classes

- class netdem::PeriDigmMaterial

Namespaces

- namespace netdem

8.218 peridigm_material.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_io.hpp"
4 #include <cmath>
5
6 namespace netdem {
7
8 class PeriDigmMaterial {
9 public:
10   enum Type { Elastic };
11
12   Type type{Type::Elastic};
13
14   // density of sand
15   double density = 2650.0;
16
17   // Peridigm parameters
18   double youngs_modulus = 70.0e9;
19   double poissons_ratio = 0.15;
20
21   void WriteInputFile(std::ostream &os, int material_id) {
22     os « "    My Material " + std::to_string(material_id + 1) + ": "
23        « std::endl;
24     switch (type) {
25     case Type::Elastic:
26       os « "      Material Model : \"Elastic\"" « std::endl;
27       os « "      Density: " + my_to_string(density) « std::endl;
28       os « "      Young's Modulus: " + my_to_string(youngs_modulus)
29          « std::endl;
30       os « "      Poisson's Ratio: " + my_to_string(poissons_ratio)
31          « std::endl;
32       break;
33
34     default:
35       PrintError("in PeriDigmMaterial::WriteToFile, material type not defined");
36       break;
37     }
38   }
39 };
40
41 } // namespace netdem
```

8.219 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/peridigm/peridigm_settings.hpp File Reference

```
#include "utils_math.hpp"
#include <cmath>
#include <fstream>
#include <sstream>
#include <string>
```

Classes

- class netdem::PeriDigmSettings

Namespaces

- namespace netdem

8.220 peridigm_settings.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_math.hpp"
4 #include <cmath>
5 #include <fstream>
6 #include <sstream>
7 #include <string>
8
9 namespace netdem {
10
11 class PeriDigmSettings {
12 public:
13   std::string result_dir{"tmp/out/peridigm/"};
14   std::string peridigm_exe{"Peridigm"};
15
16   // horizon = node_size * horizon_factor
17   double horizon_factor{3.01};
18
19   bool omit_bonds_between_blocks{false};
20
21   // timestep settings
22   bool use_auto_timestep{true};
23   double timestep{1.0e-6};
24   double timestep_factor{0.95};
25
26   double mech_time{0.0};
27
28   // for point contact, the contact force is scatter onto the nodes arround this
29   // contact point
30   double loading_radius_factor{1.5};
31   double constrain_radius_factor{1.5};
32
33   // write output every x steps in peridigm
34   int output_freqency{10};
35
36   void WriteInputFile(std::ostream &os) {
37     // solver settings
38     os « "  Solver: " « std::endl;
39     os « "    Verbose: false" « std::endl;
40     os « "    Initial Time: 0.0" « std::endl;
41     os « "    Final Time: " + my_to_string(mech_time) « std::endl;
42     os « "    Verlet: " « std::endl;
43     if (!use_auto_timestep) {
44       os « "      Fixed dt: " + my_to_string(timestep) « std::endl;
45     } else {
46       os « "      Safety Factor: " + my_to_string(timestep_factor)
47          « std::endl;
48     }
49     os « std::endl;
50
51     // compute settings, for fraguting the stored strain energy in the particle
52     os « "  Compute Class Parameters: " « std::endl;
53     os « "    Grain Stored Elastic Energy: " « std::endl;
54     os « "      Compute Class: \"Block_Data\"" « std::endl;
55     os « "      Calculation Type: \"Sum\"" « std::endl;
56     os « "      Block: \"block_1\"" « std::endl;
57     os « "      Variable: \"Stored_Elastic_Energy\"" « std::endl;
58     os « "      Output Label: \"Grain_Stored_Elastic_Energy\"" « std::endl;
59     os « std::endl;
60
61     // output settings
62     os « "  Output Data: " « std::endl;
63     os « "    Output File Type: \"ExodusII\"" « std::endl;
64     os « "    Output Format: \"BINARY\"" « std::endl;
65     os « "    Output Filename: \"output\"" « std::endl;
66     os « "    Output Frequency: " + my_to_string(output_freqency) « std::endl;
67     os « "    Parallel Write: true" « std::endl;
68     os « "    Output Variables: " « std::endl;
69     os « "      Displacement: true" « std::endl;
70     os « "      Velocity: true" « std::endl;
71     os « "      Element_Id: true" « std::endl;
72     os « "      Dilatation: true" « std::endl;
73     os « "      Weighted_Volume: true" « std::endl;
74     os « "      Volume: true" « std::endl;
75     os « "      Damage: true" « std::endl;
76     os « "      Radius: true" « std::endl;
77     os « "      Horizon: true" « std::endl;
78     os « "      Contact_Force: true" « std::endl;
79     os « "      Number_Of_Neighbors: true" « std::endl;
80     os « "      Neighborhood_Volume: true" « std::endl;
81     os « "      Force: true" « std::endl;
82     os « "      Force_Density: true" « std::endl;
```

```
83     os « "      Stored_Elastic_Energy: true" « std::endl;
84     os « "      Grain_Stored_Elastic_Energy: true" « std::endl;
85     os « "      Kinetic_Energy: true" « std::endl;
86     os « std::endl;
87   }
88 };
89
90 } // namespace netdem
```

8.221 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/peridigm_↩ simulator.cpp File Reference

```
#include "peridigm_simulator.hpp"
#include "cgal_wrapper.hpp"
#include "eigen_wrapper.hpp"
#include "utils_io.hpp"
#include <filesystem>
#include <unordered_set>
```

Namespaces

- namespace netdem

8.222 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/peridigm_↩ simulator.hpp File Reference

```
#include "peridigm_block.hpp"
#include "peridigm_boundary_condition.hpp"
#include "peridigm_damage_model.hpp"
#include "peridigm_discretization.hpp"
#include "peridigm_material.hpp"
#include "peridigm_settings.hpp"
#include <string>
```

Classes

- class netdem::PeriDigmSimulator

Namespaces

- namespace netdem

8.223 peridigm_simulator.hpp

Go to the documentation of this file.

```
1 // Originates from Fan Zhu. Modified by Zhengshou Lai to be coupled with NetDEM
2 // by Sep 12, 2021
3
4 #pragma once
5
6 #include "peridigm_block.hpp"
7 #include "peridigm_boundary_condition.hpp"
8 #include "peridigm_damage_model.hpp"
9 #include "peridigm_discretization.hpp"
10 #include "peridigm_material.hpp"
11 #include "peridigm_settings.hpp"
12 #include <string>
13
14
15 namespace netdem {
16
17 class PeriDigmSimulator {
18 public:
19   PeriDigmDiscretization discretization;
20
21   VecXT<PeriDigmMaterial> materials;
22
23   VecXT<PeriDigmDamageModel> damage_models;
24
25   VecXT<PeriDigmBlock> blocks;
26
27   VecXT<PeriDigmBoundaryCondition> boundary_conditions;
28
29   PeriDigmSettings settings;
30
31 public:
32   PeriDigmSimulator();
33
34   PeriDigmMaterial *InsertMaterial();
35   PeriDigmDamageModel *InsertDamageModel();
36   PeriDigmBlock *InsertBlock();
37   PeriDigmBoundaryCondition *InsertBoundaryCondition();
38
39   void Clear();
40
41   // init default setup based on discretization and using default material and
42   // damage model
43   void InitDefaultSetup();
44
45   void InitAutoTimestep();
46
47   void WriteNodeFile();
48
49   void WriteNodeSetFile();
50
51   void WriteInputFile();
52
53   void Solve(double mech_time);
54
55   void SetUpResultDirectory();
56   void CleanUpResultDirectory();
57 };
58
59 } // namespace netdem
```

8.224 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/tetmesh_splittor.cpp File Reference

```
#include "tetmesh_splittor.hpp"
#include "distribution_uniform.hpp"
#include "igl_wrapper.hpp"
#include "utils_math.hpp"
#include <unordered_set>
```

Namespaces

- namespace netdem

8.225 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/peridigm/tetmesh_splittor.hpp File Reference

```
#include "domain_splittor.hpp"
#include "tetmesh.hpp"
#include <iostream>
#include <string>
```

Classes

- class netdem::TetMeshSplittor

Namespaces

- namespace netdem

8.226 tetmesh_splittor.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "domain_splittor.hpp"
4 #include "tetmesh.hpp"
5 #include <iostream>
6 #include <string>
7
8 namespace netdem {
9
10 class TetMeshSplittor : public DomainSplittor {
11 public:
12   TetMesh tetmesh;
13
14   TetMeshSplittor();
15
16   void InitFromSTL(STLModel const &stl_model, int res) override;
17
18   void GetPeriDigmNodes(VecXT<Vec3d> *const nodes,
19                         VecXT<double> *const node_vols) override;
20
21   void MakePorosity(double porosity) override;
22
23   STLModel GetSTLModel() override;
24   STLModel GetSTLModel(const VecXT<int> &tet_indices) override;
25 };
26
27 } // namespace netdem
```

8.227 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/pybind/pydem.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/operators.h>
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitPyContactModel (pybind11::module &m)
- void InitPyLinearSpring (pybind11::module &m)
- void InitPyParallelBond (pybind11::module &m)
- void InitPyContactSolverFactory (pybind11::module &m)
- void InitPyDEMSolver (pybind11::module &m)
- void InitPyDEMModule (pybind11::module &m)

8.227.1 Function Documentation

8.227.1.1 InitPyContactModel()

```
void InitPyContactModel (
            pybind11::module & m )
```

8.227.1.2 InitPyContactSolverFactory()

```
void InitPyContactSolverFactory (
            pybind11::module & m )
```

8.227.1.3 InitPyDEMModule()

```
void InitPyDEMModule (
            pybind11::module & m )
```

8.227.1.4 InitPyDEMSolver()

```
void InitPyDEMSolver (
            pybind11::module & m )
```

8.227.1.5 InitPyLinearSpring()

```
void InitPyLinearSpring (
            pybind11::module & m )
```

8.227.1.6 InitPyParallelBond()

```
void InitPyParallelBond (
            pybind11::module & m )
```

8.228 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/pybind/pydomain.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitPyCellManager (pybind11::module &m)
- void InitPyDomain (pybind11::module &m)
- void InitPyDomainManager (pybind11::module &m)
- void InitPyDomainModule (pybind11::module &m)

8.228.1 Function Documentation

8.228.1.1 InitPyCellManager()

```
void InitPyCellManager (
            pybind11::module & m )
```

8.228.1.2 InitPyDomain()

```
void InitPyDomain (
            pybind11::module & m )
```

8.228.1.3 InitPyDomainManager()

```
void InitPyDomainManager (
            pybind11::module & m )
```

8.228.1.4 InitPyDomainModule()

```
void InitPyDomainModule (
            pybind11::module & m )
```

8.229 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/pybind/pyfem.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/numpy.h>
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitTetMesh (pybind11::module &m)
- void InitPyMembrane (pybind11::module &m)
- void InitPyFEM (pybind11::module &m)

8.229.1 Function Documentation

8.229.1.1 InitPyFEM()

```
void InitPyFEM (
            pybind11::module & m )
```

8.229.1.2 InitPyMembrane()

```
void InitPyMembrane (
            pybind11::module & m )
```

8.229.1.3 InitTetMesh()

```
void InitTetMesh (
            pybind11::module & m )
```

8.230 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/pybind/pymodifier.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitPyModifierManager (pybind11::module &m)
- void InitPyModifier (pybind11::module &m)
- void InitPyBreakageAnalysisPD (pybind11::module &m)
- void InitPyDataDumper (pybind11::module &m)
- void InitPyGravity (pybind11::module &m)
- void InitPyMembraneWall (pybind11::module &m)
- void InitPyWallDispControl (pybind11::module &m)
- void InitPyWallServoControl (pybind11::module &m)
- void InitPyModifierModule (pybind11::module &m)

8.230.1 Function Documentation

8.230.1.1 InitPyBreakageAnalysisPD()

```
void InitPyBreakageAnalysisPD (
            pybind11::module & m )
```

8.230.1.2 InitPyDataDumper()

```
void InitPyDataDumper (
            pybind11::module & m )
```

8.230.1.3 InitPyGravity()

```
void InitPyGravity (
            pybind11::module & m )
```

8.230.1.4 InitPyMembraneWall()

```
void InitPyMembraneWall (
            pybind11::module & m )
```

8.230.1.5 InitPyModifier()

```
void InitPyModifier (
            pybind11::module & m )
```

8.230.1.6 InitPyModifierManager()

```
void InitPyModifierManager (
            pybind11::module & m )
```

8.230.1.7 InitPyModifierModule()

```
void InitPyModifierModule (
            pybind11::module & m )
```

8.230.1.8 InitPyWallDispControl()

```
void InitPyWallDispControl (
            pybind11::module & m )
```

8.230.1.9 InitPyWallServoControl()

```
void InitPyWallServoControl (
            pybind11::module & m )
```

8.231 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/pybind/pynetdem.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitPyDEMModule (pybind11::module &m)
- void InitPyDomainModule (pybind11::module &m)
- void InitPyModifierModule (pybind11::module &m)
- void InitPyFEM (pybind11::module &m)
- void InitPyPeriDigmModule (pybind11::module &m)
- void InitPySceneModule (pybind11::module &m)
- void InitPyShapeModule (pybind11::module &m)
- void InitPySimulationModule (pybind11::module &m)
- void InitPyUtilsModule (pybind11::module &m)
- PYBIND11_MODULE (pynetdem, m)

  sphere

8.231.1 Function Documentation

8.231.1.1 InitPyDEMModule()

```
void InitPyDEMModule (
            pybind11::module & m )
```

8.231.1.2 InitPyDomainModule()

```
void InitPyDomainModule (
            pybind11::module & m )
```

8.231.1.3 InitPyFEM()

```
void InitPyFEM (
            pybind11::module & m )
```

8.231.1.4 InitPyModifierModule()

```
void InitPyModifierModule (
            pybind11::module & m )
```

8.231.1.5 InitPyPeriDigmModule()

```
void InitPyPeriDigmModule (
            pybind11::module & m )
```

8.231.1.6 InitPySceneModule()

```
void InitPySceneModule (
            pybind11::module & m )
```

8.231.1.7 InitPyShapeModule()

```
void InitPyShapeModule (
            pybind11::module & m )
```

8.231.1.8 InitPySimulationModule()

```
void InitPySimulationModule (
            pybind11::module & m )
```

8.231.1.9 InitPyUtilsModule()

```
void InitPyUtilsModule (
            pybind11::module & m )
```

8.231.1.10 PYBIND11_MODULE()

```
PYBIND11_MODULE (
            pynetdem ,
            m  )
```

sphere

8.232 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/pybind/pyperidigm.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/numpy.h>
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitPyDomainSplittor (pybind11::module &m)
- void InitPyLevelSetSplittor (pybind11::module &m)
- void InitPyTetMeshSplittor (pybind11::module &m)
- void InitPyPeriDigmDiscretization (pybind11::module &m)
- void InitPyPeriDigmMaterial (pybind11::module &m)
- void InitPyPeriDigmDamageModel (pybind11::module &m)
- void InitPyPeriDigmBlock (pybind11::module &m)
- void InitPyPeriDigmBoundaryCondition (pybind11::module &m)
- void InitPyPeriDigmSettings (pybind11::module &m)
- void InitPyDEMFragment (pybind11::module &m)
- void InitPyParticleStrengthParameters (pybind11::module &m)
- void InitPyPeriDigmSimulator (pybind11::module &m)
- void InitPyPeriDigmDEMCoupler (pybind11::module &m)
- void InitPyPeriDigmModule (pybind11::module &m)

8.232.1 Function Documentation

8.232.1.1 InitPyDEMFragment()

```
void InitPyDEMFragment (
            pybind11::module & m )
```

8.232.1.2 InitPyDomainSplittor()

```
void InitPyDomainSplittor (
            pybind11::module & m )
```

8.232.1.3 InitPyLevelSetSplittor()

```
void InitPyLevelSetSplittor (
            pybind11::module & m )
```

8.232.1.4 InitPyParticleStrengthParameters()

```
void InitPyParticleStrengthParameters (
            pybind11::module & m )
```

8.232.1.5 InitPyPeriDigmBlock()

```
void InitPyPeriDigmBlock (
            pybind11::module & m )
```

8.232.1.6 InitPyPeriDigmBoundaryCondition()

```
void InitPyPeriDigmBoundaryCondition (
            pybind11::module & m )
```

8.232.1.7 InitPyPeriDigmDamageModel()

```
void InitPyPeriDigmDamageModel (
            pybind11::module & m )
```

8.232.1.8 InitPyPeriDigmDEMCoupler()

```
void InitPyPeriDigmDEMCoupler (
            pybind11::module & m )
```

8.232.1.9 InitPyPeriDigmDiscretization()

```
void InitPyPeriDigmDiscretization (
            pybind11::module & m )
```

8.232.1.10 InitPyPeriDigmMaterial()

```
void InitPyPeriDigmMaterial (
            pybind11::module & m )
```

8.232.1.11 InitPyPeriDigmModule()

```
void InitPyPeriDigmModule (
            pybind11::module & m )
```

8.232.1.12 InitPyPeriDigmSettings()

```
void InitPyPeriDigmSettings (
            pybind11::module & m )
```

8.232.1.13 InitPyPeriDigmSimulator()

```
void InitPyPeriDigmSimulator (
            pybind11::module & m )
```

8.232.1.14 InitPyTetMeshSplittor()

```
void InitPyTetMeshSplittor (
            pybind11::module & m )
```

8.233 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/pybind/pyscene.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/numpy.h>
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitPyDEMObjectPool (pybind11::module &m)
- void InitPyScene (pybind11::module &m)
- void InitPyPackGenerator (pybind11::module &m)
- void InitPyParticle (pybind11::module &m)
- void InitPyWall (pybind11::module &m)
- void InitPyWallBoxPlane (pybind11::module &m)
- void InitPyBondedSpheres (pybind11::module &m)
- void InitPyBondedVoronois (pybind11::module &m)
- void InitPySceneModule (pybind11::module &m)

8.233.1 Function Documentation

8.233.1.1 InitPyBondedSpheres()

```
void InitPyBondedSpheres (
            pybind11::module & m )
```

8.233.1.2 InitPyBondedVoronois()

```
void InitPyBondedVoronois (
            pybind11::module & m )
```

8.233.1.3 InitPyDEMObjectPool()

```
void InitPyDEMObjectPool (
            pybind11::module & m )
```

8.233.1.4 InitPyPackGenerator()

```
void InitPyPackGenerator (
            pybind11::module & m )
```

8.233.1.5 InitPyParticle()

```
void InitPyParticle (
            pybind11::module & m )
```

8.233.1.6 InitPyScene()

```
void InitPyScene (
            pybind11::module & m )
```

8.233.1.7 InitPySceneModule()

```
void InitPySceneModule (
            pybind11::module & m )
```

8.233.1.8 InitPyWall()

```
void InitPyWall (
            pybind11::module & m )
```

8.233.1.9 InitPyWallBoxPlane()

```
void InitPyWallBoxPlane (
            pybind11::module & m )
```

8.234 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/pybind/pyshape.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/numpy.h>
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitPyShape (pybind11::module &m)
- void InitPySphere (pybind11::module &m)
- void InitPyPlane (pybind11::module &m)
- void InitPyCylinder (pybind11::module &m)
- void InitPySphericalHarmonics (pybind11::module &m)
- void InitPyPolySuperEllipsoid (pybind11::module &m)
- void InitPyLevelSet (pybind11::module &m)
- void InitPyTriMesh (pybind11::module &m)
- void InitPyShapeModule (pybind11::module &m)

8.234.1 Function Documentation

8.234.1.1 InitPyCylinder()

```
void InitPyCylinder (
            pybind11::module & m )
```

8.234.1.2 InitPyLevelSet()

```
void InitPyLevelSet (
            pybind11::module & m )
```

8.234.1.3 InitPyPlane()

```
void InitPyPlane (
            pybind11::module & m )
```

8.234.1.4 InitPyPolySuperEllipsoid()

```
void InitPyPolySuperEllipsoid (
            pybind11::module & m )
```

8.234.1.5 InitPyShape()

```
void InitPyShape (
            pybind11::module & m )
```

8.234.1.6 InitPyShapeModule()

```
void InitPyShapeModule (
            pybind11::module & m )
```

8.234.1.7 InitPySphere()

```
void InitPySphere (
            pybind11::module & m )
```

8.234.1.8 InitPySphericalHarmonics()

```
void InitPySphericalHarmonics (
            pybind11::module & m )
```

8.234.1.9 InitPyTriMesh()

```
void InitPyTriMesh (
            pybind11::module & m )
```

8.235 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/pybind/pysimulation.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitPySimulation (pybind11::module &m)
- void InitPySimulationModule (pybind11::module &m)

8.235.1 Function Documentation

8.235.1.1 InitPySimulation()

```
void InitPySimulation (
            pybind11::module & m )
```

8.235.1.2 InitPySimulationModule()

```
void InitPySimulationModule (
            pybind11::module & m )
```

8.236 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/pybind/pyutils.cpp File Reference

```
#include "netdem.hpp"
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
```

Functions

- void InitPyLevelSetFunction (pybind11::module &m)
- void InitPySTLReader (pybind11::module &m)
- void InitPySTLModel (pybind11::module &m)
- void InitPyWSCVTSampler (pybind11::module &m)
- void InitPyVoronoi (pybind11::module &m)
- void InitPyCork (pybind11::module &m)
- void InitPyOpenMP (pybind11::module &m)
- void InitPyUtilsModule (pybind11::module &m)

8.236.1 Function Documentation

8.236.1.1 InitPyCork()

```
void InitPyCork (
            pybind11::module & m )
```

8.236.1.2 InitPyLevelSetFunction()

```
void InitPyLevelSetFunction (
            pybind11::module & m )
```

8.236.1.3 InitPyOpenMP()

```
void InitPyOpenMP (
            pybind11::module & m )
```

8.236.1.4 InitPySTLModel()

```
void InitPySTLModel (
            pybind11::module & m )
```

8.236.1.5 InitPySTLReader()

```
void InitPySTLReader (
            pybind11::module & m )
```

8.236.1.6 InitPyUtilsModule()

```
void InitPyUtilsModule (
            pybind11::module & m )
```

8.236.1.7 InitPyVoronoi()

```
void InitPyVoronoi (
            pybind11::module & m )
```

8.236.1.8 InitPyWSCVTSampler()

```
void InitPyWSCVTSampler (
            pybind11::module & m )
```

8.237 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/bonded_spheres.cpp File Reference

```
#include "bonded_spheres.hpp"
#include "distribution_uniform.hpp"
#include "scene.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
```

8.238 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/bonded_spheres.hpp File Reference

```
#include "bond_solver_pp.hpp"
#include "contact_model.hpp"
#include "contact_pp.hpp"
#include "particle.hpp"
#include "shape_sphere.hpp"
#include <string>
```

Classes

- class netdem::BondedSpheres

Namespaces

- namespace netdem

8.239 bonded_spheres.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_solver_pp.hpp"
4 #include "contact_model.hpp"
5 #include "contact_pp.hpp"
6 #include "particle.hpp"
7 #include "shape_sphere.hpp"
8 #include <string>
9
10 namespace netdem {
11
12 class Scene;
13
14 class BondedSpheres {
15 public:
16   Sphere sphere;
17
18   VecXT<Particle> particle_list;
19   VecXT<ContactPP> contact_list;
20   VecXT<Vec2i> bond_pair_list;
```

```
21
22   ContactModel *bond_model{nullptr};
23
24   BondedSpheres();
25
26   BondedSpheres(BondedSpheres const &bp);
27   BondedSpheres(BondedSpheres const &&bp);
28   BondedSpheres &operator=(BondedSpheres const &bp);
29   BondedSpheres &operator=(BondedSpheres const &&bp);
30
31   void SetBondModel(ContactModel *cnt_model);
32
33   void Translate(double pos_x, double pos_y, double pos_z);
34   void RotateByRodrigues(double rot_angle, double rot_axis_x, double rot_axis_y,
35                          double rot_axis_z);
36
37   Vec3d GetCentroid();
38
39   void InitFromSTL(std::string const &filename, double sphere_size);
40   void InitFromSTL(STLModel const &stl_model, double sphere_size);
41
42   void InitFromGrid(double corner_x, double corner_y, double corner_z,
43                     double len_x, double len_y, double len_z,
44                     double sphere_size);
45
46   void MakePorosity(double porosity);
47
48   void InitBonds();
49
50   void ImportToScene(Scene *const scene) const;
51
52 private:
53   void RefreshPointers();
54 };
55
56 } // namespace netdem
```

8.240 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/bonded_voronois.cpp File Reference

```
#include "bonded_voronois.hpp"
#include "distribution_uniform.hpp"
#include "eigen_wrapper.hpp"
#include "scene.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
```

8.241 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/bonded_voronois.hpp File Reference

```
#include "bond_solver_pp.hpp"
#include "contact_model.hpp"
#include "contact_pp.hpp"
#include "particle.hpp"
#include "shape_trimesh.hpp"
#include "voronoi.hpp"
#include <string>
```

Classes

- class netdem::BondedVoronois

Namespaces

- namespace netdem

8.242 bonded_voronois.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_solver_pp.hpp"
4 #include "contact_model.hpp"
5 #include "contact_pp.hpp"
6 #include "particle.hpp"
7 #include "shape_trimesh.hpp"
8 #include "voronoi.hpp"
9 #include <string>
10
11 namespace netdem {
12
13 class Scene;
14
15 class BondedVoronois {
16 public:
17   VecXT<TriMesh> trimesh_list;
18   VecXT<Particle> particle_list;
19   VecXT<ContactPP> contact_list;
20   VecXT<Vec2i> bond_pair_list;
21
22   int cvt_max_iters{1000};
23   double cvt_tol{1.0e-3};
24
25   ContactModel *bond_model{nullptr};
26
27   BondedVoronois();
28
29   BondedVoronois(BondedVoronois const &bp);
30   BondedVoronois(BondedVoronois const &&bp);
31   BondedVoronois &operator=(BondedVoronois const &bp);
32   BondedVoronois &operator=(BondedVoronois const &&bp);
33
34   void SetBondModel(ContactModel *cnt_model);
35
36   void Translate(double pos_x, double pos_y, double pos_z);
37   void RotateByRodrigues(double rot_angle, double rot_axis_x, double rot_axis_y,
38                          double rot_axis_z);
39
40   Vec3d GetCentroid();
41
42   void InitFromSTL(std::string const &filename, int num_voros);
43   void InitFromSTL(STLModel const &stl_model, int num_voros);
44
45   void MakePorosity(double porosity);
46
47   void InitBonds();
48
49   void RefreshPointers();
50
51   void SaveAsVTK(std::string const &file_name);
52
53   void ImportToScene(Scene *const scene) const;
54
55 private:
56   VecXT<Vec3d> FindSharedVertices(STLModel const &stl_1, STLModel const &stl_2);
57
58   Vec3d PolyNormal(VecXT<Vec3d> const &verts);
59
60   void PolySortVertices(VecXT<Vec3d> *const pos_vec, Vec3d const &dir_n);
61
62   // return: centroid, area
63   std::tuple<Vec3d, double> PolyCentroid(const VecXT<Vec3d> &verts);
64 };
65
66 } // namespace netdem
```

8.243 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/contact_pp.cpp File Reference

```
#include "contact_pp.hpp"
#include "dem_object_pool.hpp"
#include "utils_math.hpp"
#include <iostream>
```

8.244 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/contact_pp.hpp File Reference

```
#include "bond_entry.hpp"
#include "collision_entry.hpp"
#include "mini_map.hpp"
#include "particle.hpp"
```

Classes

- class netdem::ContactPP
- class netdem::NeighPofP
- class netdem::NeighWofP

Namespaces

- namespace netdem

8.245 contact_pp.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_entry.hpp"
4 #include "collision_entry.hpp"
5 #include "mini_map.hpp"
6 #include "particle.hpp"
7
8 namespace netdem {
9
10 class Particle;
11
30 class ContactPP {
31 public:
32   Particle *particle_1{nullptr}, *particle_2{nullptr};
33
34   ContactModel *bond_model{nullptr}, *collision_model{nullptr};
35
36   VecXT<BondEntry> bond_entries;
37   VecXT<CollisionEntry> collision_entries;
38
39   bool active{true};
40
```

```
41   MiniMap<std::string, double> dynamic_properties;
42
43   ContactPP();
44   ContactPP(Particle *const p1, Particle *const p2);
45
46   void Init(Particle *const p1, Particle *const p2);
47
48   void SetBondModel(ContactModel *const cnt_model);
49   void SetCollisionModel(ContactModel *const cnt_model);
50
51   // evaluate contact forces and moments
52   void EvaluateForceMoment(double dt);
53
54   // apply contact forces to the particles
55   void ApplyToParticle();
56
57   void ApplyToParticle1();
58   void ApplyToParticle2();
59
60   bool IsActive();
61
62   void Clear();
63
64   void Print();
65 };
66
73 class NeighPofP {
74 public:
75   Particle *particle{nullptr};
76   int lookup_id{-1};
77   ContactPP *contact{nullptr};
78
79   NeighPofP() {}
80
81   NeighPofP(Particle *const p, int id, ContactPP *const cnt)
82       : particle(p), lookup_id(id), contact(cnt) {}
83 };
84
91 class NeighWofP {
92 public:
93   Wall *wall{nullptr};
94   int lookup_id{-1};
95   ContactPW *contact{nullptr};
96
97   NeighWofP() {}
98
99   NeighWofP(Wall *const w, int id, ContactPW *const cnt)
100       : wall(w), lookup_id(id), contact(cnt) {}
101 };
102
103 } // namespace netdem
```

8.246 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/contact_pw.cpp File Reference

```
#include "contact_pw.hpp"
#include "dem_object_pool.hpp"
#include "utils_math.hpp"
#include <iostream>
```

8.247 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/contact_pw.hpp File Reference

```
#include "bond_entry.hpp"
#include "collision_entry.hpp"
```

```
#include "mini_map.hpp"
#include "particle.hpp"
#include "wall.hpp"
#include <string>
```

Classes

- class netdem::ContactPW
- class netdem::NeighPofW

Namespaces

- namespace netdem

8.248 contact_pw.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bond_entry.hpp"
4 #include "collision_entry.hpp"
5 #include "mini_map.hpp"
6 #include "particle.hpp"
7 #include "wall.hpp"
8 #include <string>
9
10 namespace netdem {
11
30 class ContactPW {
31 public:
32   Particle *particle{nullptr};
33   Wall *wall{nullptr};
34
35   ContactModel *bond_model{nullptr}, *collision_model{nullptr};
36
37   VecXT<BondEntry> bond_entries;
38   VecXT<CollisionEntry> collision_entries;
39
40   bool active{true};
41
42   MiniMap<std::string, double> dynamic_properties;
43
44   ContactPW();
45   ContactPW(Particle *const p, Wall *const w);
46
47   void Init(Particle *const p, Wall *const w);
48
49   void SetBondModel(ContactModel *const cnt_model);
50   void SetCollisionModel(ContactModel *const cnt_model);
51
52   // evaluate contact forces and moments
53   void EvaluateForceMoment(double dt);
54
55   void ApplyToParticle(); // apply contact forces to the particle
56   void ApplyToWall();     // apply contact forces to the wall
57
58   bool IsActive();
59
60   void Clear();
61
62   void Print();
63 };
64
71 class NeighPofW {
72 public:
73   Particle *particle{nullptr};
74   int lookup_id{-1};
75   ContactPW *contact{nullptr};
76
```

```
77   NeighPofW() {}
78
79   NeighPofW(Particle *const p, int id, ContactPW *const cnt)
80       : particle(p), lookup_id(id), contact(cnt) {}
81 };
82
83 } // namespace netdem
```

8.249 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/dem_object_pool.cpp File Reference

```
#include "dem_object_pool.hpp"
#include <omp.h>
```

Namespaces

- namespace netdem

8.250 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/dem_object_pool.hpp File Reference

```
#include "contact_pp.hpp"
#include "contact_pw.hpp"
#include "particle.hpp"
#include <memory>
```

Classes

- class netdem::DEMObjectPool

  particles and contacts are frequently added to or removed from the scene. The pool strategy is used to avoid the frequently construction and de-construction of object instances. When a particle or wall needs to be added, an instances will be obtained from the pool. When a particle or wall needs to be removed,it is recycled and stored in the pool. to do: object pool need to be improved

Namespaces

- namespace netdem

8.251 dem_object_pool.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "contact_pp.hpp"
4 #include "contact_pw.hpp"
5 #include "particle.hpp"
6 #include <memory>
7
8 namespace netdem {
9
19 class DEMObjectPool {
20 public:
21   DEMObjectPool(const DEMObjectPool &) = delete;
22   DEMObjectPool &operator=(const DEMObjectPool &) = delete;
23   static DEMObjectPool &GetInstance() {
24     static DEMObjectPool instance;
25     return instance;
26   }
27
28   Particle *GetParticle();
29   ContactPP *GetContactPP();
30   ContactPW *GetContactPW();
31
32   ContactPP *Clone(ContactPP const *cnt);
33   ContactPW *Clone(ContactPW const *cnt);
34
35   void RecycleParticle(Particle **p);
36   void RecycleContactPP(ContactPP **cnt);
37   void RecycleContactPW(ContactPW **cnt);
38
39   void RecycleParticle(VecXT<Particle *> *p_list);
40   void RecycleContactPP(VecXT<ContactPP *> *cnt_list);
41   void RecycleContactPW(VecXT<ContactPW *> *cnt_list);
42
43   void RecycleParticle(VecXT<VecXT<Particle *>> *p_list);
44
45   ~DEMObjectPool();
46
47 private:
48   DEMObjectPool() {}
49
50   VecXT<Particle *> particle_pool;
51   VecXT<ContactPP *> contact_pp_pool;
52   VecXT<ContactPW *> contact_pw_pool;
53 };
54
55 } // namespace netdem
```

8.252 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/gen_pack.hpp File Reference

```
#include "bonded_spheres.hpp"
#include "bonded_voronois.hpp"
#include "distribution_uniform.hpp"
#include "particle.hpp"
#include "shape.hpp"
#include "shape_sphere.hpp"
#include "utils_math.hpp"
```

Classes

- class netdem::PackGenerator

Namespaces

- namespace netdem

8.253 gen_pack.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bonded_spheres.hpp"
4 #include "bonded_voronois.hpp"
5 #include "distribution_uniform.hpp"
6 #include "particle.hpp"
7 #include "shape.hpp"
8 #include "shape_sphere.hpp"
9 #include "utils_math.hpp"
10
11 namespace netdem {
12
28 class PackGenerator {
29 public:
30   static VecXT<Particle> GetGridPack(double len_x, double len_y, double len_z,
31                                      double center_x, double center_y,
32                                      double center_z, int num_x, int num_y,
33                                      int num_z,
34                                      const VecXT<Shape *> &shape_list) {
35     VecXT<Particle> particle_list;
36     particle_list.resize(num_x * num_y * num_z);
37
38     double spacing_x = len_x / num_x;
39     double spacing_y = len_y / num_y;
40     double spacing_z = len_z / num_z;
41
42     UniformDistribution unifrom_dist(0, 1);
43
44     for (int i = 0; i < num_x; i++) {
45       for (int j = 0; j < num_y; j++) {
46         for (int k = 0; k < num_z; k++) {
47           int shape_id = unifrom_dist.Get() * shape_list.size();
48           double pos_rand = unifrom_dist.Get() * shape_list[shape_id]->skin;
49           Particle p = Particle(shape_list[shape_id]);
50           p.SetPosition(
51               spacing_x * (i + 0.5) + center_x - 0.5 * len_x + pos_rand,
52               spacing_y * (j + 0.5) + center_y - 0.5 * len_y + pos_rand,
53               spacing_z * (k + 0.5) + center_z - 0.5 * len_z);
54           Vec3d rot_axis{unifrom_dist.Get(), unifrom_dist.Get(),
55                          unifrom_dist.Get()};
56           Math::Normalize(&rot_axis);
57           p.SetRodrigues(unifrom_dist.Get() * Math::PI, rot_axis[0], rot_axis[1],
58                          rot_axis[2]);
59           particle_list[i * num_y * num_z + j * num_z + k] = p;
60         }
61       }
62     }
63     return particle_list;
64   }
65
66   static VecXT<Particle> GetGridPack(double len_x, double len_y, double len_z,
67                                      double center_x, double center_y,
68                                      double center_z, int num_x, int num_y,
69                                      int num_z, Shape *shape) {
70     VecXT<Particle> particle_list;
71     particle_list.resize(num_x * num_y * num_z);
72
73     double spacing_x = len_x / num_x;
74     double spacing_y = len_y / num_y;
75     double spacing_z = len_z / num_z;
76
77     UniformDistribution unifrom_dist(0, 1);
78
79     for (int i = 0; i < num_x; i++) {
80       for (int j = 0; j < num_y; j++) {
81         for (int k = 0; k < num_z; k++) {
82           Particle p = Particle(shape);
83           double pos_rand = unifrom_dist.Get() * shape->skin;
84           p.SetPosition(
85               spacing_x * (i + 0.5) + center_x - 0.5 * len_x + pos_rand,
86               spacing_y * (j + 0.5) + center_y - 0.5 * len_y + pos_rand,
87               spacing_z * (k + 0.5) + center_z - 0.5 * len_z);
88           Vec3d rot_axis{unifrom_dist.Get(), unifrom_dist.Get(),
```

```
89                          unifrom_dist.Get()};
90            Math::Normalize(&rot_axis);
91            p.SetRodrigues(unifrom_dist.Get() * Math::PI, rot_axis[0], rot_axis[1],
92                           rot_axis[2]);
93            particle_list[i * num_y * num_z + j * num_z + k] = p;
94          }
95        }
96      }
97      return particle_list;
98    }
99
100   static VecXT<BondedSpheres>
101   GetGridPack(double len_x, double len_y, double len_z, double center_x,
102               double center_y, double center_z, int num_x, int num_y, int num_z,
103               const BondedSpheres &bonded_spheres_template) {
104     VecXT<BondedSpheres> particle_list;
105     particle_list.resize(num_x * num_y * num_z);
106
107     double spacing_x = len_x / num_x;
108     double spacing_y = len_y / num_y;
109     double spacing_z = len_z / num_z;
110
111     UniformDistribution unifrom_dist(0, 1);
112
113     for (int i = 0; i < num_x; i++) {
114       for (int j = 0; j < num_y; j++) {
115         for (int k = 0; k < num_z; k++) {
116           double pos_rand =
117               unifrom_dist.Get() * bonded_spheres_template.sphere.skin;
118           BondedSpheres bp = bonded_spheres_template;
119
120           bp.Translate(
121               spacing_x * (i + 0.5) + center_x - 0.5 * len_x + pos_rand,
122               spacing_y * (j + 0.5) + center_y - 0.5 * len_y + pos_rand,
123               spacing_z * (k + 0.5) + center_z - 0.5 * len_z);
124           Vec3d rot_axis{unifrom_dist.Get(), unifrom_dist.Get(),
125                          unifrom_dist.Get()};
126           Math::Normalize(&rot_axis);
127           bp.RotateByRodrigues(unifrom_dist.Get() * Math::PI, rot_axis[0],
128                                rot_axis[1], rot_axis[2]);
129           particle_list[i * num_y * num_z + j * num_z + k] = bp;
130         }
131       }
132     }
133     return particle_list;
134   }
135
136   static VecXT<BondedVoronois>
137   GetGridPack(double len_x, double len_y, double len_z, double center_x,
138               double center_y, double center_z, int num_x, int num_y, int num_z,
139               const BondedVoronois &bonded_voronois_template) {
140     VecXT<BondedVoronois> particle_list;
141     particle_list.resize(num_x * num_y * num_z);
142
143     double spacing_x = len_x / num_x;
144     double spacing_y = len_y / num_y;
145     double spacing_z = len_z / num_z;
146
147     UniformDistribution unifrom_dist(0, 1);
148
149     for (int i = 0; i < num_x; i++) {
150       for (int j = 0; j < num_y; j++) {
151         for (int k = 0; k < num_z; k++) {
152           double pos_rand = unifrom_dist.Get() *
153                             bonded_voronois_template.trimesh_list[0].skin;
154           BondedVoronois bp = bonded_voronois_template;
155           bp.RefreshPointers();
156
157           bp.Translate(
158               spacing_x * (i + 0.5) + center_x - 0.5 * len_x + pos_rand,
159               spacing_y * (j + 0.5) + center_y - 0.5 * len_y + pos_rand,
160               spacing_z * (k + 0.5) + center_z - 0.5 * len_z);
161           Vec3d rot_axis{unifrom_dist.Get(), unifrom_dist.Get(),
162                          unifrom_dist.Get()};
163           Math::Normalize(&rot_axis);
164           bp.RotateByRodrigues(unifrom_dist.Get() * Math::PI, rot_axis[0],
165                                rot_axis[1], rot_axis[2]);
166           particle_list[i * num_y * num_z + j * num_z + k] = bp;
167         }
168       }
169     }
170     return particle_list;
171   }
172 };
173
174 } // namespace netdem
```

8.254 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/gen_wall_box_plane.hpp File Reference

```
#include "scene.hpp"
#include "shape_plane.hpp"
#include "wall.hpp"
```

Classes

- class netdem::WallBoxPlane

Namespaces

- namespace netdem

8.255 gen_wall_box_plane.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "scene.hpp"
4 #include "shape_plane.hpp"
5 #include "wall.hpp"
6
7 namespace netdem {
8
16 class WallBoxPlane {
17 public:
18   Plane p_mx, p_px, p_my, p_py, p_mz, p_pz;
19   Wall w_mx, w_px, w_my, w_py, w_mz, w_pz;
20
21   WallBoxPlane(double len_x, double len_y, double len_z, double center_x,
22                double center_y, double center_z) {
23     p_mx = Plane(center_x - 0.5 * len_x, center_y, center_z, 1, 0, 0);
24     p_px = Plane(center_x + 0.5 * len_x, center_y, center_z, -1, 0, 0);
25     p_my = Plane(center_x, center_y - 0.5 * len_y, center_z, 0, 1, 0);
26     p_py = Plane(center_x, center_y + 0.5 * len_y, center_z, 0, -1, 0);
27     p_mz = Plane(center_x, center_y, center_z - 0.5 * len_z, 0, 0, 1);
28     p_pz = Plane(center_x, center_y, center_z + 0.5 * len_z, 0, 0, -1);
29
30     double sqrt_2_by_2 = 2.0 * std::sqrt(2.0);
31     p_mx.SetExtent(std::max(len_y, len_z) * sqrt_2_by_2);
32     p_px.SetExtent(std::max(len_y, len_z) * sqrt_2_by_2);
33     p_my.SetExtent(std::max(len_x, len_z) * sqrt_2_by_2);
34     p_py.SetExtent(std::max(len_x, len_z) * sqrt_2_by_2);
35     p_mz.SetExtent(std::max(len_x, len_y) * sqrt_2_by_2);
36     p_pz.SetExtent(std::max(len_x, len_y) * sqrt_2_by_2);
37
38     w_mx = Wall(&p_mx);
39     w_px = Wall(&p_px);
40     w_my = Wall(&p_my);
41     w_py = Wall(&p_py);
42     w_mz = Wall(&p_mz);
43     w_pz = Wall(&p_pz);
44   }
45
46   VecXT<Shape *> GetShapes() {
47     VecXT<Shape *> plane_list;
48     plane_list.emplace_back(&p_mx);
49     plane_list.emplace_back(&p_px);
50     plane_list.emplace_back(&p_my);
51     plane_list.emplace_back(&p_py);
52     plane_list.emplace_back(&p_mz);
```

```
53      plane_list.emplace_back(&p_pz);
54      return plane_list;
55    }
56
57    VecXT<Wall *> GetWalls() {
58      VecXT<Wall *> wall_list;
59      wall_list.emplace_back(&w_mx);
60      wall_list.emplace_back(&w_px);
61      wall_list.emplace_back(&w_my);
62      wall_list.emplace_back(&w_py);
63      wall_list.emplace_back(&w_mz);
64      wall_list.emplace_back(&w_pz);
65      return wall_list;
66    }
67
68    void ImportToScene(Scene *scene) {
69      auto shape_ptr_list_local = GetShapes();
70      auto wall_ptr_list_local = GetWalls();
71
72      auto shape_ptr_list = scene->InsertShape(shape_ptr_list_local);
73
74      for (int i = 0; i < 6; i++) {
75        wall_ptr_list_local[i]->SetShape(shape_ptr_list[i]);
76        scene->InsertWall(wall_ptr_list_local[i]);
77        wall_ptr_list_local[i]->SetShape(shape_ptr_list_local[i]);
78      }
79    }
80  };
81
82  } // namespace netdem
```

8.256 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/gen_wall_box_plate.hpp File Reference

```
#include "scene.hpp"
#include "shape_trimesh.hpp"
#include "wall.hpp"
```

Classes

- class netdem::WallBoxPlate

Namespaces

- namespace netdem

8.257 gen_wall_box_plate.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "scene.hpp"
4 #include "shape_trimesh.hpp"
5 #include "wall.hpp"
6
7 namespace netdem {
8
9 class WallBoxPlate {
10 public:
```

```
11   TriMesh p_x0, p_y0, p_z0;
12   Wall w_mx, w_px, w_my, w_py, w_mz, w_pz;
13
14   WallBoxPlate(double len_x, double len_y, double len_z, double center_x,
15                double center_y, double center_z) {
16     double t_factor = 0.1;
17     double len_factor = 1 + 2 * t_factor;
18     double pan_factor = 0.5 * (1 + t_factor);
19
20     p_x0 = GetBox(len_x * t_factor, len_y * len_factor, len_z * len_factor);
21     p_x0.bound_sphere_radius = Math::NormLen(p_x0.vertices[0]);
22     w_mx = w_px = Wall(&p_x0);
23     w_mx.SetPosition(center_x - pan_factor * len_x, center_y, center_z);
24     w_px.SetPosition(center_x + pan_factor * len_x, center_y, center_z);
25
26     p_y0 = GetBox(len_x * len_factor, len_y * t_factor, len_z * len_factor);
27     p_y0.bound_sphere_radius = Math::NormLen(p_x0.vertices[0]);
28     w_my = w_py = Wall(&p_y0);
29     w_my.SetPosition(center_x, center_y - pan_factor * len_y, center_z);
30     w_py.SetPosition(center_x, center_y + pan_factor * len_y, center_z);
31
32     p_z0 = GetBox(len_x * len_factor, len_y * len_factor, len_z * t_factor);
33     p_z0.bound_sphere_radius = Math::NormLen(p_x0.vertices[0]);
34     w_mz = w_pz = Wall(&p_z0);
35     w_mz.SetPosition(center_x, center_y, center_z - pan_factor * len_z);
36     w_pz.SetPosition(center_x, center_y, center_z + pan_factor * len_z);
37   }
38
39   VecXT<Shape *> GetShapes() {
40     VecXT<Shape *> plate_list;
41     plate_list.emplace_back(&p_x0);
42     plate_list.emplace_back(&p_y0);
43     plate_list.emplace_back(&p_z0);
44     return plate_list;
45   }
46
47   VecXT<Wall *> GetWalls() {
48     VecXT<Wall *> wall_list;
49     wall_list.emplace_back(&w_mx);
50     wall_list.emplace_back(&w_px);
51     wall_list.emplace_back(&w_my);
52     wall_list.emplace_back(&w_py);
53     wall_list.emplace_back(&w_mz);
54     wall_list.emplace_back(&w_pz);
55     return wall_list;
56   }
57
58   void ImportToScene(Scene *scene) {
59     auto shape_ptr_list_local = GetShapes();
60     auto wall_ptr_list_local = GetWalls();
61
62     auto shape_ptr_list = scene->InsertShape(shape_ptr_list_local);
63
64     for (int i = 0; i < 6; i++) {
65       wall_ptr_list_local[i]->SetShape(shape_ptr_list[i / 2]);
66       scene->InsertWall(wall_ptr_list_local[i]);
67       wall_ptr_list_local[i]->SetShape(shape_ptr_list_local[i / 2]);
68     }
69   }
70
71 private:
72   TriMesh GetBox(double len_x, double len_y, double len_z) {
73     TriMesh box_ref;
74     box_ref.vertices.resize(8);
75     box_ref.facets.resize(12);
76
77     box_ref.vertices[0][0] = -0.5 * len_x;
78     box_ref.vertices[0][1] = -0.5 * len_y;
79     box_ref.vertices[0][2] = -0.5 * len_z;
80
81     box_ref.vertices[1][0] = -0.5 * len_x;
82     box_ref.vertices[1][1] = 0.5 * len_y;
83     box_ref.vertices[1][2] = -0.5 * len_z;
84
85     box_ref.vertices[2][0] = -0.5 * len_x;
86     box_ref.vertices[2][1] = 0.5 * len_y;
87     box_ref.vertices[2][2] = 0.5 * len_z;
88
89     box_ref.vertices[3][0] = -0.5 * len_x;
90     box_ref.vertices[3][1] = -0.5 * len_y;
91     box_ref.vertices[3][2] = 0.5 * len_z;
92
93     box_ref.vertices[4][0] = 0.5 * len_x;
94     box_ref.vertices[4][1] = -0.5 * len_y;
95     box_ref.vertices[4][2] = -0.5 * len_z;
96
97     box_ref.vertices[5][0] = 0.5 * len_x;
```

```
98      box_ref.vertices[5][1] = 0.5 * len_y;
99      box_ref.vertices[5][2] = -0.5 * len_z;
100
101      box_ref.vertices[6][0] = 0.5 * len_x;
102      box_ref.vertices[6][1] = 0.5 * len_y;
103      box_ref.vertices[6][2] = 0.5 * len_z;
104
105      box_ref.vertices[7][0] = 0.5 * len_x;
106      box_ref.vertices[7][1] = -0.5 * len_y;
107      box_ref.vertices[7][2] = 0.5 * len_z;
108
109      box_ref.facets[0][0] = 0;
110      box_ref.facets[0][1] = 3;
111      box_ref.facets[0][2] = 2;
112
113      box_ref.facets[1][0] = 0;
114      box_ref.facets[1][1] = 2;
115      box_ref.facets[1][2] = 1;
116
117      box_ref.facets[2][0] = 4;
118      box_ref.facets[2][1] = 5;
119      box_ref.facets[2][2] = 6;
120
121      box_ref.facets[3][0] = 4;
122      box_ref.facets[3][1] = 6;
123      box_ref.facets[3][2] = 7;
124
125      box_ref.facets[4][0] = 7;
126      box_ref.facets[4][1] = 3;
127      box_ref.facets[4][2] = 0;
128
129      box_ref.facets[5][0] = 7;
130      box_ref.facets[5][1] = 0;
131      box_ref.facets[5][2] = 4;
132
133      box_ref.facets[6][0] = 6;
134      box_ref.facets[6][1] = 5;
135      box_ref.facets[6][2] = 1;
136
137      box_ref.facets[7][0] = 6;
138      box_ref.facets[7][1] = 1;
139      box_ref.facets[7][2] = 2;
140
141      box_ref.facets[8][0] = 0;
142      box_ref.facets[8][1] = 1;
143      box_ref.facets[8][2] = 5;
144
145      box_ref.facets[9][0] = 0;
146      box_ref.facets[9][1] = 5;
147      box_ref.facets[9][2] = 4;
148
149      box_ref.facets[10][0] = 2;
150      box_ref.facets[10][1] = 3;
151      box_ref.facets[10][2] = 7;
152
153      box_ref.facets[11][0] = 2;
154      box_ref.facets[11][1] = 7;
155      box_ref.facets[11][2] = 6;
156
157      box_ref.Init();
158      return box_ref;
159    }
160 };
161
162 } // namespace netdem
```

8.258 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/particle.cpp File Reference

```
#include "particle.hpp"
#include "cell_manager.hpp"
#include "contact_pp.hpp"
#include "contact_pw.hpp"
#include "dem_object_pool.hpp"
#include "domain_manager.hpp"
```

```
#include "mpi_manager.hpp"
#include "utils_math.hpp"
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
```

8.259 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/particle.hpp File Reference

```
#include "mini_map.hpp"
#include "shape.hpp"
#include "shape_trimesh.hpp"
#include <omp.h>
#include <unordered_map>
```

Classes

- class netdem::Particle

Namespaces

- namespace netdem

8.260 particle.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "mini_map.hpp"
4 #include "shape.hpp"
5 #include "shape_trimesh.hpp"
6 #include <omp.h>
7 #include <unordered_map>
8
9 namespace netdem {
10
11 class Wall;
12 class ContactPP;
13 class ContactPW;
14 class Cell;
15 class Domain;
16 class DomainManager;
17 class NeighPofP;
18 class NeighWofP;
19
20 class Particle {
21 public:
22   int id{0};
23
24   Shape *shape{nullptr};
25   Vec3d bound_min{0, 0, 0}, bound_max{0, 0, 0};
26   double margin{0};
27   Vec3d bound_disp{0, 0, 0}; // for linked list
28
29   int material_type{0};
```

```
30
31   double density{2650}, mass{0.0};
32   Vec3d moi_principal{0.0};
33   double damp_global{0};
34
35   Vec3d pos{0, 0, 0};
36   Vec4d quaternion{1, 0, 0, 0};
37   Vec3d vel{0, 0, 0}, spin{0, 0, 0};
38   Vec3d vel_m0p5{0, 0, 0}, spin_principal{0, 0, 0};
39   Vec3d force{0, 0, 0}, moment{0, 0, 0};
40
42   MiniMap<std::string, double> dynamic_properties;
43
46   bool enable_rotation{true}, enable_bound_aabb{false};
47
51   bool need_update_linked_list{true};
52   VecXT<std::pair<Cell *, int>> linked_cell_list;
53
54   VecXT<NeighPofP> linked_particle_list;
55   VecXT<NeighWofP> linked_wall_list;
56
57   // to recontruct the reference of contact history
58   VecXT<NeighPofP> contact_pp_ref_table;
59   VecXT<NeighWofP> contact_pw_ref_table;
60
64   bool is_on_edge{false};
65   bool need_send_out{false};
66   VecXT<std::pair<Domain *, int>> linked_domain_list;
67
70   bool need_update_stl_model{false};
71   STLModel stl_model;
72
73 public:
74   Particle();
75   Particle(Shape *const shape);
76
77   virtual Particle *Clone() const;
78
79   virtual void Init();
80
81   virtual void SetShape(Shape *const shape);
82   virtual void SetDensity(double dens);
83
84   virtual void SetForce(double fx, double fy, double fz);
85   virtual void SetMoment(double mx, double my, double mz);
86
87   virtual void SetPosition(double pos_x, double pos_y, double pos_z);
88   virtual void SetRodrigues(double angle, double axis_x, double axis_y,
89                             double axis_z);
90   virtual void SetQuaternion(double q_0, double q_1, double q_2, double q_3);
91
92   virtual void SetVelocity(double v_x, double v_y, double v_z);
93   virtual void SetSpin(double spin_x, double spin_y, double spin_z);
94
95   virtual Vec3d GetVelocity(Vec3d const &cnt_pos);
96
97   virtual void AddForce(const Vec3d &force);
98   virtual void AddMoment(const Vec3d &moment);
99
100   virtual void AddForceAtomic(const Vec3d &f);
101   virtual void AddMomentAtomic(const Vec3d &m);
102
103   virtual void ClearForce();
104   virtual void ClearMoment();
105
106   virtual void ApplyContactForce(ContactPP const *cnt);
107   virtual void ApplyContactForce(ContactPW const *cnt);
108
109   virtual void UpdateContactForce();
110
111   virtual void UpdateMotion(double timestep);
112   virtual void UpdateBound();
113
114   void ClearLinkedCells();
115   void ClearLinkedDomains();
116   void ClearLinkedNeighs();
117
118   void BuildContactRef();
119   void ClearContactRef();
120
121   void UpdateLinkedCells(DomainManager *const dm);
122   void UpdateLinkedDomains(DomainManager *const dm);
123   void UpdateLinkedNeighs(DomainManager *const dm);
124
125   VecXT<ContactPP *> GetContactPPs();
126   VecXT<ContactPW *> GetContactPWs();
127
```

```
128   virtual void UpdateSTLModel();
129   virtual STLModel GetSTLModel(int num_facet = 400);
130
131   virtual void SaveAsVTK(std::string const &filename);
132
133   virtual void Print() const;
134
135   virtual ~Particle();
136
137 public:
138   // insert a link tuple for two objects, and return the tuple belong to self
139   // note that the existing tuple is not checked, and no need to call twice for
140   // the two objects, respectively
141   NeighPofP *MakeLinked(Particle *const q);
142   NeighWofP *MakeLinked(Wall *const w);
143
144   NeighPofP *BuildContactRef(Particle *const q, ContactPP *const cnt);
145   NeighWofP *BuildContactRef(Wall *const w, ContactPW *const cnt);
146
147   int FindLinked(Particle *const q);
148   int FindLinked(Wall *const w);
149
150   int FindContactRef(Particle *const q);
151   int FindContactRef(Wall *const w);
152 };
153
154 } // namespace netdem
```

8.261 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/scene.cpp File Reference

```
#include "scene.hpp"
#include "dem_object_pool.hpp"
#include "shape_factory.hpp"
#include "simulation.hpp"
#include "utils_io.hpp"
#include <filesystem>
#include <fstream>
#include <iostream>
#include <omp.h>
```

8.262 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/scene/scene.hpp File Reference

```
#include "bonded_spheres.hpp"
#include "bonded_voronois.hpp"
#include "contact_model.hpp"
#include "contact_pp.hpp"
#include "contact_pw.hpp"
#include "particle.hpp"
#include "shape.hpp"
#include "wall.hpp"
#include <list>
#include <unordered_map>
#include <unordered_set>
```

Classes

- class netdem::Scene

Namespaces

- namespace netdem

8.263 scene.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "bonded_spheres.hpp"
4 #include "bonded_voronois.hpp"
5 #include "contact_model.hpp"
6 #include "contact_pp.hpp"
7 #include "contact_pw.hpp"
8 #include "particle.hpp"
9 #include "shape.hpp"
10 #include "wall.hpp"
11 #include <list>
12 #include <unordered_map>
13 #include <unordered_set>
14
15 namespace netdem {
16
17 class Simulation;
18
54 class Scene {
55 public:
57   Vec3d gravity_coef{0, 0, -9.81};
58
61   VecXT<Wall *> wall_list;
62
64   MiniMap<int, ContactModel *> contact_model_map;
65
67   VecXT<VecXT<ContactModel *>> bond_model_table;
68   VecXT<VecXT<ContactModel *>> collision_model_table;
69
71   VecXT<Particle *> particle_list;
72
76   VecXT<Particle *> particle_proxy_list;
77
86   VecXT<Particle *> particle_ghost_list;
87
91   VecXT<Wall *> wall_ghost_list;
92
100   std::unordered_map<int, Particle *> particle_map;
101
106   std::unordered_map<int, Shape *> shape_map;
107
110   VecXT<Shape *> local_shape_list;
111
112   Scene();
113
114   void Init(Simulation *const sim);
115
119   Shape *InsertShape(const Shape *const s_ptr);
120   VecXT<Shape *> InsertShape(const VecXT<Shape *> &s_list);
121
123   Particle *InsertParticle(const Particle *const p_ptr);
124   Particle *InsertParticle(Particle const &p);
125   VecXT<Particle *> InsertParticle(const VecXT<Particle *> &p_list);
126   VecXT<Particle *> InsertParticle(const VecXT<Particle> &p_list);
127
128   void InsertParticle(const BondedSpheres *const p_ptr);
129   void InsertParticle(const VecXT<BondedSpheres *> &p_list);
130   void InsertParticle(const VecXT<BondedSpheres> &p_list);
131
132   void InsertParticle(const BondedVoronois *const p_ptr);
133   void InsertParticle(const VecXT<BondedVoronois *> &p_list);
134   void InsertParticle(const VecXT<BondedVoronois> &p_list);
135
136   // insert derived particles
137   Particle *InsertDerivedParticle(Particle const *p_ptr);
```

```
138   VecXT<Particle *> InsertDerivedParticle(const VecXT<Particle *> &p_list);
139
141   Wall *InsertWall(const Wall *const w_ptr);
142   Wall *InsertWall(Wall const &w);
143   VecXT<Wall *> InsertWall(const VecXT<Wall *> &w_list);
144   VecXT<Wall *> InsertWall(const VecXT<Wall> &w_list);
145
147   void RemoveShape(Shape *s_ptr);
148
150   void RemoveParticle(Particle *p_ptr);
151
153   void RemoveWall(Wall *w_ptr);
154
156   ContactModel *InsertContactModel(const ContactModel *const cm_ptr);
157   VecXT<ContactModel *>
158   InsertContactModel(const VecXT<ContactModel *> &cm_list);
159
161   VecXT<Shape *> GetShapes();
162
164   bool InScene(const Shape *const s_ptr);
165   bool InScene(const ContactModel *const cnt_ptr);
166
170   void SetNumberOfMaterials(int num);
171   void SetBondModel(int mat_type_1, int mat_type_2,
172                     ContactModel *const cnt_model);
173   void SetBondModel(int mat_type_1, int mat_type_2, std::string const &label);
174   void SetCollisionModel(int mat_type_1, int mat_type_2,
175                          ContactModel *const cnt_model);
176   void SetCollisionModel(int mat_type_1, int mat_type_2,
177                          std::string const &label);
178
179   void SetGravity(double gx, double gy, double gz);
180
181   ContactModel *GetBondModel(Particle *p1, Particle *p2);
182   ContactModel *GetBondModel(Particle *p, Wall *w);
183   ContactModel *GetCollisionModel(Particle *p1, Particle *p2);
184   ContactModel *GetCollisionModel(Particle *p, Wall *w);
185
187   // shape info, 0: 0-step json, 2: time-specific json
188   void AutoReadRestart(std::string const &path, int mech_cyc,
189                        int shape_info_case = 0);
190
191   void ReadRestartShapes(std::string const &file);
192   void ReadRestartParticles(std::string const &file);
193   void ReadRestartWalls(std::string const &file);
194   void ReadRestartContacts(std::string const &file);
195
197   VecXT<ContactPP *> GetContactPPs();
198
200   VecXT<ContactPW *> GetContactPWs();
201
202   void ClearShapes();
203
204   void ClearParticles();
205
206   void ClearWalls();
207
208   void ClearContactModels();
209
210   void ClearContacts();
211
212   ~Scene();
213
214 private:
215   Simulation *sim{nullptr};
216
223   static const int max_num_particles{2000000}, max_num_shapes{2000000};
224
226   int max_id_particles{-1}, max_id_shapes{-1};
227 };
228
229 } // namespace netdem
```

8.264 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/wall.cpp File Reference

```
#include "wall.hpp"
#include "cell_manager.hpp"
```

```
#include "contact_pp.hpp"
#include "contact_pw.hpp"
#include "dem_object_pool.hpp"
#include "domain_manager.hpp"
#include "particle.hpp"
#include "utils_math.hpp"
#include <iostream>
```

8.265 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/scene/wall.hpp File Reference

```
#include "mini_map.hpp"
#include "shape.hpp"
#include "shape_trimesh.hpp"
#include <omp.h>
#include <unordered_map>
```

Classes

- class netdem::Wall

Namespaces

- namespace netdem

8.266 wall.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "mini_map.hpp"
4 #include "shape.hpp"
5 #include "shape_trimesh.hpp"
6 #include <omp.h>
7 #include <unordered_map>
8
9 namespace netdem {
10
11 class Particle;
12 class ContactPW;
13 class Cell;
14 class DomainManager;
15 class NeighPofW;
16
17 class Wall {
18 public:
19   int id{0};
20   std::string label{"default"};
21
22   Shape *shape;
23   int material_type{0};
24
28   bool enable_rotation{false}, enable_bound_aabb{true};
29
31   Vec3d bound_min, bound_max, bound_disp{0, 0, 0};
32
```

```
34   Vec3d pos{0, 0, 0};
35   Vec4d quaternion{1, 0, 0, 0};
36
38   Vec3d force{0, 0, 0}, moment{0, 0, 0};
39
41   Vec3d vel{0, 0, 0}, spin{0, 0, 0}, vel_spin{0, 0, 0};
42
44   MiniMap<std::string, double> dynamic_properties;
45
47   bool need_update_linked_list{true};
48   VecXT<std::pair<Cell *, int>> linked_cell_list;
49
50   VecXT<NeighPofW> linked_particle_list;
51
52   // to recontruct the reference of contact history
53   VecXT<NeighPofW> contact_pw_ref_table;
54
57   bool need_update_stl_model{false};
58   STLModel stl_model;
59
60 public:
61   Wall();
62   Wall(Shape *const shape);
63   Wall *Clone() const;
64
65   void Init();
66
67   void SetShape(Shape *const shape, bool auto_adapt = false);
68
69   void SetPosition(double pos_x, double pos_y, double pos_z);
70   void SetRodrigues(double angle, double axis_x, double axis_y, double axis_z);
71   void SetQuaternion(double q_0, double q_1, double q_2, double q_3);
72
73   void SetVelocity(double v_x, double v_y, double v_z);
74   void SetSpin(double spin_x, double spin_y, double spin_z);
75
76   // allows velocity varying according to radius Math::Cross spin
77   void SetVelocitySpin(double spin_x, double spin_y, double spin_z);
78
79   // get the velocity of a position within the wall
80   Vec3d GetVelocity(Vec3d const& cnt_pos);
81
82   void AddForce(const Vec3d &f);
83   void AddMoment(const Vec3d &m);
84
85   void AddForceAtomic(const Vec3d &f);
86   void AddMomentAtomic(const Vec3d &m);
87
88   void ClearForce();
89   void ClearMoment();
90
91   void ApplyContactForce(ContactPW const *cnt);
92
93   void UpdateContactForce();
94
95   void UpdateMotion(double timestep);
96
97   void UpdateMotion(const Vec3d &v, const Vec3d &s, double timestep);
98   void UpdateMotion(const Vec3d &dpos, const Vec4d &dquat);
99
100   void UpdateBound();
101
102   void ClearLinkedCells();
103   void ClearLinkedNeighs();
104
105   void BuildContactRef();
106   void ClearContactRef();
107
108   void UpdateLinkedCells(DomainManager *const dm);
109   void UpdateLinkedNeighs(DomainManager *const dm);
110
111   VecXT<ContactPW *> GetContactPWs();
112
113   void UpdateSTLModel();
114
115   void SaveAsVTK(std::string const &filename);
116
117   void Print();
118
119   ~Wall();
120
121 public:
122   // functions for neighbor list and contact reference
123   int FindLinked(Particle *const p);
124   int FindContactRef(Particle *const p);
125 };
126
```

```
127 } // namespace netdem
```

8.267 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape.cpp File Reference

```
#include "shape.hpp"
#include "utils_io.hpp"
```

8.268 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape.hpp File Reference

```
#include "stl_model.hpp"
#include "utils_math.hpp"
#include <iostream>
#include <nlohmann/json.hpp>
#include <string>
```

Classes

- class netdem::Shape

Namespaces

- namespace netdem

8.269 shape.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "stl_model.hpp"
4 #include "utils_math.hpp"
5 #include <iostream>
6 #include <nlohmann/json.hpp>
7 #include <string>
8
9 namespace netdem {
10
22 class Shape {
23 public:
24   enum Type {
25     none,
26     sphere,
27     spherical_harmonics,
28     trimesh,
29     trimesh_convex,
30     ellipsoid,
31     polybezier,
32     triangle,
```

```
33      plane,
34      cylinder,
35      poly_super_ellipsoid,
36      poly_super_quadrics,
37      level_set,
38      num_shapes
39    };
40
41    // for tracking shape
42    int id{0};
43    std::string label{"default"};
44
45    int shape_type{0};
46    std::string shape_name{"shape"};
47
48    // for particle properties. By default, it is assumed that the princial axes
49    // of the shape primitives are align with the coordinate axes.
50    double size{1.0}, volume{0.5236};
51    Mat3d inertia{{{0.05236, 0, 0}, {0, 0.05236, 0}, {0, 0, 0.05236}}};
52
53    // by default, particles are usually ball-like and we use bounding sphere
54    // algorithm for the broad phase contact detection; if particles are seriously
55    // elongated, bounding sphere may result in poor performance; as such, one can
56    // enable the bounding aabb algorithm for efficiency. And, by default, walls
57    // are usually represented by triangle mesh, and thus we use bounding aabb
58    // algorithm for the broad phase contact detection.
59
60    // for broad-phase contact detection: bounding sphere
61    double bound_sphere_radius{1.0}, skin{0.05};
62
63    // for broad-phase contact detection: bounding aabb
64    Vec3d bound_aabb_min{-0.5, -0.5, -0.5}, bound_aabb_max{0.5, 0.5, 0.5};
65
66    // for node-distance-potential contact solver
67    bool use_node{false};
68    int node_num{1000};
69    double node_spacing{sqrt(Math::PI / 1000.0)};
70    VecXT<Vec3d> nodes;
71
72    // for determine contact solvers
73    bool is_convex{true};
74
76    bool use_customized_solver{false};
77
78    // serilization and dd-serilization
79    virtual nlohmann::json PackJson();
80
81    virtual void InitFromJson(nlohmann::json const &js);
82
83    virtual void InitFromJsonFile(std::string const &js_file);
84
85    virtual void Translate(Vec3d const &pos);
86
87    virtual void UpdateNodes();
88
89    virtual void UpdateShapeProperties();
90
91    virtual void SetSize(double d);
92
93    virtual Shape *Clone() const;
94
95    virtual STLModel GetSTLModel(int num_facets = 400);
96
97    virtual void SaveAsVTK(std::string const &filename);
98
99    virtual void SaveAsSTL(std::string const &filename);
100
101    virtual std::tuple<Vec3d, Vec3d> GetBoundAABB(Vec3d const &pos,
102                                                  Vec4d const &quat);
103
104    virtual Vec3d SupportPoint(Vec3d const &dir);
105
106    virtual VecXT<Vec3d> SupportPoints(Vec3d const &dir);
107
108    virtual double SignedDistance(Vec3d const &pos);
109
112    virtual Vec3d SurfacePoint(Vec3d const &pos);
113
114    virtual bool Enclose(Vec3d const &pos);
115
116    virtual void Print();
117
118    virtual ~Shape();
119
120    MatNd<8, 3> GetBoundAABBVertices();
121  };
122
```

```
123 } // namespace netdem
```

8.270 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape_cylinder.cpp File Reference

```
#include "shape_cylinder.hpp"
#include "eigen_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "shape_spherical_harmonics.hpp"
#include "spherical_voronoi.hpp"
#include "stl_model.hpp"
#include "utils_macros.hpp"
#include "utils_math.hpp"
```

8.271 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape_cylinder.hpp File Reference

```
#include "shape.hpp"
#include <iostream>
```

Classes

- class netdem::Cylinder

Namespaces

- namespace netdem

8.272 shape_cylinder.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include <iostream>
5
6 namespace netdem {
7
8 class Cylinder : public Shape {
9 public:
10   double radius{0.5}, height{1.0};
11
12   Cylinder();
13   Cylinder(double r, double h);
14
15   Shape *Clone() const override;
16
```

```
17   nlohmann::json PackJson() override;
18   void InitFromJson(nlohmann::json const &js) override;
19
20   void Init();
21
22   void SetSize(double d) override;
23
24   void UpdateNodes() override;
25   void UpdateShapeProperties() override;
26
27   STLModel GetSTLModel(int num_facets = 400) override;
28
29   Vec3d SupportPoint(Vec3d const &dir) override;
30   VecXT<Vec3d> SupportPoints(Vec3d const &dir) override;
31
32   double SignedDistance(Vec3d const &pos) override;
33   Vec3d SurfacePoint(Vec3d const &pos) override;
34
35   double CalculateRho(Vec3d const &dir);
36
37   void Print() override;
38 };
39
40 } // namespace netdem
```

8.273 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape_ellipsoid.cpp File Reference

```
#include "shape_ellipsoid.hpp"
#include "eigen_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "shape_spherical_harmonics.hpp"
#include "spherical_voronoi.hpp"
#include "stl_model.hpp"
#include "utils_macros.hpp"
#include "utils_math.hpp"
#include "wscvt_sampler.hpp"
```

8.274 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape_ellipsoid.hpp File Reference

```
#include "shape.hpp"
#include <iostream>
```

Classes

- class netdem::Ellipsoid

Namespaces

- namespace netdem

8.275 shape_ellipsoid.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include <iostream>
5
6 namespace netdem {
7
8 class Ellipsoid : public Shape {
9 public:
10   double axis_a{0.5}, axis_b{0.5}, axis_c{0.5};
11
12   Ellipsoid();
13   Ellipsoid(double a, double b, double c);
14
15   Shape *Clone() const override;
16
17   nlohmann::json PackJson() override;
18   void InitFromJson(nlohmann::json const &js) override;
19
20   void Init();
21
22   void UpdateNodes() override;
23   void UpdateShapeProperties() override;
24
25   void SetSize(double d) override;
26
27   STLModel GetSTLModel(int num_facets = 400) override;
28
29   Vec3d SupportPoint(Vec3d const &dir) override;
30   VecXT<Vec3d> SupportPoints(Vec3d const &dir) override;
31
32   double SignedDistance(Vec3d const &pos) override;
33   Vec3d SurfacePoint(Vec3d const &pos) override;
34
35   double CalculateRho(Vec3d const &dir);
36
37   void Print() override;
38 };
39
40 } // namespace netdem
```

8.276 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_factory.cpp File Reference

```
#include "shape_factory.hpp"
#include "shape_cylinder.hpp"
#include "shape_ellipsoid.hpp"
#include "shape_level_set.hpp"
#include "shape_plane.hpp"
#include "shape_poly_super_ellipsoid.hpp"
#include "shape_poly_super_quadrics.hpp"
#include "shape_polybezier.hpp"
#include "shape_sphere.hpp"
#include "shape_spherical_harmonics.hpp"
#include "shape_triangle.hpp"
#include "shape_trimesh.hpp"
#include "utils_io.hpp"
```

8.277 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_factory.hpp File Reference

```
#include "shape.hpp"
#include <nlohmann/json.hpp>
```

Classes

- class netdem::ShapeFactory

Namespaces

- namespace netdem

8.278 shape_factory.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include <nlohmann/json.hpp>
5
6 namespace netdem {
7
8 class ShapeFactory {
9 public:
10   static std::unordered_map<std::string, Shape::Type> shape_map;
11
12   static Shape *NewShape(std::string const &shape_name,
13                          nlohmann::json const &js);
14 };
15
16 } // namespace netdem
```

8.279 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_level_set.cpp File Reference

```
#include "shape_level_set.hpp"
#include "igl_wrapper.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
```

8.280 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_level_set.hpp File Reference

```
#include "igl_wrapper.hpp"
#include "level_set_function.hpp"
#include "shape.hpp"
#include "stl_model.hpp"
#include "stl_reader.hpp"
#include <iostream>
#include <string>
```

Classes

- class netdem::LevelSet

Namespaces

- namespace netdem

8.281 shape_level_set.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "igl_wrapper.hpp"
4 #include "level_set_function.hpp"
5 #include "shape.hpp"
6 #include "stl_model.hpp"
7 #include "stl_reader.hpp"
8 #include <iostream>
9 #include <string>
10
11 namespace netdem {
12
13 class LevelSet : public LevelSetFunction, public Shape {
14 public:
15   LevelSet();
16
17   nlohmann::json PackJson() override;
18   void InitFromJson(nlohmann::json const &js) override;
19
20   void InitFromSTL(std::string const &file, int mesh_res = 25);
21   void InitFromSTL(STLModel const &stl_model, int mesh_res = 25);
22
23   void InitFromDistanceMap(double c_0, double c_1, double c_2, double sp,
24                            const VecXT<VecXT<VecXT<double>>> &dist_map);
25
26   void Init();
27
28   void AlignAxes();
29
30   void UpdateNodes() override;
31   void UpdateShapeProperties() override;
32
33   void SetSize(double d) override;
34
35   Shape *Clone() const override;
36
37   STLModel GetSTLModel(int res = 400) override;
38
39   double SignedDistance(Vec3d const &pos) override;
40   Vec3d SurfacePoint(Vec3d const &pos) override;
41
42   void Print() override;
43 };
44
45 } // namespace netdem
```

8.282 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_plane.cpp File Reference

```
#include "shape_plane.hpp"
#include "stl_model.hpp"
#include "utils_math.hpp"
```

8.283 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_plane.hpp File Reference

```
#include "shape.hpp"
#include <cmath>
#include <iostream>
```

Classes

- class netdem::Plane

Namespaces

- namespace netdem

8.284 shape_plane.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include <cmath>
5 #include <iostream>
6
7 namespace netdem {
8
15 class Plane : public Shape {
16 public:
17   Vec3d center{0, 0, 0}, dir_n{0, 0, 1};
18   double extent{5};
19   // the default extent is deemed large enough to cover the
20   // computing domain; otherwise, enlarge it. Note that the gjk
21   // algorithm would encounter precision issue for two elements with
22   // large size difference (e.g., >1e4 times of size difference)
23
24   Plane();
25   Plane(Vec3d const &c, Vec3d const &n);
26   Plane(double c_x, double c_y, double c_z, double n_x, double n_y, double n_z);
27
28   void UpdateNodes() override;
29   void UpdateShapeProperties() override;
30
31   Shape *Clone() const override;
32
33   nlohmann::json PackJson() override;
34   void InitFromJson(nlohmann::json const &js) override;
```

```
35
36   void Init();
37
38   void SetExtent(double e);
39   void SetCenter(double c_x, double c_y, double c_z);
40   void SetNormal(double n_x, double n_y, double n_z);
41
42   std::tuple<Vec3d, Vec3d> GetBoundAABB(Vec3d const &pos,
43                                         Vec4d const &quat) override;
44
45   STLModel GetSTLModel(int num_facets = 400) override;
46
47   Vec3d SupportPoint(Vec3d const &dir) override;
48   VecXT<Vec3d> SupportPoints(Vec3d const &dir) override;
49
50   double SignedDistance(Vec3d const &pos) override;
51   Vec3d SurfacePoint(Vec3d const &pos) override;
52
53   void Print() override;
54 };
55
56 } // namespace netdem
```

8.285 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_poly_super_ellipsoid.cpp File Reference

```
#include "shape_poly_super_ellipsoid.hpp"
#include "eigen_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "shape_spherical_harmonics.hpp"
#include "spherical_voronoi.hpp"
#include "stl_model.hpp"
#include "utils_macros.hpp"
#include "utils_math.hpp"
#include "wscvt_sampler.hpp"
```

8.286 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_poly_super_ellipsoid.hpp File Reference

```
#include "shape.hpp"
#include <iostream>
```

Classes

- class netdem::PolySuperEllipsoid

Namespaces

- namespace netdem

8.287 shape_poly_super_ellipsoid.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include <iostream>
5
6 namespace netdem {
7
14 class PolySuperEllipsoid : public Shape {
15 public:
16   Vec2d axis_a{0.5, 0.5}, axis_b{0.5, 0.5}, axis_c{0.5, 0.5};
17   double order_ab{1.0}, order_c{1.0};
18
19   // translating and rotating by pos_ref and quat_ref to convert to centroid and
20   // axies aligned shape
21   Vec3d pos_ref{0, 0, 0};
22   Vec4d quat_ref{1, 0, 0, 0}, quat_conj{1, 0, 0, 0};
23
24   PolySuperEllipsoid();
25   PolySuperEllipsoid(double am, double ap, double bm, double bp, double cm,
26                      double cp, double nab, double nc);
27
28   Shape *Clone() const override;
29
30   nlohmann::json PackJson() override;
31   void InitFromJson(nlohmann::json const &js) override;
32
33   void Init();
34
35   void UpdateNodes() override;
36   void UpdateShapeProperties() override;
37
38   void SetSize(double d) override;
39
40   STLModel GetSTLModel(int num_facets = 400) override;
41
42   Vec3d SupportPoint(Vec3d const &dir) override;
43   VecXT<Vec3d> SupportPoints(Vec3d const &dir) override;
44
45   double SignedDistance(Vec3d const &pos) override;
46   Vec3d SurfacePoint(Vec3d const &pos) override;
47
48   Vec3d ParametrizationPoint(Vec3d const &dir);
49
50   // double CalculateRho(Vec3d const& dir);
51
52   void Print() override;
53 };
54
55 } // namespace netdem
```

8.288 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape_poly_super_↩ quadrics.cpp File Reference

```
#include "shape_poly_super_quadrics.hpp"
#include "eigen_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "shape_spherical_harmonics.hpp"
#include "spherical_voronoi.hpp"
#include "stl_model.hpp"
#include "utils_macros.hpp"
#include "utils_math.hpp"
#include "wscvt_sampler.hpp"
```

8.289 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_poly_super_quadrics.hpp File Reference

```
#include "shape.hpp"
#include <iostream>
```

Classes

- class netdem::PolySuperQuadrics

Namespaces

- namespace netdem

8.290 shape_poly_super_quadrics.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include <iostream>
5
6 namespace netdem {
7
13 class PolySuperQuadrics : public Shape {
14 public:
15   Vec2d axis_a{0.5, 0.5}, axis_b{0.5, 0.5}, axis_c{0.5, 0.5};
16   Vec2d order_a{1.0, 1.0}, order_b{1.0, 1.0}, order_c{1.0, 1.0};
17
18   // translating and rotating by pos_ref and quat_ref to convert to centroid and
19   // axies aligned shape
20   Vec3d pos_ref{0, 0, 0};
21   Vec4d quat_ref{1, 0, 0, 0}, quat_conj{1, 0, 0, 0};
22
23   PolySuperQuadrics();
24   PolySuperQuadrics(double am, double ap, double bm, double bp, double cm,
25                     double cp, double nam, double nap, double nbm, double nbp,
26                     double ncm, double ncp);
27
28   Shape *Clone() const override;
29
30   nlohmann::json PackJson() override;
31   void InitFromJson(nlohmann::json const &js) override;
32
33   void Init();
34
35   void UpdateNodes() override;
36   void UpdateShapeProperties() override;
37
38   void SetSize(double d) override;
39
40   STLModel GetSTLModel(int res = 400) override;
41
42   Vec3d SupportPoint(Vec3d const &dir) override;
43   VecXT<Vec3d> SupportPoints(Vec3d const &dir) override;
44
45   double SignedDistance(Vec3d const &pos) override;
46   Vec3d SurfacePoint(Vec3d const &pos) override;
47
48   Vec3d ParametrizationPoint(Vec3d const &dir);
49
50   // double CalculateRho(Vec3d const& dir);
51
52   void Print() override;
53 };
54
55 } // namespace netdem
```

8.291 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_polybezier.cpp File Reference

```
#include "shape_polybezier.hpp"
#include "distribution_uniform.hpp"
#include "eigen_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "spherical_voronoi.hpp"
#include "utils_macros.hpp"
#include "wscvt_sampler.hpp"
#include <cassert>
#include <iostream>
#include <unordered_map>
#include <unordered_set>
```

8.292 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_polybezier.hpp File Reference

```
#include "shape.hpp"
#include "stl_model.hpp"
#include "stl_reader.hpp"
#include <iostream>
#include <string>
```

Classes

- class netdem::Polybezier

Namespaces

- namespace netdem

8.293 shape_polybezier.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include "stl_model.hpp"
5 #include "stl_reader.hpp"
6 #include <iostream>
7 #include <string>
8
9 namespace netdem {
10
11 class Polybezier : public Shape {
12 public:
13   // num_patches have to be even
```

```
14   int num_patches{30}, order{2}; // only second order implemented for now
15
16   // [num of faces] by num of knots (e.g., 6 for 2nd order) by 3
17   VecXT<VecXT<Vec3d>> face_patch_knots_list;
18
19   // [num of faces] by 3 vertices by 3
20   VecXT<VecXT<Vec3d>> face_patch_normals_list;
21
22   // [num of faces*3/2] by num of knots (e.g., 6 for 2nd order) by 3
23   VecXT<VecXT<Vec3d>> edge_patch_knots_list;
24
25   // [num of faces] by 3, used to find the edge patches for a face patch
26   VecXT<Vec3i> linked_edges_list;
27
28   // [6*num_cells^2] by [num of patches]
29   VecXT<VecXT<int>> linked_patches_list;
30   int num_cells{4};
31
32   Polybezier();
33
34   nlohmann::json PackJson() override;
35   void InitFromJson(nlohmann::json const &js) override;
36
37   Shape *Clone() const override;
38
39   void InitByRandom();
40   void InitFromKernelSTL(std::string const &file);
41   void InitFromKernelSTL(STLModel const &stl_model);
42
43   void Init();
44
45   void UpdateShapeProperties() override;
46
47   void SetSize(double d) override;
48
49   STLModel GetSTLModel(int res = 400) override;
50
51   void SaveNormalPatchesSpherical(std::string const &file);
52   void SaveNormalPatchesCubic(std::string const &file);
53
54   Vec3d SupportPoint(Vec3d const &dir) override;
55   VecXT<Vec3d> SupportPoints(Vec3d const &dir) override;
56
57   void Print() override;
58
59 private:
60   void AlignAxes();
61
62   void UpdateLinkedPatches();
63   void UpdataMatDuDv();
64
65   VecXT<VecNT<Vec3d, 3>> mat_du_list, mat_dv_list;
66
67   Vec3d GetEdgeKnot(Vec3d const &v0, Vec3d const &v1, Vec3d const &n01);
68
69   void GetUniqueEdges(VecXT<Vec2i> *const edges,
70                       VecXT<Vec3i> *const linked_list,
71                       STLModel const &stl_model);
72
73   Vec3d GetPatchNormal(Vec3d const &v0, Vec3d const &v1, Vec3d const &v2);
74
75   void SortNormalPatchVertices(VecXT<Vec3d> *const normals);
76
77   VecXT<Vec3d> GetCartesianProject(const VecXT<Vec3d> &normals);
78
79   VecXT<Vec3d> GetCartesianProject(Vec3d const &v1, Vec3d const &v2);
80
81   bool ContainCorner(Vec3d const &corner, const VecXT<Vec3d> &normals);
82
83   STLModel GetSTLModel(const VecXT<Vec3d> &knots, int res);
84
85   int GetSupportPatchID(Vec3d const &dir);
86 };
87
88 } // namespace netdem
```

8.294 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_sphere.cpp File Reference

```
#include "shape_sphere.hpp"
#include "igl_wrapper.hpp"
#include "spherical_voronoi.hpp"
#include "stl_model.hpp"
#include "utils_macros.hpp"
#include "utils_math.hpp"
#include "wscvt_sampler.hpp"
```

8.295 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_sphere.hpp File Reference

```
#include "shape.hpp"
#include <iostream>
```

Classes

- class netdem::Sphere

Namespaces

- namespace netdem

8.296 shape_sphere.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include <iostream>
5
6 namespace netdem {
7
8 class Sphere : public Shape {
9 public:
10   Sphere();
11   Sphere(double d);
12
13   Shape *Clone() const override;
14
15   nlohmann::json PackJson() override;
16   void InitFromJson(nlohmann::json const &js) override;
17
18   void Init();
19
20   void UpdateNodes() override;
21   void UpdateShapeProperties() override;
22
23   STLModel GetSTLModel(int num_facets = 400) override;
```

```
24
25   Vec3d SupportPoint(Vec3d const &dir) override;
26   VecXT<Vec3d> SupportPoints(Vec3d const &dir) override;
27
28   double SignedDistance(Vec3d const &pos) override;
29   Vec3d SurfacePoint(Vec3d const &pos) override;
30
31   void Print() override;
32 };
33
34 } // namespace netdem
```

8.297 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape_spherical_↩ harmonics.cpp File Reference

```
#include "shape_spherical_harmonics.hpp"
#include "eigen_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "spherical_voronoi.hpp"
#include "stl_model.hpp"
#include "stl_reader.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
#include "wscvt_sampler.hpp"
#include <cmath>
#include <iostream>
```

8.298 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape_spherical_↩ harmonics.hpp File Reference

```
#include "shape.hpp"
#include "stl_model.hpp"
#include "stl_reader.hpp"
#include <cmath>
#include <iostream>
#include <string>
```

Classes

- class netdem::SphericalHarmonics

Namespaces

- namespace netdem

8.299 shape_spherical_harmonics.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include "stl_model.hpp"
5 #include "stl_reader.hpp"
6 #include <cmath>
7 #include <iostream>
8 #include <string>
9
10 namespace netdem {
11
12 class SphericalHarmonics : public Shape {
13 public:
14   int degree{8};
15   VecXT<double> a_nm;
16
17   SphericalHarmonics();
18   SphericalHarmonics(int n);
19
20   nlohmann::json PackJson() override;
21   void InitFromJson(nlohmann::json const &js) override;
22
23   void InitFromSTL(std::string const &file);
24   void InitFromSTL(STLModel const &stl_model);
25
26   void Init();
27
28   void UpdateNodes() override;
29
30   void UpdateShapeProperties() override;
31
32   void SetSize(double d) override;
33
34   STLModel GetSTLModel(int res = 400) override;
35
36   Shape *Clone() const override;
37
38   double SignedDistance(Vec3d const &pos) override;
39   Vec3d SurfacePoint(Vec3d const &pos) override;
40
41   // the most basic and straightforward option to calculate SH terms
42   static VecXT<double> CalculateYnm(double theta, double phi, int deg);
43   static VecXT<VecXT<double>> CalculateYnm(const VecXT<double> &theta,
44                                            const VecXT<double> &phi, int deg);
45
46   // a fast option to calculate SH terms, by hard coded the spherical harmonics
47   // legendre and reusing the sin and cos values
48   static VecXT<double> CalculateYnm_Fast(double theta, double phi, int deg);
49   static VecXT<VecXT<double>> CalculateYnm_Fast(const VecXT<double> &theta,
50                                                 const VecXT<double> &phi,
51                                                 int deg);
52
53   double CalculateRho(double theta, double phi);
54
55   // another fast option by using unit VecXT as direction (thus to avoid sin
56   // and cos)
57   static VecXT<double> CalculateYnm_Fast(Vec3d const &dir, int deg);
58   static VecXT<VecXT<double>> CalculateYnm_Fast(const VecXT<Vec3d> &dir_list,
59                                                 int deg);
60
61   double CalculateRho(Vec3d const &dir);
62
63 private:
64   static VecXT<VecXT<double>> sph_legendre_fast(double theta, int deg);
65   static VecXT<VecXT<double>> sph_legendre_fast(Vec3d const &dir, int deg);
66 };
67
68 } // namespace netdem
```

8.300 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/shape/shape_triangle.cpp File Reference

```
#include "shape_triangle.hpp"
#include "stl_model.hpp"
```

```
#include "utils_math.hpp"
```

8.301 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_triangle.hpp File Reference

```
#include "shape.hpp"
#include <cmath>
#include <iostream>
```

Classes

- class netdem::Triangle

Namespaces

- namespace netdem

8.302 shape_triangle.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "shape.hpp"
4 #include <cmath>
5 #include <iostream>
6
7 namespace netdem {
8
9 class Triangle : public Shape {
10 public:
11   Mat3d vertices;
12   Vec3d dir_n{0, 0, 1};
13
14   Triangle();
15   Triangle(Vec3d const &a, Vec3d const &b, Vec3d const &c);
16
17   nlohmann::json PackJson() override;
18   void InitFromJson(nlohmann::json const &js) override;
19
20   void SetVertices(Vec3d const &a, Vec3d const &b, Vec3d const &c);
21
22   void Init();
23
24   void Translate(Vec3d const &pos) override;
25
26   void UpdateNodes() override;
27   void UpdateShapeProperties() override;
28
29   Shape *Clone() const override;
30
31   STLModel GetSTLModel(int num_facets = 400) override;
32
33   virtual std::tuple<Vec3d, Vec3d> GetBoundAABB(Vec3d const &pos,
34                                                 Vec4d const &quat) override;
35
36   Vec3d SupportPoint(Vec3d const &dir) override;
37   VecXT<Vec3d> SupportPoints(Vec3d const &dir) override;
38
39   double SignedDistance(Vec3d const &pos) override;
40   Vec3d SurfacePoint(Vec3d const &pos) override;
41
42   bool Enclose(Vec3d const &pos) override;
43 };
44
45 } // namespace netdem
```

8.303 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_trimesh.cpp File Reference

```
#include "shape_trimesh.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
```

8.304 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/shape/shape_trimesh.hpp File Reference

```
#include "igl_wrapper.hpp"
#include "shape.hpp"
#include "stl_model.hpp"
#include "stl_reader.hpp"
#include <iostream>
#include <string>
```

Classes

- class netdem::TriMesh

Namespaces

- namespace netdem

8.305 shape_trimesh.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "igl_wrapper.hpp"
4 #include "shape.hpp"
5 #include "stl_model.hpp"
6 #include "stl_reader.hpp"
7 #include <iostream>
8 #include <string>
9
10 namespace netdem {
11
12 class TriMesh : public Shape {
13 public:
14   // basic
15   VecXT<Vec3d> vertices;
16   VecXT<Vec3i> facets;
17
18   // hill climb
19   VecXT<VecXT<int>> vertices_neighs;
20
21   // for linked-patches algorithm
22   bool use_linked_patches{false};
23   int num_cells{8}; // should be even number
```

```
24   VecXT<VecXT<int>> linked_vertices;
25
26   // signed distance
27   SDFCalculator sdf_calculator;
28
29   TriMesh();
30
31   nlohmann::json PackJson() override;
32   void InitFromJson(nlohmann::json const &js) override;
33
34   void InitFromSTL(std::string const &file);
35   void InitFromOFF(std::string const &file);
36   void InitFromSTL(STLModel const &stl_model);
37
38   void Init();
39
40   void AlignAxes();
41
42   void Decimate(int num_facets);
43   void MakeConvex();
44
45   void UpdateNodes() override;
46   void UpdateShapeProperties() override;
47
48   void SetSize(double d) override;
49
50   Shape *Clone() const override;
51
52   STLModel GetSTLModel(int num_facets = 400) override;
53
54   Vec3d SupportPoint(Vec3d const &dir) override;
55   VecXT<Vec3d> SupportPoints(Vec3d const &dir) override;
56
57   Vec3d SupportPoint_HillClimb(Vec3d const &dir);
58   VecXT<Vec3d> SupportPoints_HillClimb(Vec3d const &dir);
59   void SupportPoints_HillClimbCheckCoplane(int vert_id, Vec3d const &max_pos,
60                                            VecXT<int> *const vert_id_list,
61                                            Vec3d const &dir);
62
63   Vec3d SupportPoint_Sweep(Vec3d const &dir);
64   VecXT<Vec3d> SupportPoints_Sweep(Vec3d const &dir);
65
66   Vec3d SupportPoint_LinkedVertices(Vec3d const &dir);
67   VecXT<Vec3d> SupportPoints_LinkedVertices(Vec3d const &dir);
68
69   double SignedDistance(Vec3d const &pos) override;
70   Vec3d SurfacePoint(Vec3d const &pos) override;
71
72   int ClosestFacet(Vec3d const &pos);
73
74   void Print() override;
75
76   void SaveNormalPatchesSpherical(std::string const &file);
77   void SaveNormalPatchesCubic(std::string const &file);
78
79 private:
80   void UpdateVerticesNeighs();
81
82   void UpdateLinkedVertices();
83   void UpdateLinkedVerticesSub(int vid);
84
85   VecXT<Vec3d> ComputeNormalPatch(int vid);
86
87   void SortNormalPatchVertices(VecXT<Vec3d> *const normals);
88
89   VecXT<Vec3d> ComputeCartesianProject(const VecXT<Vec3d> &normals);
90
91   VecXT<Vec3d> ComputeCartesianProject(const Vec3d &v1, const Vec3d &v2);
92
93   bool ContainCorner(Vec3d const &corner, const VecXT<Vec3d> &normals);
94
95   bool Find(const VecXT<int> &vert_id_list, int id);
96 };
97
98 } // namespace netdem
```

8.306 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/simulation.cpp File Reference

```
#include "simulation.hpp"
#include <iostream>
```

8.307 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/simulation.hpp File Reference

```
#include "dem_solver.hpp"
#include "domain_manager.hpp"
#include "modifier_manager.hpp"
#include "input_processor.hpp"
#include "mpi_manager.hpp"
#include "scene.hpp"
#include <string>
```

Classes

- class netdem::Simulation

Namespaces

- namespace netdem

8.308 simulation.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "dem_solver.hpp"
4 #include "domain_manager.hpp"
5 #include "modifier_manager.hpp"
6 #include "input_processor.hpp"
7 #include "mpi_manager.hpp"
8 #include "scene.hpp"
9 #include <string>
10
11 namespace netdem {
12
13 class Simulation {
14 public:
16   InputProcessor input_processor;
17
19   DomainManager domain_manager;
20
22   MPIManager mpi_manager;
23
26   Scene scene;
27
29   DEMSolver dem_solver;
```

```
30
34   ModifierManager modifier_manager;
35
37   double mech_time{0};
38
41   int mech_cycles{0};
42
44   bool log_flag{true};
45
46   Simulation();
47
49   void Init();
50
52   // shape info, 0: 0-step json, 2: time-specific json
53   void AutoReadRestart(std::string const &path, int mech_cyc,
54                        int shape_info_case = 0);
55
57   void Run(double time);
58 };
59
60 } // namespace netdem
```

8.309 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/cgal_wrapper.cpp File Reference

```
#include "cgal_wrapper.hpp"
#include "stl_model.hpp"
#include "utils_math.hpp"
#include <CGAL/AABB_face_graph_triangle_primitive.h>
#include <CGAL/AABB_traits.h>
#include <CGAL/AABB_tree.h>
#include <CGAL/Alpha_shape_3.h>
#include <CGAL/Alpha_shape_cell_base_3.h>
#include <CGAL/Alpha_shape_vertex_base_3.h>
#include <CGAL/Compact_container.h>
#include <CGAL/Delaunay_triangulation_3.h>
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/IO/output_to_vtu.h>
#include <CGAL/Mesh_complex_3_in_triangulation_3.h>
#include <CGAL/Mesh_criteria_3.h>
#include <CGAL/Mesh_triangulation_3.h>
#include <CGAL/Modifier_base.h>
#include <CGAL/Object.h>
#include <CGAL/Polygon_mesh_processing/detect_features.h>
#include <CGAL/Polygon_mesh_processing/smooth_mesh.h>
#include <CGAL/Polyhedral_mesh_domain_with_features_3.h>
#include <CGAL/Polyhedron_3.h>
#include <CGAL/Polyhedron_incremental_builder_3.h>
#include <CGAL/Side_of_triangle_mesh.h>
#include <CGAL/Simple_cartesian.h>
#include <CGAL/Surface_mesh.h>
#include <CGAL/Tetrahedron_3.h>
#include <CGAL/Triangulation_3.h>
#include <CGAL/Triangulation_utils_3.h>
#include <CGAL/Unique_hash_map.h>
#include <CGAL/algorithm.h>
#include <CGAL/basic.h>
#include <CGAL/boost/graph/graph_traits_Polyhedron_3.h>
#include <CGAL/boost/graph/helpers.h>
#include <CGAL/internal/Lazy_alpha_nt_3.h>
```

```
#include <CGAL/iterator.h>
#include <CGAL/make_mesh_3.h>
#include <CGAL/refine_mesh_3.h>
#include <set>
#include <unordered_map>
```

Classes

- class PolyhedronBuilder< HDS >

8.310 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/cgal_wrapper.hpp File Reference

```
#include "utils_macros.hpp"
```

Namespaces

- namespace netdem

Functions

- void netdem::cgal_tetmesh (const VecXT< Vec3d > &vv, const VecXT< Vec3i > &ff, VecXT< Vec3d > *const tv, VecXT< Vec4i > *const tt, double mesh_size)
- void netdem::cgal_smooth_mesh (VecXT< Vec3d > *const vv, VecXT< Vec3i > *const ff, int num_iters)
- void netdem::cgal_alpha_shape (VecXT< Vec3d > *vv_out, VecXT< Vec3i > *ff_out, const VecXT< Vec3d > &vv_in, double alpha=0.7)

8.311 cgal_wrapper.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_macros.hpp"
4
5 namespace netdem {
6
7 void cgal_tetmesh(const VecXT<Vec3d> &vv, const VecXT<Vec3i> &ff,
8                   VecXT<Vec3d> *const tv, VecXT<Vec4i> *const tt,
9                   double mesh_size);
10
11 void cgal_smooth_mesh(VecXT<Vec3d> *const vv, VecXT<Vec3i> *const ff,
12                       int num_iters);
13
14 void cgal_alpha_shape(VecXT<Vec3d> *vv_out, VecXT<Vec3i> *ff_out,
15                       const VecXT<Vec3d> &vv_in, double alpha = 0.7);
16
17 } // namespace netdem
```

8.312 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/cork_decls.hpp File Reference

```
#include <cork.h>
#include <mesh.h>
```

Classes

- struct CorkVertex
- struct CorkTriangle

Typedefs

- typedef RawMesh< CorkVertex, CorkTriangle > RawCorkMesh
- typedef Mesh< CorkVertex, CorkTriangle > CorkMesh

8.312.1 Typedef Documentation

8.312.1.1 CorkMesh

```
typedef Mesh<CorkVertex, CorkTriangle> CorkMesh
```

8.312.1.2 RawCorkMesh

```
typedef RawMesh<CorkVertex, CorkTriangle> RawCorkMesh
```

8.313 cork_decls.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include <cork.h>
4 #include <mesh.h>
5
6 // the following is copied from cork.cpp
7 struct CorkTriangle;
8
9 struct CorkVertex : public MinimalVertexData,
10                     public RemeshVertexData,
11                     public IsctVertexData,
12                     public BoolVertexData {
13   void merge(const CorkVertex &v0, const CorkVertex &v1) {
14     double a0 = 0.5;
15     if (v0.manifold && !v1.manifold)
16       a0 = 0.0;
17     if (!v0.manifold && v1.manifold)
18       a0 = 1.0;
19     double a1 = 1.0 - a0;
20
21     pos = a0 * v0.pos + a1 * v1.pos;
22   }
23   void interpolate(const CorkVertex &v0, const CorkVertex &v1) {
24     double a0 = 0.5;
25     double a1 = 0.5;
26     pos = a0 * v0.pos + a1 * v1.pos;
27   }
28
29   void isct(IsctVertEdgeTriInput<CorkVertex, CorkTriangle> input) {
30     Vec2d a_e = Vec2d(1, 1) / 2.0;
31     Vec3d a_t = Vec3d(1, 1, 1) / 3.0;
32     a_e /= 2.0;
33     a_t /= 2.0;
34   }
35   void isct(IsctVertTriTriTriInput<CorkVertex, CorkTriangle> input) {
36     Vec3d a[3];
37     for (uint k = 0; k < 3; k++) {
38       a[k] = Vec3d(1, 1, 1) / 3.0;
39       a[k] /= 3.0;
40     }
41     for (uint i = 0; i < 3; i++) {
42       for (uint j = 0; j < 3; j++) {
43       }
44     }
45   }
46   void isctInterpolate(const CorkVertex &v0, const CorkVertex &v1) {
47     double a0 = len(v1.pos - pos);
48     double a1 = len(v0.pos - pos);
49     if (a0 + a1 == 0.0)
50       a0 = a1 = 0.5; // safety
51     double sum = a0 + a1;
52     a0 /= sum;
53     a1 /= sum;
54   }
55 };
56
57 struct CorkTriangle : public MinimalTriangleData,
58                       public RemeshTriangleData,
59                       public IsctTriangleData,
60                       public BoolTriangleData {
61   void merge(const CorkTriangle &, const CorkTriangle &) {}
62   static void split(CorkTriangle &, CorkTriangle &, const CorkTriangle &) {}
63   void move(const CorkTriangle &) {}
64   void subdivide(SubdivideTriInput<CorkVertex, CorkTriangle> input) {
65     bool_alg_data = input.pt->bool_alg_data;
66   }
67 };
68
69 typedef RawMesh<CorkVertex, CorkTriangle> RawCorkMesh;
70 typedef Mesh<CorkVertex, CorkTriangle> CorkMesh;
```

8.314 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/cork_wrapper.cpp File Reference

```
#include "cork_wrapper.hpp"
#include "cork_decls.hpp"
#include "utils_io.hpp"
#include "utils_macros.hpp"
```

8.315 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/cork_wrapper.hpp File Reference

```
#include "utils_math.hpp"
```

Classes

- class netdem::Cork

Namespaces

- namespace netdem

8.316 cork_wrapper.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_math.hpp"
4
5 namespace netdem {
6
7 class Cork {
8 public:
9   static void MeshIntersect(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
10                             const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
11                             VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab,
12                             VecXT<int> *const jab);
13
14   static void MeshUnion(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
15                         const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
16                         VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab,
17                         VecXT<int> *const jab);
18
19   static void MeshDifference(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
20                              const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
21                              VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab,
22                              VecXT<int> *const jab);
23
24   static void MeshXor(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
25                       const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
26                       VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab,
27                       VecXT<int> *const jab);
28
29   static void MeshIntersect(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
```

```
30                             const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
31                             VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab);
32
33   static void MeshUnion(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
34                         const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
35                         VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab);
36
37   static void MeshDifference(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
38                              const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
39                              VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab);
40
41   static void MeshXor(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
42                       const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
43                       VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab);
44
45   static void MeshIntersect(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
46                             double dist_pc_to_plane, Vec3d const &dir_n,
47                             VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab,
48                             VecXT<int> *const jab);
49 };
50
51 } // namespace netdem
```

8.317 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/distribution.hpp File Reference

```
#include "utils_macros.hpp"
```

Classes

- class netdem::Distribution

Namespaces

- namespace netdem

8.318 distribution.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_macros.hpp"
4
5 namespace netdem {
6
10 class Distribution {
11 public:
12   virtual double Get() = 0;
13   virtual VecXT<double> Get(int num) = 0;
14   virtual ~Distribution() {}
15 };
16
17 } // namespace netdem
```

8.319 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/distribution_uniform.hpp File Reference

```
#include "distribution.hpp"
#include <random>
```

Classes

- class UniformDistribution

8.320 distribution_uniform.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "distribution.hpp"
4 #include <random>
5
6 using namespace netdem;
7
8 class UniformDistribution : public Distribution {
9 public:
11   double bound_min, bound_max;
12
13   UniformDistribution() : bound_min(0), bound_max(1) {
14     std::random_device rd;
15     mt_eng = std::mt19937(rd());
16     real_dist = std::uniform_real_distribution<double>(0, 1);
17   }
18
19   UniformDistribution(double bound_min, double bound_max)
20       : bound_min(bound_min), bound_max(bound_max) {
21     std::random_device rd;
22     mt_eng = std::mt19937(rd());
23     real_dist = std::uniform_real_distribution<double>(bound_min, bound_max);
24   }
25
26   double Get() override { return real_dist(mt_eng); }
27
28   VecXT<double> Get(int num) override {
29     VecXT<double> num_list(num, 0);
30
31     for (int i = 0; i < num; i++) {
32       num_list[i] = real_dist(mt_eng);
33     }
34
35     return num_list;
36   }
37
38 private:
39   std::mt19937 mt_eng;
40   std::uniform_real_distribution<double> real_dist;
41 };
```

8.321 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/eigen_wrapper.cpp File Reference

```
#include "eigen_wrapper.hpp"
#include "utils_math.hpp"
#include <Eigen/Dense>
```

Namespaces

- namespace netdem

Functions

- void netdem::STDToEigen (const VecXT< VecXT< double > > &std_mat, Eigen::MatrixXd *eigen_mat)
- void netdem::STDToEigen (const Mat3d &std_mat, Eigen::Matrix3d *eigen_mat)
- void netdem::STDToEigen (const VecXT< double > &std_vec, Eigen::VectorXd *eigen_vec)
- void netdem::STDToEigen (const Vec3d &std_vec, Eigen::Vector3d *eigen_vec)
- void netdem::EigenToSTD (VecXT< VecXT< double > > *const std_mat, const Eigen::MatrixXd &eigen_↩
mat)
- void netdem::EigenToSTD (Mat3d *const std_mat, const Eigen::Matrix3d &eigen_mat)
- void netdem::EigenToSTD (VecXT< double > *const std_vec, const Eigen::VectorXd &eigen_vec)
- void netdem::EigenToSTD (Vec3d *const std_vec, const Eigen::Vector3d &eigen_vec)
- Mat3d netdem::EigenVector (const Mat3d &mat)
- VecXT< double > netdem::EigenSolve (const VecXT< VecXT< double > > &a, const VecXT< double > &b)
- Vec3d netdem::EigenSolve (Mat3d const &a, const Vec3d &b)

8.322 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/eigen_wrapper.hpp File Reference

```
#include "utils_math.hpp"
```

Namespaces

- namespace netdem

Functions

- Mat3d netdem::EigenVector (const Mat3d &mat)
- VecXT< double > netdem::EigenSolve (const VecXT< VecXT< double > > &a, const VecXT< double > &b)
- Vec3d netdem::EigenSolve (Mat3d const &a, const Vec3d &b)

8.323 eigen_wrapper.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_math.hpp"
4
5 namespace netdem {
6
7 Mat3d EigenVector(const Mat3d &mat);
8
9 VecXT<double> EigenSolve(const VecXT<VecXT<double>> &a, const VecXT<double> &b);
10
11 Vec3d EigenSolve(Mat3d const &a, const Vec3d &b);
12
13 } // namespace netdem
```

8.324 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/igl_wrapper.cpp File Reference

```
#include "igl_wrapper.hpp"
#include "eigen_wrapper.hpp"
#include "utils_macros.hpp"
#include <Eigen/Dense>
#include <igl/bfs_orient.h>
#include <igl/copyleft/cgal/convex_hull.h>
#include <igl/copyleft/cgal/mesh_boolean.h>
#include <igl/piecewise_constant_winding_number.h>
#include <igl/copyleft/cgal/points_inside_component.h>
#include <igl/boundary_facets.h>
#include <igl/decimate.h>
#include <igl/default_num_threads.h>
#include <igl/facet_components.h>
#include <igl/loop.h>
#include <igl/marching_cubes.h>
#include <igl/per_edge_normals.h>
#include <igl/per_face_normals.h>
#include <igl/per_vertex_normals.h>
#include <igl/point_mesh_squared_distance.h>
#include <igl/remove_duplicate_vertices.h>
#include <igl/remove_unreferenced.h>
#include <iostream>
```

Namespaces

- namespace netdem

Functions

- void netdem::STDToEigen (const VecXT< VecXT< double > > &std_mat, Eigen::MatrixXd *eigen_mat)
- void netdem::STDToEigen (const VecXT< Vec3d > &std_mat, Eigen::MatrixXd *eigen_mat)
- void netdem::STDToEigen (const Mat3d &std_mat, Eigen::Matrix3d *eigen_mat)
- void netdem::STDToEigen (const VecXT< Vec3i > &std_mat, Eigen::MatrixXi *eigen_mat)
- void netdem::STDToEigen (const VecXT< Vec4i > &std_mat, Eigen::MatrixXi *eigen_mat)
- void netdem::STDToEigen (const VecXT< double > &std_vec, Eigen::VectorXd *eigen_vec)
- void netdem::STDToEigen (const Vec3d &std_vec, Eigen::Vector3d *eigen_vec)
- void netdem::EigenToSTD (VecXT< VecXT< double > > *const std_mat, const Eigen::MatrixXd &eigen_↩ mat)
- void netdem::EigenToSTD (VecXT< Vec3d > *const std_mat, const Eigen::MatrixXd &eigen_mat)
- void netdem::EigenToSTD (Mat3d *const std_mat, const Eigen::Matrix3d &eigen_mat)
- void netdem::EigenToSTD (VecXT< Vec3i > *const std_mat, const Eigen::MatrixXi &eigen_mat)
- void netdem::EigenToSTD (VecXT< Vec4i > *const std_mat, const Eigen::MatrixXi &eigen_mat)
- void netdem::EigenToSTD (VecXT< int > *const std_vec, const Eigen::VectorXi &eigen_vec)
- void netdem::EigenToSTD (VecXT< double > *const std_vec, const Eigen::VectorXd &eigen_vec)
- void netdem::EigenToSTD (Vec3d *const std_vec, const Eigen::Vector3d &eigen_vec)
- Mat3d netdem::EigenVector (const Mat3d &mat)
- void netdem::igl_remove_unreferenced_vertices (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f)
- void netdem::igl_remove_duplicate_vertices (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f)

- void netdem::igl_remove_duplicate_vertices (VecXT< Vec3d > *const v)
- void netdem::igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- void netdem::igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- void netdem::igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, double dist_pc_to_plane, Vec3d const &dir_n, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- void netdem::igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, double dist_pc_to_plane, Vec3d const &dir_n, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- void netdem::igl_mesh_refine (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f, int num_refines)
- void netdem::igl_mesh_decimate (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f, int num_facets)
- int netdem::igl_facet_components (const VecXT< Vec3i > &fi, VecXT< int > *const fc)
- void netdem::igl_reorient_facets (const VecXT< Vec3d > &v, VecXT< Vec3i > *f)
- bool netdem::igl_check_winding (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f)
- void netdem::igl_convex_hull (const VecXT< Vec3d > &v0, VecXT< Vec3d > *const v1, VecXT< Vec3i > *const f1)
- void netdem::igl_tetmesh_boundary (const VecXT< Vec4i > &tt, VecXT< Vec3i > *const ff, VecXT< int > *const fj)
- void netdem::igl_tetmesh_boundary (const VecXT< Vec4i > &tt, VecXT< Vec3i > *const ff)
- VecXT< int > netdem::igl_points_inside_mesh (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f, const VecXT< Vec3d > &v_querry)
- void netdem::igl_marching_cubes (VecXT< Vec3d > *const vv, VecXT< Vec3i > *const ff, VecXT< VecXT< VecXT< double > > > const &sdf, Vec3d const &corner, Vec3d const &spacing, double iso_value)

8.325 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/igl_wrapper.hpp File Reference

```
#include "stl_model.hpp"
#include "utils_math.hpp"
#include <Eigen/Dense>
#include <igl/signed_distance.h>
```

Classes

- class netdem::SDFCalculator

Namespaces

- namespace netdem

Functions

- void netdem::igl_remove_unreferenced_vertices (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f)
- void netdem::igl_remove_duplicate_vertices (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f)
- void netdem::igl_remove_duplicate_vertices (VecXT< Vec3d > *const v)
- void netdem::igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- void netdem::igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, const VecXT< Vec3d > &vb, const VecXT< Vec3i > &fb, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- void netdem::igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, double dist_↩ pc_to_plane, Vec3d const &dir_n, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab, VecXT< int > *const jab)
- void netdem::igl_mesh_intersect (const VecXT< Vec3d > &va, const VecXT< Vec3i > &fa, double dist_↩ pc_to_plane, Vec3d const &dir_n, VecXT< Vec3d > *const vab, VecXT< Vec3i > *const fab)
- void netdem::igl_mesh_refine (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f, int num_refines)
- void netdem::igl_mesh_decimate (VecXT< Vec3d > *const v, VecXT< Vec3i > *const f, int num_facets)
- int netdem::igl_facet_components (const VecXT< Vec3i > &fi, VecXT< int > *const fc)
- void netdem::igl_reorient_facets (const VecXT< Vec3d > &v, VecXT< Vec3i > *f)
- bool netdem::igl_check_winding (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f)
- void netdem::igl_convex_hull (const VecXT< Vec3d > &v0, VecXT< Vec3d > *const v1, VecXT< Vec3i > *const f1)
- void netdem::igl_tetmesh_boundary (const VecXT< Vec4i > &tt, VecXT< Vec3i > *const ff, VecXT< int > *const fj)
- void netdem::igl_tetmesh_boundary (const VecXT< Vec4i > &tt, VecXT< Vec3i > *const ff)
- void netdem::igl_marching_cubes (VecXT< Vec3d > *const vv, VecXT< Vec3i > *const ff, VecXT< VecXT< VecXT< double > > > const &sdf, Vec3d const &corner, Vec3d const &spacing, double iso_value)
- VecXT< int > netdem::igl_points_inside_mesh (const VecXT< Vec3d > &v, const VecXT< Vec3i > &f, const VecXT< Vec3d > &v_querry)

8.326 igl_wrapper.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "stl_model.hpp"
4 #include "utils_math.hpp"
5 #include <Eigen/Dense>
6 #include <igl/signed_distance.h>
7
8 namespace netdem {
9
10 void igl_remove_unreferenced_vertices(VecXT<Vec3d> *const v,
11                                       VecXT<Vec3i> *const f);
12
13 void igl_remove_duplicate_vertices(VecXT<Vec3d> *const v,
14                                    VecXT<Vec3i> *const f);
15
16 void igl_remove_duplicate_vertices(VecXT<Vec3d> *const v);
17
18 void igl_mesh_intersect(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
19                         const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
20                         VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab,
21                         VecXT<int> *const jab);
22
23 void igl_mesh_intersect(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
24                         const VecXT<Vec3d> &vb, const VecXT<Vec3i> &fb,
25                         VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab);
26
27 void igl_mesh_intersect(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
28                         double dist_pc_to_plane, Vec3d const &dir_n,
29                         VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab,
30                         VecXT<int> *const jab);
31
32 void igl_mesh_intersect(const VecXT<Vec3d> &va, const VecXT<Vec3i> &fa,
33                         double dist_pc_to_plane, Vec3d const &dir_n,
```

```
34                         VecXT<Vec3d> *const vab, VecXT<Vec3i> *const fab);
35
36 void igl_mesh_refine(VecXT<Vec3d> *const v, VecXT<Vec3i> *const f,
37                       int num_refines);
38
39 void igl_mesh_decimate(VecXT<Vec3d> *const v, VecXT<Vec3i> *const f,
40                        int num_facets);
41
42 int igl_facet_components(const VecXT<Vec3i> &fi, VecXT<int> *const fc);
43
44 void igl_reorient_facets(const VecXT<Vec3d> &v, VecXT<Vec3i> *const f);
45
46 bool igl_check_winding(const VecXT<Vec3d> &v, const VecXT<Vec3i> &f);
47
48 void igl_convex_hull(const VecXT<Vec3d> &v0, VecXT<Vec3d> *const v1,
49                      VecXT<Vec3i> *const f1);
50
51 void igl_tetmesh_boundary(const VecXT<Vec4i> &tt, VecXT<Vec3i> *const ff,
52                           VecXT<int> *const fj);
53
54 void igl_tetmesh_boundary(const VecXT<Vec4i> &tt, VecXT<Vec3i> *const ff);
55
56 void igl_marching_cubes(VecXT<Vec3d> *const vv, VecXT<Vec3i> *const ff,
57                         VecXT<VecXT<VecXT<double»> const &sdf,
58                         Vec3d const &corner, Vec3d const &spacing,
59                         double iso_value = 0);
60
61 // need tetgen library, not added yet
62 // void igl_tetmesh(const VecXT<Vec3d> &vv,
63 //                          const VecXT<Vec3i> &ff,
64 //                          VecXT<Vec3d> *const tv,
65 //                          VecXT< Vec4i> *const tt,
66 //                          VecXT<Vec3i> *const tf);
67
68 VecXT<int> igl_points_inside_mesh(const VecXT<Vec3d> &v, const VecXT<Vec3i> &f,
69                                   const VecXT<Vec3d> &v_querry);
70
71 // reference: igl tutorial 704
72 class SDFCalculator {
73 public:
74   double max_distance{0.0};
75
76   SDFCalculator();
77
78   void InitFromSTL(const VecXT<Vec3d> &vv, const VecXT<Vec3i> &ff);
79
80   void InitFromSTL(STLModel const &stl_model);
81
82   void Init();
83
84   double SignedDistance(Vec3d const &pos) const;
85   Vec3d SurfacePoint(Vec3d const &pos) const;
86   int ClosestFacet(Vec3d const &pos) const;
87
88 private:
89   Eigen::MatrixXd vertices;
90   Eigen::MatrixXi facets;
91
92   Eigen::MatrixXi T;
93
94   igl::AABB<Eigen::MatrixXd, 3> tree;
95
96   Eigen::MatrixXd FN, VN, EN;
97   Eigen::MatrixXi E;
98   Eigen::VectorXi EMAP;
99 };
100
101 } // namespace netdem
```

8.327 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/level_set_function.cpp File Reference

```
#include "level_set_function.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
```

8.328 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/level_set_function.hpp File Reference

```
#include "igl_wrapper.hpp"
```

Classes

- class netdem::LevelSetFunction

Namespaces

- namespace netdem

8.329 level_set_function.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "igl_wrapper.hpp"
4
5 namespace netdem {
6
7 class LevelSetFunction {
8 public:
9   // assuming same spacing in each dimension
10   Vec3d corner{-0.5, -0.5, -0.5};
11   double spacing{0.05};
12   Vec3i dim{21, 21, 21};
13
14   VecXT<VecXT<VecXT<double>>> signed_distance_table;
15
16   LevelSetFunction();
17
18   void SetCorner(double corner_x, double corner_y, double corner_z);
19   void SetSpacing(double sp);
20   void SetDimension(double dim_x, double dim_y, double dim_z);
21
22   void InitFromSDFCalculator(const SDFCalculator &sdf_calculator);
23
24   double SignedDistance(Vec3d const &pos);
25
26   Vec3d GradientInterpolate(Vec3d const &pos);
27
28   Vec3d GradientMinus(int i, int j, int k);
29   Vec3d GradientPlus(int i, int j, int k);
30
31   void Reinitialization(int iter, double dt);
32
33   void Reinitialization();
34 };
35
36 } // namespace netdem
```

8.330 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/mini_map.hpp File Reference

```
#include "utils_io.hpp"
#include "utils_macros.hpp"
```

Classes

- struct netdem::my_pair< T_key, T_val >
- class netdem::MiniMap< T_key, T_val >

Namespaces

- namespace netdem

8.331 mini_map.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_io.hpp"
4 #include "utils_macros.hpp"
5
6 namespace netdem {
7
8 template <typename T_key, typename T_val> struct my_pair {
9 public:
10   T_key first;
11   T_val second;
12
13   my_pair() {}
14   my_pair(const T_key &key, const T_key &val) : first(key), second(val) {}
15 };
16
17 template <typename T_key, typename T_val> class MiniMap {
18 public:
19   const my_pair<T_key, T_val> *begin() const { return &(pair_list.front()); }
20
21   my_pair<T_key, T_val> *begin() { return &(pair_list.front()); }
22
23   const my_pair<T_key, T_val> *end() const { return &(pair_list.back()) + 1; }
24
25   my_pair<T_key, T_val> *end() { return &(pair_list.back()) + 1; }
26
27   void erase(my_pair<T_key, T_val> *it) {
28     *it = pair_list.back();
29     pair_list.pop_back();
30   }
31
32   void erase(const T_key &key) {
33     auto it = find(key);
34     erase(it);
35   }
36
37   const my_pair<T_key, T_val> *find(const T_key &key) const {
38     for (auto &pair_item : pair_list) {
39       if (pair_item.first == key) {
40         return &pair_item;
41       }
42     }
43     return end();
44   }
45
46   my_pair<T_key, T_val> *find(const T_key &key) {
47     for (auto &pair_item : pair_list) {
48       if (pair_item.first == key) {
49         return &pair_item;
50       }
51     }
52     return end();
53   }
54
55   int size() const { return pair_list.size(); };
56
57   int size() { return pair_list.size(); };
58
59   const T_val &operator[](const T_key &key) const {
60     auto it = find(key);
61     if (it != end()) {
62       return it->second;
63     } else {
64       PrintError("in MiniMap[], key not exist");
```

```
65       return pair_list.back().second;
66     }
67   }
68
69   T_val &operator[](const T_key &key) {
70     auto it = find(key);
71     if (it != end()) {
72       return it->second;
73     } else {
74       pair_list.emplace_back();
75       pair_list.back().first = key;
76       return pair_list.back().second;
77     }
78   }
79
80   void clear() { pair_list.clear(); }
81
82 private:
83   VecXT<my_pair<T_key, T_val>> pair_list;
84 };
85
86 } // namespace netdem
```

8.332 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/spherical_voronoi.cpp File Reference

```
#include "spherical_voronoi.hpp"
#include "distribution_uniform.hpp"
#include "igl_wrapper.hpp"
#include "shape_spherical_harmonics.hpp"
#include "utils_io.hpp"
#include "utils_math.hpp"
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
```

8.333 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/spherical_voronoi.hpp File Reference

```
#include "utils_macros.hpp"
#include <string>
```

Classes

- class netdem::SphericalVoronoi

Namespaces

- namespace netdem

8.334 spherical_voronoi.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_macros.hpp"
4 #include <string>
5
6 namespace netdem {
7
8 class SphericalVoronoi {
9 public:
11   static std::tuple<VecXT<Vec3d>, VecXT<VecXT<int>>
12   Solve(VecXT<Vec3d> const &vt_seeds);
13
15   static std::tuple<VecXT<Vec3d>, VecXT<VecXT<int>>
16   Solve(VecXT<Vec3d> const &vt_seeds, VecXT<double> const &vt_weights);
17
19   static VecXT<Vec3d> Solve(int num_seeds, int max_iter = 10000,
20                             double tol = 1.0e-4);
21
24   static VecXT<Vec3d> Solve(int num_seeds, VecXT<double> const &weights_sh_coff,
25                             int max_iter, double tol);
26
28   static void SaveAsVTK(std::string const &file, VecXT<Vec3d> const &vt_nodes,
29                         VecXT<VecXT<int>> const &vt_cells,
30                         VecXT<Vec3d> const &vt_seeds);
31
32 private:
33   static int Find(VecXT<int> const &ids, int id);
34   static int Find(Vec3i const &ids, int id);
35
36   static bool IsSharingEdge(Vec3i const &facet_i, Vec3i const &facet_j);
37
38   static VecXT<int> FacetsContainVertex(VecXT<Vec3i> const &facets, int vid);
39
40   static Vec3d WeightedMiddle(Vec3d const &v1, Vec3d const &v2, double w1,
41                               double w2);
42
43   static Vec3d LineIntersection(Vec3d const &v1, Vec3d const &n1,
44                                 Vec3d const &v2, Vec3d const &n2);
45
46   static Vec3d WeightedCentroid(VecXT<Vec3d> const &vertices,
47                                 VecXT<double> const &weights,
48                                 Vec3i const &facet);
49
50   static Vec3d PolyCentroid(VecXT<Vec3d> const &verts, VecXT<int> const &facet);
51 };
52
53 } // namespace netdem
```

8.335 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/stl_model.cpp File Reference

```
#include "stl_model.hpp"
#include "cgal_wrapper.hpp"
#include "eigen_wrapper.hpp"
#include "igl_wrapper.hpp"
#include "stl_reader.hpp"
#include "utils_io.hpp"
#include "utils_macros.hpp"
#include "utils_math.hpp"
#include <cstring>
#include <fstream>
#include <iostream>
#include <memory>
#include <sstream>
#include <string>
```

8.336 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/stl_model.hpp File Reference

```
#include "utils_math.hpp"
```

Classes

- class netdem::STLModel

Namespaces

- namespace netdem

8.337 stl_model.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_math.hpp"
4
5 namespace netdem {
6
7 class STLModel {
8 public:
10   VecXT<Vec3d> vertices;
11
14   VecXT<Vec3i> facets;
15
16   STLModel();
17   STLModel(const VecXT<Vec3d> &vv, const VecXT<Vec3i> &ff);
18
19   void InitFromSTL(std::string const &file);
20   void InitFromOFF(std::string const &file);
21
22   void Translate(Vec3d const &disp);
23   void Rotate(Vec4d const &quat);
24
25   void SaveAsVTK(std::string const &file) const;
26   void SaveAsSTL(std::string const &file) const;
27
28   void RemoveUnreferencedVertices();
29   void RemoveDuplicateVertices();
30
31   void ReorientFacets();
32   void Decimate(int num_facets);
33   void Standardize();
34   void SetSize(double size);
35   void MakeConvex();
36
37   void Refine(int num_refines = 1);
38   void SmoothMesh(int num_trials = 1);
39
40   void MergeSTLModel(STLModel const &stl_model);
41
43   VecXT<int> GetTriangleStrips() const;
44
45   bool IsFaceOutside(bool flip_outside = true);
46
47   bool IsConvex();
48
50   bool Enclose(Vec3d const &pos) const;
51
53   void Print();
54
55   // return: bound_min, bound_max
56   std::tuple<Vec3d, Vec3d> GetBoundAABB() const;
```

```
57
58   Vec3d GetCenter() const;
59   double GetSurfaceArea() const;
60   double GetVolume() const;
61
62   // returns the intertia with respect to the centroid, please use with caution
63   Mat3d GetInertia() const;
64
65   static Vec3d GetCenter(const VecXT<Vec3d> &v, const VecXT<Vec3i> &f);
66   static double GetSurfaceArea(const VecXT<Vec3d> &v, const VecXT<Vec3i> &f);
67   static double GetVolume(const VecXT<Vec3d> &v, const VecXT<Vec3i> &f);
68
69   // returns the intertia with respect to the centroid, please use with caution
70   static Mat3d GetInertia(const VecXT<Vec3d> &v, const VecXT<Vec3i> &f);
71
72   static bool IsConvex(const VecXT<Vec3d> &v, const VecXT<Vec3i> &f);
73
74 private:
76   int VertexIndexInFacet(const Vec3d &facet, int vertex_id);
77 };
78
79 } // namespace netdem
```

8.338 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/stl_reader.cpp File Reference

```
#include "stl_reader.hpp"
#include "stl_model.hpp"
#include "utils_macros.hpp"
#include <algorithm>
#include <cstring>
#include <fstream>
#include <iostream>
#include <memory>
#include <sstream>
#include <string>
```

8.339 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/stl_reader.hpp File Reference

```
#include <string>
```

Classes

- class netdem::STLReader

Namespaces

- namespace netdem

8.340 stl_reader.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include <string>
4
5 namespace netdem {
6
7 class STLModel;
8
9 class STLReader {
10 public:
11   static STLModel ReadFile(std::string const &filename);
12
13 private:
14   static bool IsASCII(std::string const &filename);
15   static STLModel ReadASCII(const char *buffer);
16   static STLModel ReadBinary(const char *buffer);
17   static int cpyint(const char *&p);
18   static double cpydouble(const char *&p);
19 };
20
21 } // namespace netdem
```

8.341 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/utils_io.cpp File Reference

```
#include "utils_io.hpp"
#include "utils_math.hpp"
#include <fstream>
#include <iostream>
#include <sstream>
```

8.342 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/utils_io.hpp File Reference

```
#include "utils_macros.hpp"
#include <fstream>
#include <iostream>
#include <sstream>
```

Namespaces

- namespace netdem

Functions

- void netdem::PrintWarning (std::string const &info)
- void netdem::PrintError (std::string const &info)
- void netdem::PrintDebug (std::string const &info)
- std::string netdem::my_to_string (int value)
- std::string netdem::my_to_string (double value)
- VecXT< VecXT< double > > netdem::ImportDataTxtToVec (std::string const &filename, int lines_to_skip=0)
- bool netdem::FileExist (std::string const &filename)

8.343 utils_io.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_macros.hpp"
4 #include <fstream>
5 #include <iostream>
6 #include <sstream>
7
8 namespace netdem {
9
10 inline void PrintWarning(std::string const &info) {
11   std::cout « "warning: " « info « std::endl;
12 }
13
14 inline void PrintError(std::string const &info) {
15   std::cout « "error: " « info « std::endl;
16   abort();
17 }
18
19 inline void PrintDebug(std::string const &info) {
20   std::cout « "debug: " « info « std::endl;
21 }
22
23 inline std::string my_to_string(int value) {
24   std::stringstream ss;
25   ss « std::internal « value;
26   return ss.str();
27 }
28
29 inline std::string my_to_string(double value) {
30   std::stringstream ss;
31   ss « std::scientific « value;
32   return ss.str();
33 }
34
35 VecXT<VecXT<double» ImportDataTxtToVec(std::string const &filename,
36                                        int lines_to_skip = 0);
37
38 bool FileExist(std::string const &filename);
39
40 } // namespace netdem
```

8.344 /Users/lzhshou/Documents/Research/myProjects/dem_developments/net_dem/netdem/src/utils/utils_macros.hpp File Reference

```
#include <array>
#include <cmath>
#include <tuple>
#include <vector>
```

Classes

- struct netdem::pair_hash

Namespaces

- namespace netdem
- namespace netdem::Math

Typedefs

- using netdem::size_t = std::size_t
- using netdem::Vec2i = std::array< int, 2 >
- using netdem::Vec3i = std::array< int, 3 >
- using netdem::Vec4i = std::array< int, 4 >
- using netdem::Vec2d = std::array< double, 2 >
- using netdem::Vec3d = std::array< double, 3 >
- using netdem::Vec4d = std::array< double, 4 >
- using netdem::Mat2d = std::array< std::array< double, 2 >, 2 >
- using netdem::Mat3d = std::array< std::array< double, 3 >, 3 >
- template<size_t N>
  using netdem::VecNi = std::array< int, N >
- template<size_t N>
  using netdem::VecNd = std::array< double, N >
- template<size_t Nr, size_t Nc>
  using netdem::MatNd = std::array< std::array< double, Nc >, Nr >
- template<typename T >
  using netdem::VecXT = std::vector< T >
- template<typename T , size_t N>
  using netdem::VecNT = std::array< T, N >

Variables

- constexpr double netdem::Math::PI = 3.14159265358979323846264338327950288419 71
- constexpr double netdem::Math::Infinity = 1.0e15

8.345 utils_macros.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include <array>
4 #include <cmath>
5 #include <tuple>
6 #include <vector>
7
8 namespace netdem {
9
10 using size_t = std::size_t;
11
12 using Vec2i = std::array<int, 2>;
13 using Vec3i = std::array<int, 3>;
14 using Vec4i = std::array<int, 4>;
15
16 using Vec2d = std::array<double, 2>;
17 using Vec3d = std::array<double, 3>;
18 using Vec4d = std::array<double, 4>;
19
20 using Mat2d = std::array<std::array<double, 2>, 2>;
21 using Mat3d = std::array<std::array<double, 3>, 3>;
22
23 template <size_t N> using VecNi = std::array<int, N>;
24 template <size_t N> using VecNd = std::array<double, N>;
25 template <size_t Nr, size_t Nc> using MatNd = std::array<std::array<double, Nc>, Nr>;
26
27 template <typename T> using VecXT = std::vector<T>;
28 template <typename T, size_t N> using VecNT = std::array<T, N>;
29
30 namespace Math {
31
32 constexpr double PI = 3.14159265358979323846264338327950288419 71;
33 constexpr double Infinity = 1.0e15;
34
35 } // namespace Math
```

```
36
37 struct pair_hash {
38   template <class T1, class T2>
39   int operator()(const std::pair<T1, T2> &p) const {
40     auto h1 = std::hash<T1>{}(p.first);
41     auto h2 = std::hash<T2>{}(p.second);
42
43     return h1 ^ h2;
44   }
45 };
46
47 } // namespace netdem
```

8.346 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/utils_math.cpp File Reference

```
#include "utils_math.hpp"
```

8.347 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/utils_math.hpp File Reference

```
#include "utils_macros.hpp"
#include <cmath>
#include <ctime>
#include <iostream>
#include <tuple>
```

Namespaces

- namespace netdem
- namespace netdem::Math
- namespace netdem::Math::Quaternion

Functions

- std::ostream & netdem::operator<< (std::ostream &os, Vec3i const &obj)
- std::ostream & netdem::operator<< (std::ostream &os, Vec3d const &obj)
- std::ostream & netdem::operator<< (std::ostream &os, Vec4d const &obj)
- std::ostream & netdem::operator<< (std::ostream &os, Mat3d const &obj)
- Vec3d netdem::operator+ (Vec3d const &lhs, double rhs)
- Vec3d netdem::operator+ (double lhs, Vec3d const &rhs)
- Vec3i netdem::operator+ (Vec3i const &lhs, int rhs)
- Vec3i netdem::operator+ (int lhs, Vec3i const &rhs)
- Vec3d netdem::operator- (Vec3d const &lhs, double rhs)
- Vec3d netdem::operator- (double lhs, Vec3d const &rhs)
- Vec3d netdem::operator∗ (Vec3d const &lhs, double rhs)
- Vec3d netdem::operator∗ (double lhs, Vec3d const &rhs)

- Vec3d netdem::operator/ (Vec3d const &lhs, double rhs)
- Vec3d netdem::operator/ (double lhs, Vec3d const &rhs)
- Vec3d netdem::operator+ (Vec3d const &lhs, Vec3d const &rhs)
- Vec3d netdem::operator- (Vec3d const &lhs, Vec3d const &rhs)
- Vec3d netdem::operator∗ (Vec3d const &lhs, Vec3d const &rhs)
- Vec3d netdem::operator/ (Vec3d const &lhs, Vec3d const &rhs)
- template<typename T >
  int netdem::Math::Sign (T val)
- double netdem::Math::NormLen (Vec2d const &val)
- double netdem::Math::NormLen (Vec3d const &val)
- double netdem::Math::NormLen (double val_0, double val_1)
- double netdem::Math::NormLen (double val_0, double val_1, double val_2)
- double netdem::Math::NormLen (double val_0, double val_1, double val_2, double val_3)
- double netdem::Math::Determinant (Mat2d const &mat)
- double netdem::Math::Determinant (Mat3d const &mat)
- Mat2d netdem::Math::Inverse (Mat2d const &m_val)
- Mat3d netdem::Math::Inverse (Mat3d const &m_val)
- template<size_t r, size_t cr, size_t c>
  MatNd< r, c > netdem::Math::Dot (MatNd< r, cr > const &m_1, MatNd< cr, c > const &m_2)
- template<size_t r, size_t cr, size_t c>
  MatNd< r, c > netdem::Math::DotTransportLHS (MatNd< cr, r > const &m_1, MatNd< cr, c > const &m_2)
- template<size_t r, size_t cr, size_t c>
  MatNd< r, c > netdem::Math::DotTransportRHS (MatNd< r, cr > const &m_1, MatNd< c, cr > const &m_2)
- Vec3d netdem::Math::Cross (Vec3d const &val_1, Vec3d const &val_2)
- double netdem::Math::Dot (Vec3d const &val_1, Vec3d const &val_2)
- double netdem::Math::Dot (VecXT< double > const &val_1, VecXT< double > const &val_2)
- void netdem::Math::Normalize (Vec3d ∗const val)
- Vec4d netdem::Math::Quaternion::FromRodrigues (double rot_angle, Vec3d const &rot_axis)
- std::tuple< double, Vec3d > netdem::Math::Quaternion::ToRodrigues (Vec4d const &quat)
- Vec4d netdem::Math::Quaternion::FromMatrix (Mat3d const &rot_mat)
- Mat3d netdem::Math::Quaternion::ToMatrix (Vec4d const &quat)
- Vec4d netdem::Math::Quaternion::Multiply (Vec4d const &p, Vec4d const &q)
- Vec4d netdem::Math::Quaternion::Add (Vec4d const &p, Vec4d const &q)
- Vec4d netdem::Math::Quaternion::Conjugate (Vec4d const &p)
- void netdem::Math::Quaternion::Normalize (Vec4d ∗const q)
- Vec3d netdem::Math::Rotate (Vec3d const &val_old, double rot_angle_cos, double rot_angle_sin, Vec3d const &rot_axis)
- Vec3d netdem::Math::Rotate (Vec3d const &val_old, double rot_angle, Vec3d const &rot_axis)
- Vec3d netdem::Math::Rotate (Vec3d const &val_old, Vec4d const &quat)
- Vec3d netdem::Math::Rotate (Vec3d const &val_old, Mat3d const &rot_mat)
- Vec3d netdem::Math::CartesianToSpherical (Vec3d const &vert_cart)
- Vec3d netdem::Math::SphericalToCartesian (Vec3d const &vert_sph)

8.348 utils_math.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "utils_macros.hpp"
4 #include <cmath>
5 #include <ctime>
6 #include <iostream>
7 #include <tuple>
8
9 namespace netdem {
10
11 inline std::ostream &operator«(std::ostream &os, Vec3i const &obj) {
```

```
12   os « obj[0] « ", " « obj[1] « ", " « obj[2];
13   return os;
14 }
15
16 inline std::ostream &operator«(std::ostream &os, Vec3d const &obj) {
17   os « obj[0] « ", " « obj[1] « ", " « obj[2];
18   return os;
19 }
20
21 inline std::ostream &operator«(std::ostream &os, Vec4d const &obj) {
22   os « obj[0] « ", " « obj[1] « ", " « obj[2] « ", " « obj[3];
23   return os;
24 }
25
26 inline std::ostream &operator«(std::ostream &os, Mat3d const &obj) {
27   os « obj[0] « std::endl « obj[1] « std::endl « obj[2];
28   return os;
29 }
30
31 inline Vec3d operator+(Vec3d const &lhs, double rhs) {
32   Vec3d res;
33   res[0] = lhs[0] + rhs;
34   res[1] = lhs[1] + rhs;
35   res[2] = lhs[2] + rhs;
36   return res;
37 }
38
39 inline Vec3d operator+(double lhs, Vec3d const &rhs) {
40   Vec3d res;
41   res[0] = lhs + rhs[0];
42   res[1] = lhs + rhs[1];
43   res[2] = lhs + rhs[2];
44   return res;
45 }
46
47 inline Vec3i operator+(Vec3i const &lhs, int rhs) {
48   Vec3i res;
49   res[0] = lhs[0] + rhs;
50   res[1] = lhs[1] + rhs;
51   res[2] = lhs[2] + rhs;
52   return res;
53 }
54
55 inline Vec3i operator+(int lhs, Vec3i const &rhs) {
56   Vec3i res;
57   res[0] = lhs + rhs[0];
58   res[1] = lhs + rhs[1];
59   res[2] = lhs + rhs[2];
60   return res;
61 }
62
63 inline Vec3d operator-(Vec3d const &lhs, double rhs) {
64   Vec3d res;
65   res[0] = lhs[0] - rhs;
66   res[1] = lhs[1] - rhs;
67   res[2] = lhs[2] - rhs;
68   return res;
69 }
70
71 inline Vec3d operator-(double lhs, Vec3d const &rhs) {
72   Vec3d res;
73   res[0] = lhs - rhs[0];
74   res[1] = lhs - rhs[1];
75   res[2] = lhs - rhs[2];
76   return res;
77 }
78
79 inline Vec3d operator*(Vec3d const &lhs, double rhs) {
80   Vec3d res;
81   res[0] = lhs[0] * rhs;
82   res[1] = lhs[1] * rhs;
83   res[2] = lhs[2] * rhs;
84   return res;
85 }
86
87 inline Vec3d operator*(double lhs, Vec3d const &rhs) {
88   Vec3d res;
89   res[0] = lhs * rhs[0];
90   res[1] = lhs * rhs[1];
91   res[2] = lhs * rhs[2];
92   return res;
93 }
94
95 inline Vec3d operator/(Vec3d const &lhs, double rhs) {
96   Vec3d res;
97   res[0] = lhs[0] / rhs;
98   res[1] = lhs[1] / rhs;
```

```
99   res[2] = lhs[2] / rhs;
100   return res;
101 }
102
103 inline Vec3d operator/(double lhs, Vec3d const &rhs) {
104   Vec3d res;
105   res[0] = lhs / rhs[0];
106   res[1] = lhs / rhs[1];
107   res[2] = lhs / rhs[2];
108   return res;
109 }
110
111 inline Vec3d operator+(Vec3d const &lhs, Vec3d const &rhs) {
112   Vec3d res;
113   res[0] = lhs[0] + rhs[0];
114   res[1] = lhs[1] + rhs[1];
115   res[2] = lhs[2] + rhs[2];
116   return res;
117 }
118
119 inline Vec3d operator-(Vec3d const &lhs, Vec3d const &rhs) {
120   Vec3d res;
121   res[0] = lhs[0] - rhs[0];
122   res[1] = lhs[1] - rhs[1];
123   res[2] = lhs[2] - rhs[2];
124   return res;
125 }
126
127 inline Vec3d operator*(Vec3d const &lhs, Vec3d const &rhs) {
128   Vec3d res;
129   res[0] = lhs[0] * rhs[0];
130   res[1] = lhs[1] * rhs[1];
131   res[2] = lhs[2] * rhs[2];
132   return res;
133 }
134
135 inline Vec3d operator/(Vec3d const &lhs, Vec3d const &rhs) {
136   Vec3d res;
137   res[0] = lhs[0] / rhs[0];
138   res[1] = lhs[1] / rhs[1];
139   res[2] = lhs[2] / rhs[2];
140   return res;
141 }
142
143 namespace Math {
144
145 template <typename T> inline int Sign(T val) {
146   return (T(0) < val) - (val < T(0));
147 }
148
149 inline double NormLen(Vec2d const &val) {
150   return std::sqrt((long double)(val[0]) * (long double)(val[0]) +
151                    (long double)(val[1]) * (long double)(val[1]));
152 }
153
154 inline double NormLen(Vec3d const &val) {
155   return std::sqrt((long double)(val[0]) * (long double)(val[0]) +
156                    (long double)(val[1]) * (long double)(val[1]) +
157                    (long double)(val[2]) * (long double)(val[2]));
158 }
159
160 inline double NormLen(double val_0, double val_1) {
161   return std::sqrt((long double)val_0 * (long double)val_0 +
162                    (long double)val_1 * (long double)val_1);
163 }
164
165 inline double NormLen(double val_0, double val_1, double val_2) {
166   return std::sqrt((long double)val_0 * (long double)val_0 +
167                    (long double)val_1 * (long double)val_1 +
168                    (long double)val_2 * (long double)val_2);
169 }
170
171 inline double NormLen(double val_0, double val_1, double val_2, double val_3) {
172   return std::sqrt((long double)val_0 * (long double)val_0 +
173                    (long double)val_1 * (long double)val_1 +
174                    (long double)val_2 * (long double)val_2 +
175                    (long double)val_3 * (long double)val_3);
176 }
177
178 inline double Determinant(Mat2d const &mat) {
179   return mat[0][0] * mat[1][1] - mat[0][1] * mat[1][0];
180 }
181
182 inline double Determinant(Mat3d const &mat) {
183   return mat[0][0] * mat[1][1] * mat[2][2] + mat[0][1] * mat[1][2] * mat[2][0] +
184          mat[0][2] * mat[1][0] * mat[2][1] - mat[0][0] * mat[1][2] * mat[2][1] -
185          mat[0][1] * mat[1][0] * mat[2][2] - mat[0][2] * mat[1][1] * mat[2][0];
```

```
186 }
187
188 inline Mat2d Inverse(Mat2d const &m_val) {
189   double m_det = Determinant(m_val);
190   if (std::abs(m_det) < 1.0e-24) {
191     std::cout « "in Math::Inverse: mat determinant less than 1.0e-24"
192               « std::endl;
193   }
194
195   Mat2d m_res;
196
197   m_res[0][0] = m_val[1][1] / m_det;
198   m_res[0][1] = -m_val[0][1] / m_det;
199
200   m_res[1][0] = -m_val[1][0] / m_det;
201   m_res[1][1] = m_val[0][0] / m_det;
202
203   return m_res;
204 }
205
206 inline Mat3d Inverse(Mat3d const &m_val) {
207   double m_det = Determinant(m_val);
208   if (std::abs(m_det) < 1.0e-24) {
209     std::cout « "in Math::Inverse: mat determinant less than 1.0e-24"
210               « std::endl;
211   }
212
213   Mat3d m_res;
214
215   m_res[0][0] = (m_val[1][1] * m_val[2][2] - m_val[2][1] * m_val[1][2]) / m_det;
216   m_res[0][1] = (m_val[2][1] * m_val[0][2] - m_val[0][1] * m_val[2][2]) / m_det;
217   m_res[0][2] = (m_val[0][1] * m_val[1][2] - m_val[1][1] * m_val[0][2]) / m_det;
218
219   m_res[1][0] = (m_val[1][2] * m_val[2][0] - m_val[2][2] * m_val[1][0]) / m_det;
220   m_res[1][1] = (m_val[2][2] * m_val[0][0] - m_val[0][2] * m_val[2][0]) / m_det;
221   m_res[1][2] = (m_val[0][2] * m_val[1][0] - m_val[1][2] * m_val[0][0]) / m_det;
222
223   m_res[2][0] = (m_val[1][0] * m_val[2][1] - m_val[2][0] * m_val[1][1]) / m_det;
224   m_res[2][1] = (m_val[2][0] * m_val[0][1] - m_val[0][0] * m_val[2][1]) / m_det;
225   m_res[2][2] = (m_val[0][0] * m_val[1][1] - m_val[1][0] * m_val[0][1]) / m_det;
226
227   return m_res;
228 }
229
230 template <size_t r, size_t cr, size_t c>
231 inline MatNd<r, c> Dot(MatNd<r, cr> const &m_1, MatNd<cr, c> const &m_2) {
232   MatNd<r, c> m_res;
233
234   for (size_t i = 0; i < r; i++) {
235     for (size_t j = 0; j < c; j++) {
236       m_res[i][j] = 0;
237     }
238   }
239
240   for (size_t i = 0; i < r; i++) {
241     for (size_t k = 0; k < cr; k++) {
242       for (size_t j = 0; j < c; j++) {
243         m_res[i][j] += m_1[i][k] * m_2[k][j];
244       }
245     }
246   }
247
248   return m_res;
249 }
250
251 template <size_t r, size_t cr, size_t c>
252 inline MatNd<r, c> DotTransportLHS(MatNd<cr, r> const &m_1,
253                                    MatNd<cr, c> const &m_2) {
254   MatNd<r, c> m_res;
255
256   for (size_t i = 0; i < r; i++) {
257     for (size_t j = 0; j < c; j++) {
258       m_res[i][j] = 0;
259     }
260   }
261
262   for (size_t i = 0; i < r; i++) {
263     for (size_t k = 0; k < cr; k++) {
264       for (size_t j = 0; j < c; j++) {
265         m_res[i][j] += m_1[k][i] * m_2[k][j];
266       }
267     }
268   }
269
270   return m_res;
271 }
272
```

```
273 template <size_t r, size_t cr, size_t c>
274 inline MatNd<r, c> DotTransportRHS(MatNd<r, cr> const &m_1,
275                                    MatNd<c, cr> const &m_2) {
276   MatNd<r, c> m_res;
277
278   for (size_t i = 0; i < r; i++) {
279     for (size_t j = 0; j < c; j++) {
280       m_res[i][j] = 0;
281     }
282   }
283
284   for (size_t i = 0; i < r; i++) {
285     for (size_t j = 0; j < c; j++) {
286       for (size_t k = 0; k < cr; k++) {
287         m_res[i][j] += m_1[i][k] * m_2[j][k];
288       }
289     }
290   }
291
292   return m_res;
293 }
294
295 inline Vec3d Cross(Vec3d const &val_1, Vec3d const &val_2) {
296   Vec3d val_res;
297   val_res[0] = val_1[1] * val_2[2] - val_1[2] * val_2[1];
298   val_res[1] = val_1[2] * val_2[0] - val_1[0] * val_2[2];
299   val_res[2] = val_1[0] * val_2[1] - val_1[1] * val_2[0];
300   return val_res;
301 }
302
303 inline double Dot(Vec3d const &val_1, Vec3d const &val_2) {
304   return val_1[0] * val_2[0] + val_1[1] * val_2[1] + val_1[2] * val_2[2];
305 }
306
307 inline double Dot(VecXT<double> const &val_1, VecXT<double> const &val_2) {
308   if (val_1.size() != val_2.size()) {
309     std::cout « "in Math::Dot, vector size not equal: " « std::endl;
310     std::abort();
311   }
312
313   double res = 0.0;
314   for (int i = 0; i < val_1.size(); i++) {
315     res += val_1[i] * val_2[i];
316   }
317   return res;
318 }
319
320 inline void Normalize(Vec3d *const val) {
321   double tmp_norm = NormLen(*val);
322   if (tmp_norm > 0.0) {
323     double by_val_norm = 1.0 / tmp_norm;
324     (*val)[0] *= by_val_norm;
325     (*val)[1] *= by_val_norm;
326     (*val)[2] *= by_val_norm;
327   } else {
328     std::cout « "in Math::Normalize, vector is of zero norm len" « std::endl;
329     (*val)[0] = 1.0;
330     (*val)[1] = 0.0;
331     (*val)[2] = 0.0;
332   }
333 }
334
335 namespace Quaternion {
336
337 inline Vec4d FromRodrigues(double rot_angle, Vec3d const &rot_axis) {
338   Vec4d quat;
339
340   quat[0] = std::cos(0.5 * rot_angle);
341   double sin_half_angle_by_axis_norm =
342       std::sin(0.5 * rot_angle) / NormLen(rot_axis);
343
344   quat[1] = rot_axis[0] * sin_half_angle_by_axis_norm;
345   quat[2] = rot_axis[1] * sin_half_angle_by_axis_norm;
346   quat[3] = rot_axis[2] * sin_half_angle_by_axis_norm;
347
348   return quat;
349 }
350
351 inline std::tuple<double, Vec3d> ToRodrigues(Vec4d const &quat) {
352   double rot_angle =
353       2.0 * std::atan2(NormLen(quat[1], quat[2], quat[3]), quat[0]);
354   Vec3d rot_axis;
355
356   double sin_half_angle = std::sin(0.5 * rot_angle);
357   if (std::abs(sin_half_angle) > 1.0e-15) {
358     double term_norm =
359         1.0 / sin_half_angle / NormLen(quat[0], quat[1], quat[2], quat[3]);
```

```
360     rot_axis[0] = quat[1] * term_norm;
361     rot_axis[1] = quat[2] * term_norm;
362     rot_axis[2] = quat[3] * term_norm;
363   } else {
364     rot_axis[0] = 1.0;
365     rot_axis[1] = 0.0;
366     rot_axis[2] = 0.0;
367   }
368
369   return std::tie(rot_angle, rot_axis);
370 }
371
372 inline Vec4d FromMatrix(Mat3d const &rot_mat) {
373   Vec4d quat;
374
375   quat[0] =
376       0.5 * std::sqrt(std::max(1.0e-24, 1.0 + rot_mat[0][0] + rot_mat[1][1] +
377                                             rot_mat[2][2]));
378   quat[1] = 0.25 * (rot_mat[2][1] - rot_mat[1][2]) / quat[0];
379   quat[2] = 0.25 * (rot_mat[0][2] - rot_mat[2][0]) / quat[0];
380   quat[3] = 0.25 * (rot_mat[1][0] - rot_mat[0][1]) / quat[0];
381
382   // double by_q_norm = 1.0 / NormLen(quat[0], quat[1], quat[2], quat[3]);
383   // quat[0] *= by_q_norm;
384   // quat[1] *= by_q_norm;
385   // quat[2] *= by_q_norm;
386   // quat[3] *= by_q_norm;
387
388   return quat;
389 }
390
391 inline Mat3d ToMatrix(Vec4d const &quat) {
392   Mat3d rot_mat;
393
394   rot_mat[0][0] = 1 - 2 * quat[2] * quat[2] - 2 * quat[3] * quat[3];
395   rot_mat[0][1] = 2 * quat[1] * quat[2] - 2 * quat[0] * quat[3];
396   rot_mat[0][2] = 2 * quat[1] * quat[3] + 2 * quat[0] * quat[2];
397
398   rot_mat[1][0] = 2 * quat[1] * quat[2] + 2 * quat[0] * quat[3];
399   rot_mat[1][1] = 1 - 2 * quat[1] * quat[1] - 2 * quat[3] * quat[3];
400   rot_mat[1][2] = 2 * quat[2] * quat[3] - 2 * quat[0] * quat[1];
401
402   rot_mat[2][0] = 2 * quat[1] * quat[3] - 2 * quat[0] * quat[2];
403   rot_mat[2][1] = 2 * quat[2] * quat[3] + 2 * quat[0] * quat[1];
404   rot_mat[2][2] = 1 - 2 * quat[1] * quat[1] - 2 * quat[2] * quat[2];
405
406   return rot_mat;
407 }
408
409 inline Vec4d Multiply(Vec4d const &p, Vec4d const &q) {
410   Vec4d res;
411
412   Vec3d p_vec{p[1], p[2], p[3]};
413   Vec3d q_vec{q[1], q[2], q[3]};
414   Vec3d p_cross_q = Cross(p_vec, q_vec);
415
416   res[0] = p[0] * q[0] - Dot(p_vec, q_vec);
417   res[1] = p[0] * q_vec[0] + q[0] * p_vec[0] + p_cross_q[0];
418   res[2] = p[0] * q_vec[1] + q[0] * p_vec[1] + p_cross_q[1];
419   res[3] = p[0] * q_vec[2] + q[0] * p_vec[2] + p_cross_q[2];
420
421   return res;
422 }
423
424 inline Vec4d Add(Vec4d const &p, Vec4d const &q) {
425   Vec4d res;
426
427   res[0] = p[0] + q[0];
428   res[1] = p[1] + q[1];
429   res[2] = p[2] + q[2];
430   res[3] = p[3] + q[3];
431
432   return res;
433 }
434
435 inline Vec4d Conjugate(Vec4d const &p) {
436   Vec4d res;
437
438   res[0] = p[0];
439   res[1] = -p[1];
440   res[2] = -p[2];
441   res[3] = -p[3];
442
443   return res;
444 }
445
446 inline void Normalize(Vec4d *const q) {
```

```
447   double by_q_norm = 1.0 / NormLen((*q)[0], (*q)[1], (*q)[2], (*q)[3]);
448   (*q)[0] *= by_q_norm;
449   (*q)[1] *= by_q_norm;
450   (*q)[2] *= by_q_norm;
451   (*q)[3] *= by_q_norm;
452 }
453
454 } // namespace Quaternion
455
456 inline Vec3d Rotate(Vec3d const &val_old, double rot_angle_cos,
457                     double rot_angle_sin, Vec3d const &rot_axis) {
458   Vec3d val_new;
459
460   double by_axis_norm = 1.0 / NormLen(rot_axis);
461   Vec3d rot_axis_unit{rot_axis[0] * by_axis_norm, rot_axis[1] * by_axis_norm,
462                       rot_axis[2] * by_axis_norm};
463
464   auto axis_cross_val = Cross(rot_axis_unit, val_old);
465   double axis_dot_val = Dot(rot_axis_unit, val_old);
466
467   val_new[0] = val_old[0] * rot_angle_cos + axis_cross_val[0] * rot_angle_sin +
468                rot_axis_unit[0] * axis_dot_val * (1.0 - rot_angle_cos);
469   val_new[1] = val_old[1] * rot_angle_cos + axis_cross_val[1] * rot_angle_sin +
470                rot_axis_unit[1] * axis_dot_val * (1.0 - rot_angle_cos);
471   val_new[2] = val_old[2] * rot_angle_cos + axis_cross_val[2] * rot_angle_sin +
472                rot_axis_unit[2] * axis_dot_val * (1.0 - rot_angle_cos);
473
474   return val_new;
475 }
476
477 inline Vec3d Rotate(Vec3d const &val_old, double rot_angle,
478                     Vec3d const &rot_axis) {
479   return Rotate(val_old, std::cos(rot_angle), std::sin(rot_angle), rot_axis);
480 }
481
482 inline Vec3d Rotate(Vec3d const &val_old, Vec4d const &quat) {
483   Vec3d val_new;
484
485   double val_norm = NormLen(val_old);
486   if (val_norm > 1.0e-15) {
487     auto quat_conj = Quaternion::Conjugate(quat);
488     Vec4d quat_val_old{0, val_old[0], val_old[1], val_old[2]};
489
490     auto quat_val_tmp = Quaternion::Multiply(quat_val_old, quat_conj);
491     auto quat_val_new = Quaternion::Multiply(quat, quat_val_tmp);
492
493     double renorm_factor =
494         val_norm / NormLen(quat_val_new[1], quat_val_new[2], quat_val_new[3]);
495     val_new[0] = quat_val_new[1] * renorm_factor;
496     val_new[1] = quat_val_new[2] * renorm_factor;
497     val_new[2] = quat_val_new[3] * renorm_factor;
498   } else {
499     val_new[0] = 0;
500     val_new[1] = 0;
501     val_new[2] = 0;
502   }
503
504   return val_new;
505 }
506
507 inline Vec3d Rotate(Vec3d const &val_old, Mat3d const &rot_mat) {
508   Vec3d val_new;
509
510   val_new[0] = rot_mat[0][0] * val_old[0] + rot_mat[0][1] * val_old[1] +
511                rot_mat[0][2] * val_old[2];
512   val_new[1] = rot_mat[1][0] * val_old[0] + rot_mat[1][1] * val_old[1] +
513                rot_mat[1][2] * val_old[2];
514   val_new[2] = rot_mat[2][0] * val_old[0] + rot_mat[2][1] * val_old[1] +
515                rot_mat[2][2] * val_old[2];
516
517   return val_new;
518 }
519
520 inline Vec3d CartesianToSpherical(Vec3d const &vert_cart) {
521   Vec3d vert_sph;
522   vert_sph[0] = NormLen(vert_cart);
523   vert_sph[1] = std::acos(vert_cart[2] / vert_sph[0]);
524   vert_sph[2] = std::atan2(vert_cart[1], vert_cart[0]);
525   return vert_sph;
526 }
527
528 inline Vec3d SphericalToCartesian(Vec3d const &vert_sph) {
529   Vec3d vert_cart;
530   vert_cart[0] = vert_sph[0] * std::sin(vert_sph[1]) * std::cos(vert_sph[2]);
531   vert_cart[1] = vert_sph[0] * std::sin(vert_sph[1]) * std::sin(vert_sph[2]);
532   vert_cart[2] = vert_sph[0] * std::cos(vert_sph[1]);
533   return vert_cart;
```

```
534 }
535
536 } // namespace Math
537
538 } // namespace netdem
```

8.349 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/voronoi.cpp File Reference

```
#include "voronoi.hpp"
#include "cork_wrapper.hpp"
#include "distribution_uniform.hpp"
#include "igl_wrapper.hpp"
#include "utils_io.hpp"
#include <voro++/voro++.hh>
```

8.350 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/voronoi.hpp File Reference

```
#include "igl_wrapper.hpp"
#include "stl_model.hpp"
```

Classes

- class netdem::Voronoi

Namespaces

- namespace netdem

8.351 voronoi.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "igl_wrapper.hpp"
4 #include "stl_model.hpp"
5
6 namespace netdem {
7
8 class Voronoi {
9 public:
10   // basic voronoi. return: vt_nodes, vt_cells
11   static std::tuple<VecXT<Vec3d>, VecXT<VecXT<Vec3i>>
12   Solve(VecXT<Vec3d> const &vt_seeds, STLModel const &stl_model,
13         bool use_cork = true);
14
15   // centroidal voronoi. return: vt_nodes, vt_cells, vt_seeds
```

```
16   static std::tuple<VecXT<Vec3d>, VecXT<VecXT<Vec3i>>, VecXT<Vec3d>>
17   Solve(STLModel const &stl_model, int num_seeds, int max_iter = 1000,
18         double tol = 1.0e-3, bool use_cork = true);
19
20   static void SaveAsVTK(std::string const &file, VecXT<Vec3d> const &vt_nodes,
21                         VecXT<VecXT<Vec3i>> const &vt_cells,
22                         VecXT<Vec3d> const &vt_seeds);
23
24 private:
25   // basic voronoi. return: vt_nodes, vt_cells
26   static std::tuple<VecXT<Vec3d>, VecXT<VecXT<Vec3i>>»
27   Solve(VecXT<Vec3d> const &vt_seeds, STLModel const &stl_model,
28         SDFCalculator const &sdf_calculator, bool use_cork = true);
29 };
30
31 } // namespace netdem
```

8.352 /Users/lzhshou/Documents/Research/myProjects/dem_↩ developments/net_dem/netdem/src/utils/wscvt_sampler.hpp File Reference

```
#include "mini_map.hpp"
#include "spherical_voronoi.hpp"
#include "utils_math.hpp"
```

Classes

- class netdem::WSCVTSampler

Namespaces

- namespace netdem

8.353 wscvt_sampler.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 #include "mini_map.hpp"
4 #include "spherical_voronoi.hpp"
5 #include "utils_math.hpp"
6
7 namespace netdem {
8
9 class WSCVTSampler {
10 public:
11   int max_iters{10000};
12   double tol{1.0e-4};
13
14   WSCVTSampler(const WSCVTSampler &) = delete;
15   WSCVTSampler &operator=(const WSCVTSampler &) = delete;
16   static WSCVTSampler &GetInstance() {
17     static WSCVTSampler instance;
18     return instance;
19   }
20
21   VecXT<Vec3d> Get(int num_samples, bool new_random = false) {
22     VecXT<Vec3d> samples;
23     if (new_random) {
24       samples = Create(num_samples);
25       if (samples_map.find(num_samples) == samples_map.end()) {
26         samples_map[num_samples] = samples;
```

```
27       }
28     } else {
29       if (samples_map.find(num_samples) == samples_map.end()) {
30         samples = Create(num_samples);
31         samples_map[num_samples] = samples;
32       } else {
33         samples = samples_map[num_samples];
34       }
35     }
36     return samples;
37   }
38
39 private:
40   WSCVTSampler() {}
41
42   MiniMap<int, VecXT<Vec3d>> samples_map;
43
44   VecXT<Vec3d> Create(int num_samples) {
45     return SphericalVoronoi::Solve(num_samples, 10000, 1.0e-4);
46   }
47 };
48
49 } // namespace netdem
```