# Tutorial: BwUniCluster 2.0/HoreKa
# Lagrangain library development

In this tutorial we will learn how to develop a Lagrangian sub-model in OpenFOAM.

1. **Creating a new Lagrangian Library and solver**

In the cluster, you are not allowed to change the source code of OpenFOAM. It is also better to keep the original source code intact. Therefore, first we create a personal Lagrangian library in the user directory.

*$>  module load cae/openfoam/v2112*

*$>  source $FOAM_INIT*

*$>  mkdir -p $WM_PROJECT_USER_DIR/src/lagrangian*

*$>  cd $WM_PROJECT_USER_DIR/src/lagrangian*

*$>  cp -rp $FOAM_SRC/lagrangian/intermediate $WM_PROJECT_USER_DIR/src/lagrangian*

*$>  mv intermediate intermediateNew*

*$>  nano $WM_PROJECT_USER_DIR/src/lagrangian/intermediateNew/Make/files*

At the end of the opened file, apply the following changes:

LIB = $(FOAM_LIBBIN)/liblagrangianIntermediate → LIB = $(FOAM_USER_LIBBIN)/liblagrangianIntermediateNew

Now we have a Lagrangian library in our user directory that can be modified. This is how we compile this library:

*$>  cd $WM_PROJECT_USER_DIR/src/lagrangian/intermediateNew*

*$>  wclean*

*$>  wmake libso*

Now we can use our own compiled library in OpenFOAM solvers. To modify the solver, we must copy it to the user directory and then link the new Lagrangian library to it.

*$>  mkdir -p $WM_PROJECT_USER_DIR/applications/solvers/lagrangian*

*$>  cp -rp $FOAM_SOLVERS/lagrangian/kinematicParcelFoam $WM_PROJECT_USER_DIR/applications/solvers/lagrangian*

*$>  cd $WM_PROJECT_USER_DIR/applications/solvers/lagrangian*

*$>  mv kinematicParcelFoam kinematicParcelFoamNew*

*$>  cd kinematicParcelFoamNew*

*$>  mv kinematicParcelFoam.C kinematicParcelFoamNew.C*

*$>  nano Make/files*

The following modifications should be applied in the opened file:

kinematicParcelFoam.C → kinematicParcelFoamNew.C
EXE = $(FOAM_APPBIN)/kinematicParcelFoam → EXE = $(FOAM_USER_APPBIN)/kinematicParcelFoamNew

We can now link our own Lagrangian library to this new solver:

*$>  nano Make/options*

Modify the options file as below to link the intermediateNew library to the kinematicParcelFoamNew solver:

```
EXE_INC = \
    -I$(FOAM_SOLVERS)/lagrangian/reactingParcelFoam \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/finiteArea/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude \
    -I$(LIB_SRC)/sampling/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/turbulenceModels/lnInclude \
    -I$(LIB_SRC)/TurbulenceModels/incompressible/lnInclude \
    -I$(LIB_SRC)/transportModels \
    -I$(LIB_SRC)/transportModels/incompressible/singlePhaseTransportModel \
    -I$(LIB_SRC)/dynamicMesh/lnInclude \
    -I$(LIB_SRC)/dynamicFvMesh/lnInclude \
    -I$(LIB_SRC)/lagrangian/distributionModels/lnInclude \
    -I$(LIB_SRC)/regionModels/regionModel/lnInclude \
    -I$(LIB_SRC)/regionModels/surfaceFilmModels/lnInclude \
    -I$(LIB_SRC)/regionFaModels/lnInclude \
    -I$(LIB_SRC)/faOptions/lnInclude \
    -I$(LIB_SRC)/lagrangian/basic/lnInclude \
    -I$(WM_PROJECT_USER_DIR)/src/lagrangian/intermediateNew/lnInclude


EXE_LIBS = \
    -lfiniteVolume \
    -lfvOptions \
    -lmeshTools \
    -lsampling \
    -lturbulenceModels \
    -lincompressibleTurbulenceModels \
    -lincompressibleTransportModels \
    -ldynamicMesh \
    -ldynamicFvMesh \
    -ltopoChangerFvMesh \
    -latmosphericModels \
    -lregionModels \
    -lsurfaceFilmModels \
    -lsurfaceFilmDerivedFvPatchFields \
    -llagrangian \
    -L$(FOAM_USER_LIBBIN) \
    -llagrangianIntermediateNew \
    -llagrangianTurbulence \
    -lregionFaModels \
    -lfiniteArea \
    -lfaOptions
```

Now you can compile the new solver with the following commands:

*$>  wclean*

*$>  wmake*

*$>  mkdir -p $WM_PROJECT_USER_DIR/run/*

*$>  cp -rn $FOAM_TUTORIALS/lagrangian/kinematicParcelFoam/spinningDisk*

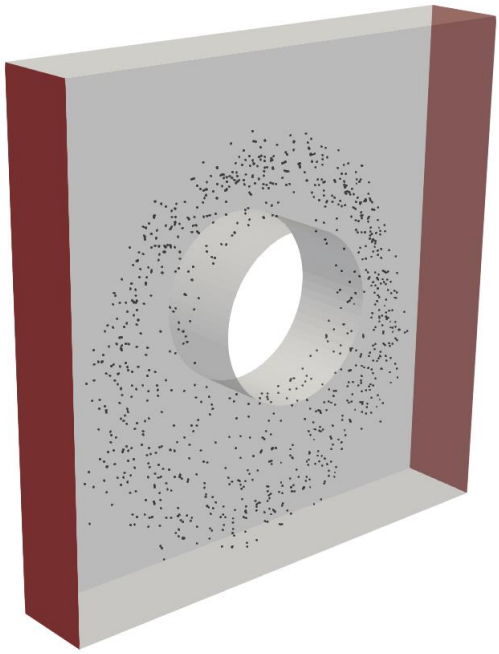
*$WM_PROJECT_USER_DIR/run*

*$>  cd $WM_PROJECT_USER_DIR/run/spinningDisk*

*$>  blockMesh*

*$>  kinematicParcelFoamNew*

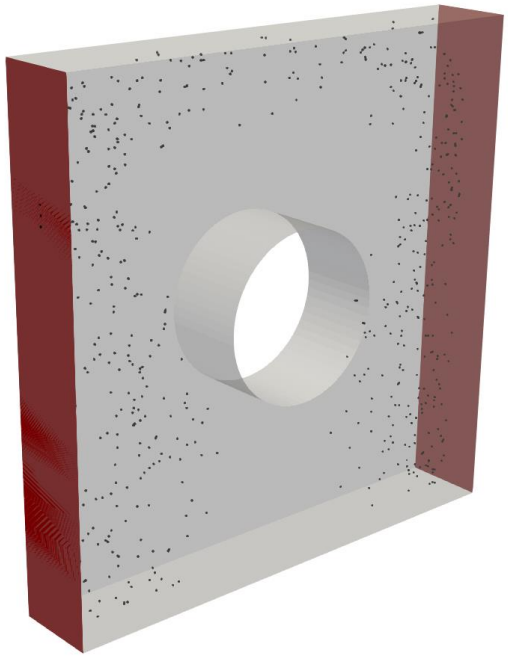
With the last command, the simulation is done by the new solver and the new Lagrangian library. Let us look at the test case:



Particles are injected from the rotating cylinder in the middle of the domain. If we increase the density of the particles and increase the duration of the simulation, they will hit the walls shown due to inertia. The Lagrangian part of the simulation is set in the kinematicCloudProperties file located in the constant folder.

```
constantProperties                constantProperties
{                                 {
    rho0          1000;     →       rho0          2700;
}                                 }
```

As seen in the picture below, in this case the particles hit the red walls. In the next section, we will develop a new erosion model in the Lagrangian library to calculate the erosion caused by particles hitting the wall.



## 2. Developing a new sub model (erosion model) in the Lagrangian library

In the first step, we copy the OpenFOAM default erosion model in our user directory and give it a new name:

*$> cp -rn $WM_PROJECT_USER_DIR/src/lagrangian/intermediateNew/submodels/CloudFunctionObjects/ParticleErosion $WM_PROJECT_USER_DIR/src/lagrangian/intermediateNew/submodels/CloudFunctionObjects/ParticleErosionNew*

*$> cd $WM_PROJECT_USER_DIR/src/lagrangian/intermediateNew/submodels/CloudFunctionObjects/ParticleErosionNew*

*$> mv ParticleErosion.C ParticleErosionNew.C*

*$> mv ParticleErosion.H ParticleErosionNew.H*

*$> sed -i 's/ParticleErosion/ParticleErosionNew/g' ParticleErosionNew.C*

*$> sed -i 's/ParticleErosion/ParticleErosionNew/g' ParticleErosionNew.H*

*$> sed -i 's/particleErosion/particleErosionNew/g' ParticleErosionNew.H*

To introduce this renamed model to the library, we modify the following file,

*$> nano $WM_PROJECT_USER_DIR/src/lagrangian/intermediateNew/parcels/include/makeParcelCloudFunctionObjects.H*

```
#include "FaceInteraction.H"
#include "FacePostProcessing.H"
#include "ParticleCollector.H"
#include "ParticleErosion.H"
#include "ParticleErosionNew.H"
#include "ParticleTracks.H"
#include "ParticleTrap.H"
#include "PatchCollisionDensity.H"
#include "PatchInteractionFields.H"
#include "PatchPostProcessing.H"
#include "PatchParticleHistogram.H"
#include "RemoveParcels.H"
#include "VoidFraction.H"
#include "KinematicReynoldsNumber.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#define makeParcelCloudFunctionObjects(CloudType)                             \
                                                                              \
    makeCloudFunctionObject(CloudType);                                       \
                                                                              \
    makeCloudFunctionObjectType(FaceInteraction, CloudType);                  \
    makeCloudFunctionObjectType(FacePostProcessing, CloudType);               \
    makeCloudFunctionObjectType(ParticleCollector, CloudType);                \
    makeCloudFunctionObjectType(ParticleErosion, CloudType);                  \
    makeCloudFunctionObjectType(ParticleErosionNew, CloudType);               \
    makeCloudFunctionObjectType(ParticleTracks, CloudType);                   \
    makeCloudFunctionObjectType(ParticleTrap, CloudType);                     \
    makeCloudFunctionObjectType(PatchCollisionDensity, CloudType);            \
    makeCloudFunctionObjectType(PatchInteractionFields, CloudType);           \
    makeCloudFunctionObjectType(PatchPostProcessing, CloudType);              \
    makeCloudFunctionObjectType(PatchParticleHistogram, CloudType);           \
    makeCloudFunctionObjectType(RemoveParcels, CloudType);                    \
    makeCloudFunctionObjectType(VoidFraction, CloudType);
```

*$>  cd $WM_PROJECT_USER_DIR/src/lagrangian/intermediateNew*

*$>  wclean*

*$>  wmake libso*

So far, we have only changed the name of the erosion model and it has not been modified yet. Different parameters are used in the literature to quantify the local erosion. Here, the erosion depth (thickness), Er(mm), is defined as:

$$Er = \frac{\sum_{impact}(n_p m_p Er_\alpha)}{A_{Cell}}$$

where Er_α is the volume of material removed from the wall surface per unit mass of impacting particle (with impact angle of α). A myriad of empirical correlations have been proposed for estimating Er_α. Here, the model by Oka et al.[1] is used:

[1] Oka, Y. Isomoto, K. Okamura, and T. Yoshida. "Practical estimation of erosion damage caused by solid particle impact: Part 1: Effects of impact parameters on a predictive equation." *Wear* 259.1-6 (2005): 95-101.
Oka, Y. I., and T. Yoshida. "Practical estimation of erosion damage caused by solid particle impact: Part 2: Mechanical properties of materials directly associated with erosion damage." *Wear* 259.1-6 (2005): 102-109.

$$Er_{\alpha} = g(\alpha)Er_{90}$$

$$g(\alpha) = (\sin\alpha)^{n_1}\left(1 + H_v(1-\sin\alpha)\right)^{n_2}$$

$$Er_{90} = K(aH_v)^{k_1 b}\left(\frac{|\mathbf{u}_p|}{u_{\text{ref}}}\right)^{k_2}\left(\frac{d_p}{d_{\text{ref}}}\right)^{k_3}$$

where Hv(GPa) is the wall Vickers hardness number. To develop the above model in OpenFOAM, we modify *ParticleErosionNew.C* and *ParticleErosionNew.H* as follows:

*$>  nano $WM_PROJECT_USER_DIR/src/lagrangian/intermediateNew/submodels/CloudFunctionObjects/ParticleErosionNew/ ParticleErosionNew.C*

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | www.openfoam.com
     \\/     M anipulation  |
-------------------------------------------------------------------------------
    Copyright (C) 2011-2017 OpenFOAM Foundation
    Copyright (C) 2019-2021 OpenCFD Ltd.
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

\*---------------------------------------------------------------------------*/

#include "ParticleErosionNew.H"

// * * * * * * * * * * * * * * * Protected Member Functions  * * * * * * * * * * //

template<class CloudType>
void Foam::ParticleErosionNew<CloudType>::resetQ()
{
    if (QPtr_)
    {
        QPtr_->primitiveFieldRef() = 0.0;
    }
    else
    {
        const fvMesh& mesh = this->owner().mesh();

        QPtr_.reset
        (
            new volScalarField
            (
                IOobject
                (
                    this->owner().name() + "Q",
                    mesh.time().timeName(),
                    mesh,
                    IOobject::READ_IF_PRESENT,
                    IOobject::NO_WRITE
                ),
                mesh,
```

```
                dimensionedScalar(dimVolume, Zero)
            )
        );
    }
}


template<class CloudType>
Foam::label Foam::ParticleErosionNew<CloudType>::applyToPatch
(
    const label globalPatchi
) const
{
    return patchIDs_.find(globalPatchi);
}


template<class CloudType>
void Foam::ParticleErosionNew<CloudType>::write()
{
    if (QPtr_)
    {
        QPtr_->write();
    }
    else
    {
        FatalErrorInFunction
            << "QPtr not valid" << abort(FatalError);
    }
}


// * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * * * //

template<class CloudType>
Foam::ParticleErosionNew<CloudType>::ParticleErosionNew
(
    const dictionary& dict,
    CloudType& owner,
    const word& modelName
)
:
    CloudFunctionObject<CloudType>(dict, owner, modelName, typeName),
    QPtr_(nullptr),
    patchIDs_(),
    n1_(this->coeffDict().template lookupOrDefault<scalar>("n1", 2.0)),
    n2_(this->coeffDict().template lookupOrDefault<scalar>("n1", 2.0)),
    Hv_(this->coeffDict().template lookupOrDefault<scalar>("Hv", 2.9)),
    K_(this->coeffDict().template lookupOrDefault<scalar>("K", 65.0)),
    a_(this->coeffDict().template lookupOrDefault<scalar>("a", 0.333)),
    b_(this->coeffDict().template lookupOrDefault<scalar>("b", 1.0)),
    k1_(this->coeffDict().template lookupOrDefault<scalar>("k1", -0.12)),
    k2_(this->coeffDict().template lookupOrDefault<scalar>("k2", 2.395)),
    k3_(this->coeffDict().template lookupOrDefault<scalar>("k3", 0.19)),
    vPrime_(this->coeffDict().template lookupOrDefault<scalar>("vPrime", 104.0)),
    dPrime_(this->coeffDict().template lookupOrDefault<scalar>("dPrime", 326.0e-6))
{
    const wordList allPatchNames(owner.mesh().boundaryMesh().names());
    const wordRes patchNames
    (
        this->coeffDict().template get<wordRes>("patches")
    );

    labelHashSet uniqIds;
    for (const wordRe& re : patchNames)
    {
        labelList ids = findMatchingStrings(re, allPatchNames);

        if (ids.empty())
        {
            WarningInFunction
                << "Cannot find any patch names matching " << re
                << endl;
        }

        uniqIds.insert(ids);
    }

    patchIDs  = uniqIds.sortedToc();

    // Trigger creation of the Q field
```

```
    resetQ();
}


template<class CloudType>
Foam::ParticleErosionNew<CloudType>::ParticleErosionNew
(
    const ParticleErosionNew<CloudType>& pe
)
:
    CloudFunctionObject<CloudType>(pe),
    QPtr_(nullptr),
    patchIDs_(pe.patchIDs_),
    n1_(pe.n1_),
    n2_(pe.n2_),
    Hv_(pe.Hv_),
    K_(pe.K_),
    a_(pe.a_),
    b_(pe.b_),
    k1_(pe.k1_),
    k2_(pe.k2_),
    k3_(pe.k3_),
    vPrime_(pe.vPrime_),
    dPrime_(pe.dPrime_)
{}


// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //

template<class CloudType>
void Foam::ParticleErosionNew<CloudType>::preEvolve
(
    const typename parcelType::trackingData& td
)
{
    resetQ();
}


template<class CloudType>
void Foam::ParticleErosionNew<CloudType>::postPatch
(
    const parcelType& p,
    const polyPatch& pp,
    bool&
)
{
    const label patchi = pp.index();

    const label localPatchi = applyToPatch(patchi);

    if (localPatchi != -1)
    {
        vector nw;
        vector Up;

        // patch-normal direction
        this->owner().patchData(p, pp, nw, Up);

        // particle velocity relative to patch
        const vector& U = p.U() - Up;

        // particle diameter
        const scalar& dp = p.d();

        // quick reject if particle travelling away from the patch
        if ((nw & U) < 0)
        {
            return;
        }

        const scalar magU = mag(U);
        const vector Udir = U/magU;

        // determine impact angle, alpha
        const scalar alpha = mathematical::piByTwo - acos(nw & Udir);

        const scalar gAlpha = pow(sin(alpha),n1_)*pow((1.0+Hv_*(1.0-sin(alpha))),n2_);//n1,n2,Hv
//added by Hesam
        const scalar E90 = K_*pow((a_*Hv_),(k1_*b_))*pow((magU/vPrime_),k2_)*pow((dp/dPrime_),k3_);
//K_,a_,k1_,b_,vPrime_,dPrime_
```

```
        // mesh things
        const label patchFacei = pp.whichFace(p.face());
        const fvMesh& mesh = this->owner().mesh();
        const surfaceScalarField& magSf = mesh.magSf();
        scalar area = magSf.boundaryField()[patchi][patchFacei];

        scalar& Q = QPtr_->boundaryFieldRef()[patchi][patchFacei];
        Q += 0.000001*(gAlpha*E90*p.nParticle()*p.mass())/area;
    }
}


// ************************************************************************* //
```

*$> nano $WM_PROJECT_USER_DIR/src/lagrangian/intermediateNew/submodels/CloudFunctionObjects/ParticleErosionNew/ ParticleErosionNew.H*

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | www.openfoam.com
     \\/     M anipulation  |
-------------------------------------------------------------------------------
    Copyright (C) 2011-2017 OpenFOAM Foundation
    Copyright (C) 2020 OpenCFD Ltd.
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Class
    Foam::ParticleErosionNew

Group
    grpLagrangianIntermediateFunctionObjects

Description
    Creates particle erosion field, Q

SourceFiles
    ParticleErosionNew.C

\*---------------------------------------------------------------------------*/

#ifndef ParticleErosionNew_H
#define ParticleErosionNew_H

#include "CloudFunctionObject.H"
#include "volFields.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam
{

/*---------------------------------------------------------------------------*\
                      Class ParticleErosionNew Declaration
\*---------------------------------------------------------------------------*/

template<class CloudType>
class ParticleErosionNew
:
```

```
    public CloudFunctionObject<CloudType>
{
    // Private Data

        // Typedefs

            //- Convenience typedef for parcel type
            typedef typename CloudType::parcelType parcelType;


        //- Particle erosion field
        autoPtr<volScalarField> QPtr_;

        //- List of patch indices to post-process
        labelList patchIDs_;

        //- n1
        scalar n1_;

        //- n2
        scalar n2_;

        //- Hv
        scalar Hv_;

        //- K
        scalar K_;

        //- a
        scalar a_;

        //- b
        scalar b_;

        //- k1
        scalar k1_;

        //- k2
        scalar k2_;

        //- k3
        scalar k3_;

        //- vPrime
        scalar vPrime_;

        //- dPrime
        scalar dPrime_;


protected:

    // Protected Member Functions

        //- Create|read|reset the Q field
        void resetQ();

        //- Returns local patchi if patch is in patchIds_ list
        label applyToPatch(const label globalPatchi) const;

        //- Write post-processing info
        virtual void write();


public:

    //- Runtime type information
    TypeName("particleErosionNew");


    // Constructors

        //- Construct from dictionary
        ParticleErosionNew
        (
            const dictionary& dict,
            CloudType& owner,
            const word& modelName
        );

        //- Construct copy
```

```
        ParticleErosionNew(const ParticleErosionNew<CloudType>& pe);

        //- Construct and return a clone
        virtual autoPtr<CloudFunctionObject<CloudType>> clone() const
        {
            return autoPtr<CloudFunctionObject<CloudType>>
            (
                new ParticleErosionNew<CloudType>(*this)
            );
        }


    //- Destructor
    virtual ~ParticleErosionNew() = default;


    // Member Functions

        // Evaluation

            //- Pre-evolve hook
            virtual void preEvolve
            (
                const typename parcelType::trackingData& td
            );

            //- Post-patch hook
            virtual void postPatch
            (
                const parcelType& p,
                const polyPatch& pp,
                bool& keepParticle
            );
};


// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace Foam

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#ifdef NoRepository
    #include "ParticleErosionNew.C"
#endif

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#endif

// ************************************************************************* //
```

Now we can recompile the library and run the simulation with the solver. To consider the new sub-model in the simulation add the following lines the *kinematicCloudProperties*.

```
cloudFunctions
{
   particleErosion1
   {
    type particleErosionNew;
      patches    (walls);
   }
}
```

In the image below, you can see the contour of the erosion depth.