DEMFEMVolumeCouplingApplication

Implementation and Usage Guide

Overview

This document describes the implementation and usage of the **DEMFEMVolumeCouplingApplication** in Kratos Multiphysics. It provides the information needed to compile, run, and extend the volume–coupled DEM–FEM solver. A complete, runnable example is available in the `applications/DEMFEMVolumeCouplingApplication/tests` folder.

The coupling strategy is based on the *volume–coupling* approach and is implemented entirely within Kratos as a new application. It builds on existing Kratos modules: **StructuralMechanicsApplication** (FEM), **DEMApplication** (DEM), **MappingApplication**, **ConstitutiveLawsApplication**, and the solver-agnostic **CoSimulationApplication**.

1 Source Tree and Build Integration

Source Locations

New functionality is implemented in three main locations:

`applications/DEMFEMVolumeCouplingApplication` Contains all FEM/DEM extensions:

- **Top level**: `CMakeLists.txt`, `DEMFEM_volume_coupling_application.{cpp,h,py}`
- **custom_elements**: `volume_coupling_element.{cpp,h}`, `volume_coupling_particle.{cpp,h}`
- **custom_python**: `add_custom_utilities_to_python.{cpp,h}`, `DEMFEM_volume_coupling_python_application.cpp`
- **custom_utilities**: `utility_functions.h`

`applications/CoSimulationApplication` Thin integration layer:

- `python_scripts/coupled_solvers/volume_coupling.py`: orchestrates DEM and FEM time stepping, executes coupling operations, manages data exchange
- `python_scripts/coupling_operations/compute_dem_fem_volume_coupling_force.py`: computes FEM nodal coupling forces

- python_scripts/coupling_operations/compute_dem_momentum.py: evaluates DEM momentum and contact forces
- python_scripts/solver_wrappers/kratos/volume_coupling_structural_wrapper.py: toggles ACTIVATION_LEVEL for weighted post-processing

applications/DEMApplication Minimal patch in python_scripts/DEM_procedures.py to register new variables:

```
import KratosMultiphysics.DEMFEMVolumeCouplingApplication as VCA
model_part.AddNodalSolutionStepVariable(VCA.PARTICLE_COUPLING_WEIGHT)
model_part.AddNodalSolutionStepVariable(VCA.DISPLACEMENT_MULTIPLIED_MASS)
model_part.AddNodalSolutionStepVariable(VCA.VELOCITY_MULTIPLIED_MASS)
model_part.AddNodalSolutionStepVariable(VCA.DEMFEM_VOLUME_COUPLING_FORCE)
```

No other DEM core changes are required.

Build Instructions

The root CMakeLists.txt integrates the application into the Kratos super-build. Enable at configure time:

```
–DENABLE_DEMFEM_VOLUME_COUPLING=ON
```

Recompile Kratos to include the new application.

2 Coupling Utilities

All helper routines are in custom_utilities/utility_functions.h. They operate exclusively on Kratos-native data structures and MPI communicators, ensuring thread safety and scalability.

Available Operations

- AssignPointLoads – applies frictional wall loads on selected FEM boundaries (optional)
- SetNodalCouplingWeightsOnFEMLinearly – assigns hybrid weights by linear interpolation between two horizontal planes
- SetNodalCouplingWeightsFromLayers – assigns hybrid weights from user-defined SubModelParts
- CalculateDisplacementDifference – accumulates DEM–FEM velocity mismatch over time
- CalculateNodalCouplingForces – converts the mismatch into FEM nodal forces using a penalty parameter ϵ
- CalculateNodalDEMCouplingForces – rescales FEM penalty forces with DEM lumped mass before mapping back
- CalculateMomentum – stores $m\,\mathbf{u}$ and $m\,\mathbf{v}$ in nodal variables for export
- CalculateDEMForces – multiplies external forces by nodal mass where $\varpi \neq 0$

Hybrid Region Weighting

Two strategies assign weights $\omega \in [0, 1]$:

1. **Linear interpolation** between planes at y_{FEM} and y_{DEM}
2. **Layer-based**: user supplies a dictionary mapping `SubModelPart` names to weights

The second method supports arbitrary geometry and dynamic hybrid zones.

Penalty Enforcement

Coupling uses a velocity-based penalty:

1. Integrate mismatch: $\Delta\mathbf{u} \mathrel{+}= (\mathbf{v}_{\text{DEM}} - \mathbf{v}_{\text{FEM}})\,\Delta t$
2. Multiply by penalty ϵ to compute FEM nodal forces
3. Rescale forces with DEM lumped mass and map back to particles

3 FEM Side – VolumeCouplingElement

`VolumeCouplingElement` derives from `SmallDisplacementElement`. Modifications:

- Override `GetIntegrationWeight`: scale Gauss weight W_0 by $(1-\bar{\omega})$ where $\bar{\omega}$ is the weighted sum of nodal ω_i
- Override `CalculateOnIntegrationPoints`: optionally weight output stresses by $(1-\bar{\omega})$ only when `ACTIVATION_LEVEL` $= 1$

Stress weighting is used for post-processing to form hybrid stresses; can be disabled.

4 DEM Side – VolumeCouplingParticle

`VolumeCouplingParticle` derives from `SphericParticle`. Changes:

1. `Initialize`: set $\varpi = 1$
2. `GetMass`: return $m^* = \varpi m$
3. `EvaluateBallToBallForcesForPositiveIndentations`: apply symmetric weight $\varpi_{12} = \frac{\varpi_1 a_2 + \varpi_2 a_1}{a_1 + a_2}$
4. `EvaluateBallToRigidFaceForcesForPositiveIndentations`: rescale ball–wall forces similarly

5 CoSimulation Workflow

A custom coupled solver `DemFemVolumeCoupledSolver` in `volume_coupling.py` runs the following per step:

1. Compute DEM momentum and velocity
2. Map DEM data → FEM
3. Assemble FEM penalty forces
4. Map FEM forces → DEM
5. Finalize coupling (mass scaling, extra loads)
6. Advance both DEM and FEM solvers

Lightweight Python modules implement each stage:

- `compute_dem_momentum.py` – gather DEM momentum and scale forces
- `compute_dem_fem_volume_coupling_force.py` – assemble and map FEM penalty forces
- `volume_coupling.py` – orchestrates the full sequence

6 Case Setup

A typical folder layout is: (Please check the .json files for the exact folder-file layout)

```
case/
  CoSimulationParameters.json
  dem/
    ProjectParametersDEM.json
    Materials.json
    particles.mdpa
    boundaries.mdpa
  fem/
    ProjectParametersFEM.json
    StructuralMaterials.json
    model.mdpa
```

Check the exact folder structure and names in the JSON files in the `tests` folder.

DEM Inputs

- **ProjectParametersDEM.json** – solver settings, time stepping, processes
- **Materials.json** – particle and wall material data
- **.mdpa** – particle and boundary geometries

FEM Inputs

- **ProjectParametersFEM.json** – solver settings, BCs, output
- **StructuralMaterials.json** – FEM material properties
- **model.mdpa** – mesh and `SubModelParts`

CoSimulation Inputs

- **CoSimulationParameters.json** – participants, coupling sequence, mapping settings, scheme, time control

Practical Notes

- Maintain consistent units between DEM and FEM
- Use `SubModelParts` for BCs, loads, mapping interfaces
- Example cases are in `applications/DEMFEMVolumeCouplingApplication/tests`

7 Quick Start

1. Build Kratos with:

   ```
   -DENABLE_DEMFEM_VOLUME_COUPLING=ON
   ```

2. Prepare DEM, FEM, and CoSimulation parameter files (see `tests`)
3. Run with:

   ```
   python3 co_simulation_run.py --cosp-file CoSimulationParameters.json
   ```

License

Distributed under the same license as Kratos Multiphysics.