Report on creating FSI simulation using OpenFoam

Table of Contents

[1. Installing solids4Foam 2](#_Toc70863755)

[1.1. Preparation 2](#_Toc70863756)

[1.2. Installation 2](#_Toc70863757)

[1.3. Running tutorial case 3](#_Toc70863758)

[2. Implementing one-way FSI for a wind turbine 4](#_Toc70863759)

[2.1. Creating one-way FSI solver 4](#_Toc70863760)

[2.2. Modifying existing case 5](#_Toc70863761)

[2.3. Running a case 9](#_Toc70863762)

[3. Codes 10](#_Toc70863763)

[3.1. oneWayFSI.C 10](#_Toc70863764)

[3.2. oneWayFSI.H 15](#_Toc70863765)

# Installing solids4Foam

## Preparation

There are different tools for implementing FSI into OpenFoam, however, solids4Foam is the easiest and most versatile among them. You can read more about this tool at the following link:

<https://bitbucket.org/philip_cardiff/solids4foam-release/src/master/>

This tool requires extend version of the OpenFoam to be installed (*note: not .com or .org version*). It is highly recommended to use Foam Extend 4.0 because, according to the author, almost all tests were done on this version. You can build it by following this link (*note: installation process may differ for different ubuntu versions*):

<https://openfoamwiki.net/index.php/Installation/Linux/foam-extend-4.0>

## Installation

After Foam Extend 4.0 is installed, the solids4Foam could be installed. You can follow instructions on the site or described in this report:

1. Clone git repository. It is suggested to use main branch of the git. To clone it, type following command into your terminal window:

The dev branch, however, has the most recent fixes and should be stable enough for simulations. To do this, use following command instead:

Both commands will store the content of the git in the *solids4Foam* folder. You can locate this folder anywhere, however, it is suggested to keep it close to the Foam Extend folder.

1. Run Foam Extend shell environment. If you follow official instructions, the shell environment loads by typing *fe40* into the terminal window. To check if your environment is load properly, type *icoFoam -help*, and additional information about icoFoam should appear.
2. Compile the source code. Navigate to the solids4Foam folder and run compilation by typing:

If the program suggests replacing some files (*note: required for .com and .org versions*), please, follow instructions provided by the program. To check if you install everything correctly, you can run any tutorial case.

## Running tutorial case

It is suggested to run beamInCrossFlow because it is the most stable 3D FSI case. This case is located at *solids4Foam/tutorials/fluidSolidInteraction/beamInCrossFlow/elasticBeam* (*note: you can also use linearGeometryElasticBeam as well*). Similar to regular OpenFoam cases, the FSI case consists of *0*, *constant*, and *system* folders. However, each folder contains *fluid* and *solid* folders, where the rest files, required for specific domain, are located. Thus, domain should be specified each time with running program (except solvers). For example, to run create a mesh using blockMesh, you need to run *blockMesh -region fluid* or *blockMesh -region solid*. To run this case, you can follow instructions from the Allrun file, or just type *./Allrun* or *./Allrun parallel*.

# Implementing one-way FSI for a wind turbine

## Creating one-way FSI solver

Despite one-way FSI is present as a separate solver, it is limited to work with fixed dynamic mesh. Therefore, we are modifying two-way FSI solver to run it with rotating mesh. Our new solver would be named *oneWayFSI*. You could use your own name, however, be sure to change it everywhere else too.

The first step is to copy existing solver. To do this, we need to duplicate the

*solids4Foam/src/solids4FoamModels/fluidSolidInterfaces/AitkenCouplingInterface* and rename it to any other solver. As a result, we will have two folders: *AitkenCouplingInterface* and *AitkenCouplingInterface (duplicate)*. The duplicate folder should be renamed to *oneWayFSI* (*note: be sure, that oneWayFSI folder is located at*

*solids4Foam/src/solids4FoamModels/fluidSolidInterfaces /*).

Inside the oneWayFSI will be 3 files:

* *AitkenCouplingInterface.C*
* *AitkenCouplingInterface.dep*
* *AitkenCouplingInterface.H*

The *.dep* file should be deleted. It will be generated during code compiling stage. The rest two file should be renamed to *oneWayFSI.C* and *oneWayFSI.H*. In both files, the *AitkenCouplingInterface* should be replaced with *oneWayFSI* (*note: the TypeName("Aitken"); should also be changed to TypeName("oneWayFSI"); in the .H file)* Moreover, because we are using one-way FSI, following part of the .C file should be removed:

* updateDisplacement();
* moveFluidMesh();
* residualNorm = updateResidual();

The last step is to include our new solver to compiler list. Add the following line:

To the

And run the compiler at

By typing

## Modifying existing case

For this stage, it is assumed that both fluid and solid meshes are created. The fluid and solid meshed should be placed to *constant/fluid/polyMesh* and *constant/solid/polyMesh* folders respectively.

In the *constant/fluid/* folder, the following files should be modified:

* *dynamicMeshDict*

*turboFvMeshCoeffs{*

*coordinateSystem{*

*type cylindrical;*

*origin (0 0 0);*

*axis (0 1 0);*

*direction (1 0 0); }*

*rpm {rotatingZone -12; }*

*slider {}}*

where *rotatingZone* is the name of a zone.

* *fluidProperties, RASProperties, turbulenceProperties*

These file control the turbulence model. It should be changed to icoFoam for testing with laminar flow, and to pimpleFoam for accurate simulations.

* transportProperties

Physical properties of fluid. Should be changed to air

In the *constant/solid/* folder, the following files should be modified:

* *dynamicMeshDict*

*dynamicFvMesh solidBodyMotionFvMesh;*

*solidBodyMotionFvMeshCoeffs{*

*solidBodyMotionFunction constantVelocity;*

*constantVelocityCoeffs {*

*origin (0 0 0);*

*translationalVelocity (0 0 0);*

*rotationalVelocity (0 -72 0); startMotionTime 0; inDegrees yes;}}*

where *rotationalVelocity* is measured in degrees per second.

* *g*

This file controls gravitational acceleration.

* mechanicalProperties

Contains material properties.

* solidProperties

This file controls deformation type, like linear or elastic, number of solid iterations per timestep, tolerance, etc.

The *constant* folder also contains two more files:

* physicalProperties

Switches between pure fluid, pure solid, or fluid solid interactions (*note, if you will use pure fluid or pure solid case, the fluid and solid folders should no longer be present and all files should be located in initial folder, like common OpenFoam case*)

* fsiProperties

*fluidSolidInterface* should be changed from *Aitken* to *oneWayFSI*. *oneWayFSICoeff* should also be added to the rest coefficient names.

*solidPatch interfaceName;* and *fluidPatch interfaceName;* should contain names of FSI interfaces. For more than one interfaces, use *solidPatches (interfaceName1 interfaceName2);* and *fluidPatches (interfaceName1 interfaceName2);*

*interfaceTransferMethod* should be set to *GGI*. The *directMap* requires absolute perfectly align interfaces, and *RBF* takes a lot of memory for fine meshes.

The *0/fluid* folder contains the same boundary conditions, as common CFD simulations. However, the interface patch should have the following boundary conditions:

*interfaceName*

*{*

*type newMovingWallVelocity;*

*value uniform (0 0 0);*

*}*

The *0/solid* folder contain *D* or *DD* file, depending on solid solver (*note: sometimes it is better to have both files*). The boundary conditions of these files should be identical to each other and look like this:

interfaceName

{

type solidTraction;

traction uniform ( 0 0 0 );

pressure uniform 0;

value uniform (0 0 0);

}

staticPatchName

{

type fixedDisplacement;

value uniform (0 0 0);

}

The *system* folder contains the discretization and solver parameters, these should be changed according to the required simulation.

To run parallel case, the solid part is similar to common CFD simulation. However, because we are using GGI to transfer data from static to rotation domain, the *system/fluid/decomposePartDict* should look like this (*note: GGI zones are not the same as GGI patchs*):

numberOfSubdomains 8;

method patchConstrained;

globalFaceZones

(

nameOfGGIRotatingZone1

nameOfGGIStaticZone1

nameOfGGIRotatingZone2

nameOfGGIStaticZone2

);

patchConstrainedCoeffs

{

method metis;

numberOfSubdomains 8;

patchConstraints

(

(nameOfGGIRotatingPach1 1)

(nameOfGGIStaticPach1 1)

(nameOfGGIRotatingPach2 2)

(nameOfGGIStaticPach2 2)

);

}

metisCoeffs

{

processorWeights

(

1

1

1

1

1

1

1

1

);

}

distributed no;

roots ();

## Running a case

Similar to the test case, simulation could be started with *solids4Foam* command. However, it is possible to run it in parallel. To do this, the following command should be used:

* *decomposePar -region fluid*
* *decomposePar -region solid*
* *mpirun -np 8 solids4Foam -parallel*
* *reconstructPar -region fluid*
* *recontructPar -region solid*

# Codes

## oneWayFSI.C

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

========= |

\\ / F ield | OpenFOAM: The Open Source CFD Toolbox

\\ / O peration |

\\ / A nd | Copyright held by original author

\\/ M anipulation |

-------------------------------------------------------------------------------

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 2 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, write to the Free Software Foundation,

Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

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

#include "OneWayFSI.H"

#include "addToRunTimeSelectionTable.H"

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

namespace Foam

{

namespace fluidSolidInterfaces

{

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* Static Data Members \* \* \* \* \* \* \* \* \* \* \* \* \* //

defineTypeNameAndDebug(OneWayFSI, 0);

addToRunTimeSelectionTable

(

physicsModel, OneWayFSI, fluidSolidInteraction

);

addToRunTimeSelectionTable

(

fluidSolidInterface, OneWayFSI, dictionary

);

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* Constructors \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

OneWayFSI::OneWayFSI

(

Time& runTime,

const word& region

)

:

fluidSolidInterface(typeName, runTime, region),

relaxationFactor\_

(

fsiProperties().lookupOrDefault<scalar>("relaxationFactor", 0.01)

),

predictSolid\_(fsiProperties().lookupOrDefault<bool>("predictSolid", true)),

aitkenRelaxationFactors\_(nGlobalPatches(), relaxationFactor\_)

{}

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

bool OneWayFSI::evolve()

{

initializeFields();

updateInterpolatorAndGlobalPatches();

scalar residualNorm = 0;

if (predictSolid\_)

{

updateForce();

solid().evolve();

residualNorm =

updateResidual();

}

do

{

outerCorr()++;

// Transfer the displacement from the solid to the fluid

//updateDisplacement(); //!!!

// Move the fluid mesh

//moveFluidMesh(); //!!!

// Solve fluid

fluid().evolve();

// Transfer the force from the fluid to the solid

updateForce();

// Solve solid

solid().evolve();

// Calculate the FSI residual

//residualNorm = updateResidual(); //!!!

// Optional: write residuals to file //!!!

//if (writeResidualsToFile() && Pstream::master())

//{

// residualFile()

// << runTime().value() << " "

// << outerCorr() << " "

// << residualNorm << endl;

//}

}

while (residualNorm > outerCorrTolerance() && outerCorr() < nOuterCorr());

solid().updateTotalFields();

return 0;

}

void OneWayFSI::updateDisplacement()

{

Info<< nl << "Time = " << fluid().runTime().timeName()

<< ", iteration: " << outerCorr() << endl;

if (outerCorr() < 3)

{

Info<< "Current fsi under-relaxation factor (fixed): "

<< relaxationFactor\_ << endl;

forAll(fluid().globalPatches(), interfaceI)

{

fluidZonesPointsDisplsPrev()[interfaceI] =

fluidZonesPointsDispls()[interfaceI];

fluidZonesPointsDispls()[interfaceI] +=

relaxationFactor\_\*residuals()[interfaceI];

}

}

else

{

forAll(fluid().globalPatches(), interfaceI)

{

aitkenRelaxationFactors\_[interfaceI] =

-aitkenRelaxationFactors\_[interfaceI]

\*(

sum

(

residualsPrev()[interfaceI]

& (residuals()[interfaceI] - residualsPrev()[interfaceI])

)

/(

sum

(

(

residuals()[interfaceI]

- residualsPrev()[interfaceI]

)

& (

residuals()[interfaceI]

- residualsPrev()[interfaceI]

)

)

)

);

if (Pstream::parRun())

{

if (!Pstream::master())

{

aitkenRelaxationFactors\_[interfaceI] = 0.0;

}

// Pass to all procs

reduce(aitkenRelaxationFactors\_[interfaceI], sumOp<scalar>());

}

aitkenRelaxationFactors\_[interfaceI] =

mag(aitkenRelaxationFactors\_[interfaceI]);

if (aitkenRelaxationFactors\_[interfaceI] > 1)

{

// PC: in this case, would 1.0 be a better option?

// Of course, the current option is more more stable

// aitkenRelaxationFactors\_[interfaceI] = relaxationFactor\_;

aitkenRelaxationFactors\_[interfaceI] = 1.0;

}

Info<< "Current fsi under-relaxation factor (Aitken) of "

<< fluidMesh().boundary()

[

fluid().globalPatches()[interfaceI].patch().index()

].name()

<< ": " << aitkenRelaxationFactors\_[interfaceI] << endl;

fluidZonesPointsDisplsPrev()[interfaceI] =

fluidZonesPointsDispls()[interfaceI];

fluidZonesPointsDispls()[interfaceI] +=

aitkenRelaxationFactors\_[interfaceI]\*residuals()[interfaceI];

}

}

// Update movingWallPressure boundary conditions, if found

fluidSolidInterface::updateMovingWallPressureAcceleration();

// Make sure that displacement on all processors is equal to one

// calculated on master processor

fluidSolidInterface::syncFluidZonePointsDispl(fluidZonesPointsDispls());

}

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

} // End namespace fluidSolidInterfaces

} // End namespace Foam

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //

## oneWayFSI.H

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

========= |

\\ / F ield | OpenFOAM: The Open Source CFD Toolbox

\\ / O peration |

\\ / A nd | Copyright held by original author

\\/ M anipulation |

-------------------------------------------------------------------------------

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 2 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, write to the Free Software Foundation,

Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Class

AitkenCouplingInterface

Description

Strong Dirichlet-Neumann coupling with Aitken accelerated under-relaxation.

Author

Zeljko Tukovic, FSB Zagreb. All rights reserved.

Philip Cardiff, UCD. All rights reserved.

SourceFiles

AitkenCouplingInterface.C

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

#ifndef OneWayFSI\_H

#define OneWayFSI\_H

#include "fluidSolidInterface.H"

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

namespace Foam

{

namespace fluidSolidInterfaces

{

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

Class AitkenCouplingInterface Declaration

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

class OneWayFSI

:

public fluidSolidInterface

{

// Private data

//- Fixed under-relaxation factor

const scalar relaxationFactor\_;

//- Predict solid

const bool predictSolid\_;

//- List of Aitken under-relaxation factors

List<scalar> aitkenRelaxationFactors\_;

// Private Member Functions

//- Disallow default bitwise copy construct

OneWayFSI(const OneWayFSI&);

//- Disallow default bitwise assignment

void operator=(const OneWayFSI&);

public:

//- Runtime type information

TypeName("OneWayFSI");

// Constructors

//- Construct from components

OneWayFSI

(

Time& runTime,

const word& region = dynamicFvMesh::defaultRegion

);

// Destructor

virtual ~OneWayFSI()

{}

// Member Functions

// Edit

//- Evolve the interface

virtual bool evolve();

//- Calculate interface displacement

virtual void updateDisplacement();

};

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

} // End namespace fluidSolidInterfaces

} // End namespace Foam

// \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* //

#endif

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* //