Description of matrix discretisation with focus on the Gauss laplacian discretisation operator and how to create a modified version

Jesper Roland Kjærgaard Qwist

Section for Fluid Mechanics, Coastal and Maritime Engineering,
Technical University of Denmark (DTU),
Lyngby, Denmark

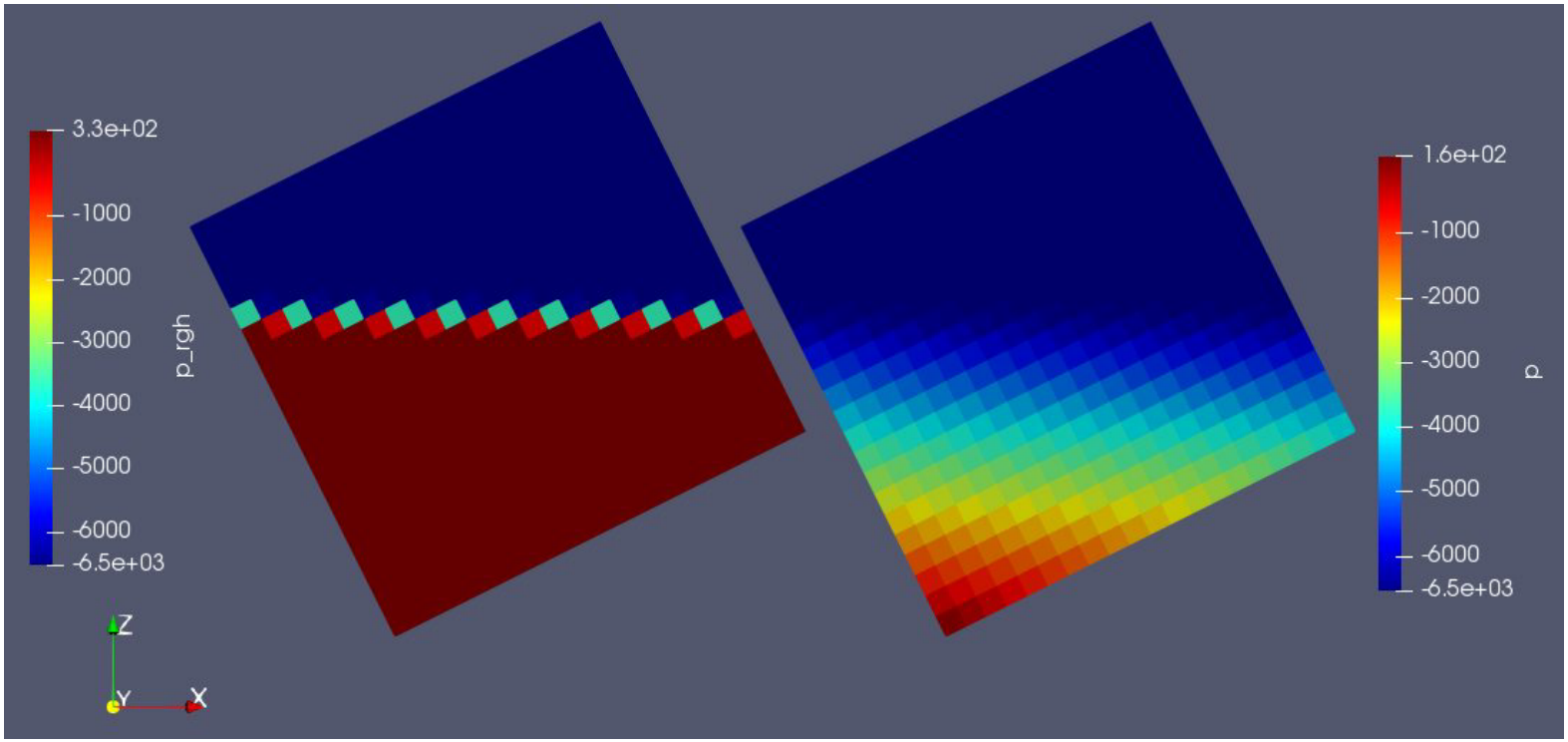
Prepared for OpenFOAM-v1906

2019-11-26

Outline

- Why do I need to modify the Laplacian operator?
- Concept of Ghost Fluid Method (GFM)
- Create copy of `gaussLaplacianScheme`
- Remove template functionality in `myGaussScalarLaplacianScheme`
- Implement GFM-method in Gauss Laplacian operator `GFMGaussLaplacianScheme`
- Still water level with GFM-method in Gauss Laplacian operator

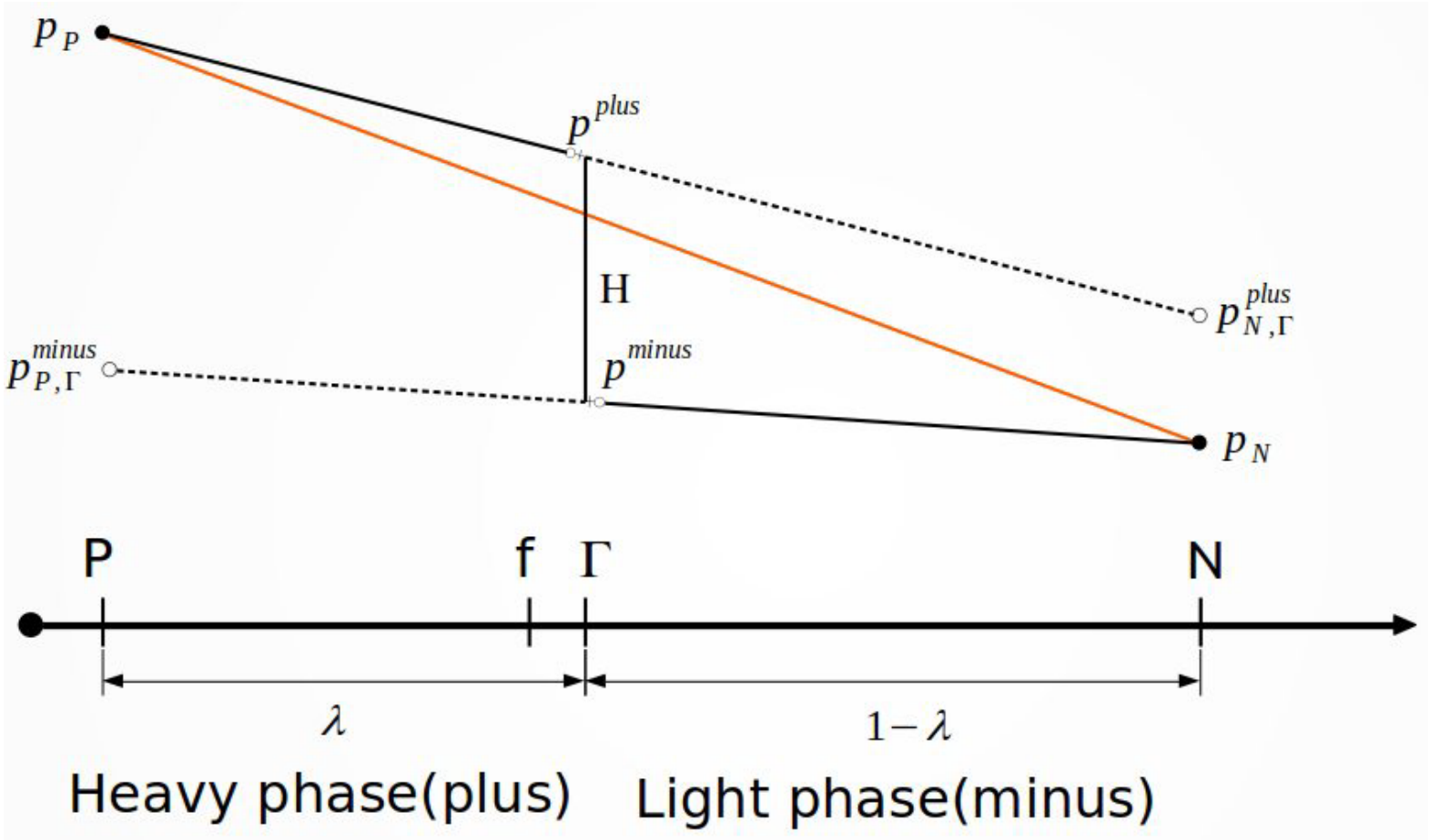
Why do I need to modify the Laplacian operator?



Dynamic (p_rgh) and total pressure (p) fields for a static free surface discretised with standard Gauss Laplacian scheme in InterIsoFoam.

$$p\_rgh = p - \rho \mathbf{g} \bullet \mathbf{x}$$

Concept of Ghost Fluid Method (GFM)



One-sided interpolation for a wet owner cell.

Create copy of gaussLaplacianScheme

Download `slideSkip0_files_JesperRolandKjrgaardQwist.tgz`, unpack, open terminal in folder, source OpenFOAM and execute Allcopy.

Copy original source files:

```
OFv1906
src
mkdir $WM_PROJECT_USER_DIR/src
cp -r --parents finiteVolume/finiteVolume/laplacianSchemes/gaussLaplacianScheme \
$WM_PROJECT_USER_DIR/src
```

Rename files and folders:

```
//Break column pattern (Latex/PDF issue).
cd $WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume/laplacianSchemes
mv gaussLaplacianScheme myGaussLaplacianScheme
cd myGaussLaplacianScheme
mv gaussLaplacianScheme.C myGaussLaplacianScheme.C
mv gaussLaplacianScheme.H myGaussLaplacianScheme.H
mv gaussLaplacianSchemes.C myGaussLaplacianSchemes.C
```

Change class name in source files:

```
sed -i s/gaussLaplacianScheme/myGaussLaplacianScheme/g \
myGaussLaplacianScheme.H
sed -i s/gaussLaplacianScheme/myGaussLaplacianScheme/g \
myGaussLaplacianScheme.C
sed -i s/gaussLaplacianScheme/myGaussLaplacianScheme/g \
myGaussLaplacianSchemes.C
```

Change scheme keyword:

```
sed s/'TypeName("Gauss");'/'TypeName("myGauss");'/g myGaussLaplacianScheme.H
\\ CHECK IF THE SUBSTITUTION IS OK, THEN DO
sed -i s/'TypeName("Gauss");'/'TypeName("myGauss");'/g myGaussLaplacianScheme.H
```

Create a Make folder for library:

```
mkdir $WM_PROJECT_USER_DIR/src/finiteVolume/Make
```

Create `Make/files`:

```
vi $WM_PROJECT_USER_DIR/src/finiteVolume/Make/files
```

The content of `Make/files` should be:

```
laplacianSchemes = finiteVolume/laplacianSchemes
$(laplacianSchemes)/myGaussLaplacianScheme/myGaussLaplacianSchemes.C
LIB = $(FOAM_USER_LIBBIN)/libmyFiniteVolume
```

Save and close `files` by typing command: `SHIFT+zz`.

Create `Make/options`:

```
vi $WM_PROJECT_USER_DIR/src/finiteVolume/Make/options
```

The content of `Make/options` should be:

```
EXE_INC = \
    -I$(LIB_SRC)/fileFormats/lnInclude \
    -I$(LIB_SRC)/surfMesh/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude \
    -I$(LIB_SRC)/finiteVolume/lnInclude

LIB_LIBS = \
    -lOpenFOAM \
    -lfileFormats \
    -lsurfMesh \
    -lmeshTools \
    -lfiniteVolume
```

Save and close `options` by typing command: `SHIFT+zz`.

Compile myFiniteVolume library:

```
wmake $WM_PROJECT_USER_DIR/src/finiteVolume
```

Test myGaussLaplacianScheme:

```
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity $FOAM_RUN/myGaussCavity
```

Open scheme settings in vi:

```
vi $FOAM_RUN/myGaussCavity/system/fvSchemes
```

Change laplacianSchemes settings to:

```
laplacianSchemes
{
    default none;
    laplacian(nu,U) Gauss linear orthogonal;
    laplacian((1|A(U)),p) myGauss linear orthogonal;
}
```

Insert at `libs ("libmyFiniteVolume.so");` the end of `system/controlDict`

```
sed -i '$a libs ("libmyFiniteVolume.so");' $FOAM_RUN/myGaussCavity/system/controlDict
```

To build the mesh and run the case execute:

```
blockMesh -case $FOAM_RUN/myGaussCavity >& $FOAM_RUN/myGaussCavity/log.blockMesh
icoFoam -case $FOAM_RUN/myGaussCavity >& $FOAM_RUN/myGaussCavity/log.icoFoam
```

Now open the log file log.icoFoam and verify that it works.

```
vi $FOAM_RUN/myGaussCavity/log.icoFoam
```

```
Time = 0.5

Courant Number mean: 0.222158 max: 0.852134
smoothSolver:  Solving for Ux, Initial residual = 2.3091e-07, Final residual = 2.3091e-07, No Iterations 0
smoothSolver:  Solving for Uy, Initial residual = 5.0684e-07, Final residual = 5.0684e-07, No Iterations 0
DICPCG:  Solving for p, Initial residual = 8.63844e-07, Final residual = 8.63844e-07, No Iterations 0
time step continuity errors : sum local = 8.8828e-09, global = 4.94571e-19, cumulative = 1.10417e-17
DICPCG:  Solving for p, Initial residual = 9.59103e-07, Final residual = 9.59103e-07, No Iterations 0
time step continuity errors : sum local = 9.66354e-09, global = 1.13175e-18, cumulative = 1.21735e-17
ExecutionTime = 0.09 s  ClockTime = 0 s

End
```

Remove template functionality in myGaussLaplacianScheme

Create copy of myGaussLaplacianScheme called myScalarGaussLaplacianScheme with keyword "myScalarGauss":

```
cp -r $WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume/laplacianSchemes/myGaussLaplacianScheme \
$WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume/laplacianSchemes/myScalarGaussLaplacianScheme
cd $WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume/laplacianSchemes/myScalarGaussLaplacianScheme
mv myGaussLaplacianScheme.C myScalarGaussLaplacianScheme.C
mv myGaussLaplacianScheme.H myScalarGaussLaplacianScheme.H
mv myGaussLaplacianSchemes.C myScalarGaussLaplacianSchemes.C
sed -i s/myGaussLaplacianScheme/myScalarGaussLaplacianScheme/g myScalarGaussLaplacianScheme.H
sed -i s/myGaussLaplacianScheme/myScalarGaussLaplacianScheme/g myScalarGaussLaplacianScheme.C
sed -i s/myGaussLaplacianScheme/myScalarGaussLaplacianScheme/g myScalarGaussLaplacianSchemes.C
sed -i s/'TypeName("myGauss");'/'TypeName("myScalarGauss");'/g myScalarGaussLaplacianScheme.H
```

The modification steps are

- Remove the declaration and definition of the function `gammaSnGradCorr`.

```
sed -i -e '66,71d' myScalarGaussLaplacianScheme.H
sed -i -e '91,132d' myScalarGaussLaplacianScheme.C
```

- Remove `template<class Type, class GType>` in source and header

```
sed -i '/template<class Type, class GType>/d' myScalarGaussLaplacianScheme.H
sed -i '/template<class Type, class GType>/d' myScalarGaussLaplacianScheme.C
```

- Swap `myScalarGaussLaplacianScheme<Type, GType>` with `myScalarGaussLaplacianScheme` in source

```
sed -i s/'myScalarGaussLaplacianScheme<Type, GType>'/'myScalarGaussLaplacianScheme'/g \
myScalarGaussLaplacianScheme.C
```

- Remove inclusion of `myScalarGaussLaplacianScheme.C` in header file.

```
sed -i -e '175,180d' myScalarGaussLaplacianScheme.H
```

- Remove macro `defineFvmLaplacianScalarGamma` and the calls to this macro.

```
sed -i -e '137,164d' myScalarGaussLaplacianScheme.H
```

Continues on next slide ...

- Copy implementation of

```
myScalarGaussLaplacianScheme::fvcLaplacian
(
    const GeometricField<scalar, fvsPatchField, surfaceMesh>& gamma,
    const GeometricField<scalar, fvPatchField, volMesh>& vf
)
```

and

```
myScalarGaussLaplacianScheme::fvmLaplacian
(
    const GeometricField<scalar, fvsPatchField, surfaceMesh>& gamma,
    const GeometricField<scalar, fvPatchField, volMesh>& vf
)
```

from `myScalarGaussLaplacianSchemes.C` to `myScalarGaussLaplacianScheme.C` and remove line shifts `\`.

- Remove `myScalarGaussLaplacianSchemes.C`
- Replace template parameter `Type` and `GType` with `scalar` in header file.

```
sed -i s/'<Type, GType>'/'<scalar, scalar>'/g myScalarGaussLaplacianScheme.H
sed -i s/'<GType'/'<scalar'/g myScalarGaussLaplacianScheme.H
sed -i s/'<Type'/'<scalar'/g myScalarGaussLaplacianScheme.H
```

- Replace template parameter `Type` and `GType` with `scalar` in source file.

```
sed -i s/'<GType'/'<scalar'/g myScalarGaussLaplacianScheme.C
sed -i s/'<Type'/'<scalar'/g myScalarGaussLaplacianScheme.C
```

- Add makeMyGaussLaplacianScheme(SS)

```
// Add the patch constructor functions to the hash tables
#define makeMyGaussLaplacianScheme(SS)\
    typedef Foam::scalar Type;\
    typedef Foam::scalar GType;\
    typedef Foam::fv::SS SS##Type##GType;\
    defineTypeName(SS##Type##GType);\
    namespace Foam\
    {\
        namespace fv\
        {\
            typedef SS SS##Type##GType;\
\
            laplacianScheme<Type, GType>::\
                addIstreamConstructorToTable<SS>\
                add##SS##Type##GType##IstreamConstructorToTable_;\
        }\
    }

// Define symbol lookup:
//  SS: Name of current class
makeMyGaussLaplacianScheme(myScalarGaussLaplacianScheme)
```

after `#include "fvMatrices.H"` and before `namespace Foam` in `myScalarGaussLaplacianScheme.C`.

- Open "Make/files"

```
vi $WM_PROJECT_USER_DIR/src/finiteVolume/Make/files
```

and modify the content to

```
laplacianSchemes = finiteVolume/laplacianSchemes
$(laplacianSchemes)/myGaussLaplacianScheme/myGaussLaplacianSchemes.C
$(laplacianSchemes)/myScalarGaussLaplacianScheme/myScalarGaussLaplacianScheme.C
LIB = $(FOAM_USER_LIBBIN)/libmyFiniteVolume
```

- Compile:

```
wmake $WM_PROJECT_USER_DIR/src/finiteVolume
```

The library is located at:

```
$FOAM_USER_LIBBIN/libmyFiniteVolume.so
```

Test implementation:

```
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity $FOAM_RUN/myScalarGaussCavity
sed -i '/Gauss linear orthogonal;/c \
default none; \
laplacian(nu,U) Gauss linear orthogonal; \
laplacian((1|A(U)),p) myScalarGauss linear orthogonal;' \
$FOAM_RUN/myScalarGaussCavity/system/fvSchemes
sed -i '$a libs ("libmyFiniteVolume.so");' $FOAM_RUN/myScalarGaussCavity/system/controlDict
blockMesh -case $FOAM_RUN/myScalarGaussCavity
icoFoam -case $FOAM_RUN/myScalarGaussCavity
```

Implement GFM-method in class `GFMGaussLaplacianScheme`

Self study:

Study the interfaceJump class. The general functionality is described in the report, and there are comments in the source files interfaceJump.H and interfaceJump.C.

Hands-on:

I will show how to

- add interfaceJump to myFiniteVolume library
- make a copy of myScalarGaussLaplacianScheme called GFMGaussLaplacianScheme
- implementation GFM method in GFMGaussLaplacianScheme

Place the class folder "interfaceJump" in the folder

```
$WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume
```

Create copy of myScalarGaussLaplacianScheme called GFMGaussLaplacianScheme:

```
cp -r $WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume/laplacianSchemes/myScalarGaussLaplacianScheme \
$WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume/laplacianSchemes/GFMGaussLaplacianScheme
cd $WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume/laplacianSchemes/GFMGaussLaplacianScheme
mv myScalarGaussLaplacianScheme.C GFMGaussLaplacianScheme.C
mv myScalarGaussLaplacianScheme.H GFMGaussLaplacianScheme.H
sed -i s/myScalarGaussLaplacianScheme/GFMGaussLaplacianScheme/g GFMGaussLaplacianScheme.H
sed -i s/myScalarGaussLaplacianScheme/GFMGaussLaplacianScheme/g GFMGaussLaplacianScheme.C
sed -i s/'TypeName("myScalarGauss");'/'TypeName("GFMGauss");'/g GFMGaussLaplacianScheme.H
sed -i '/myScalarGaussLaplacianScheme.C/a \
$(laplacianSchemes)/GFMGaussLaplacianScheme/GFMGaussLaplacianScheme.C\
interfaceJump = finiteVolume/interfaceJump\
$(interfaceJump)/interfaceJump.C' $WM_PROJECT_USER_DIR/src/finiteVolume/Make/files
wmake $WM_PROJECT_USER_DIR/src/finiteVolume
```

Test that new copy works:

```
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity $FOAM_RUN/GFMGaussCavity
sed -i '/Gauss linear orthogonal;/c \
default none; \
laplacian(nu,U) Gauss linear orthogonal; \
laplacian((1|A(U)),p) GFMGauss linear orthogonal;' \
$FOAM_RUN/GFMGaussCavity/system/fvSchemes
sed -i '$a libs ("libmyFiniteVolume.so");' $FOAM_RUN/GFMGaussCavity/system/controlDict
blockMesh -case $FOAM_RUN/GFMGaussCavity
icoFoam -case $FOAM_RUN/GFMGaussCavity
```

GFMGaussLaplacianScheme.H

Open file:

```
vi $WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume/laplacianSchemes\
/GFMGaussLaplacianScheme/GFMGaussLaplacianScheme.H
```

After `#include "laplacianScheme.H"` add `#include "interfaceJump.H"`

After:

```
class myGaussLaplacianScheme
:
    public fv::laplacianScheme<scalar, scalar>
{
```

insert:

```
    // Private Member Data

        //- VOF field reference
        const volScalarField& alpha1_;

        //- Density field reference
        const volScalarField& rho_;
```

Change constructors by inserting:

```
,
alpha1_(mesh.lookupObject<volScalarField>("alpha.water")),
rho_(mesh.lookupObject<volScalarField>("rho"))
```

after `laplacianScheme<scalar, scalar>(mesh)`, `laplacianScheme<scalar, scalar>(mesh, is)` and `laplacianScheme<scalar, scalar>(mesh, igs, sngs)`.

Declare new inputs to fvmLaplacianUncorrected function by adding

```
,
const volScalarField& alpha1,
const volScalarField& rho
```

after `const GeometricField<scalar, fvPatchField, volMesh>&`.

GFMGaussLaplacianScheme.C

Open file:

```
vi $WM_PROJECT_USER_DIR/src/finiteVolume/finiteVolume/laplacianSchemes\
/GFMGaussLaplacianScheme/GFMGaussLaplacianScheme.C
```

Declare new inputs to fvmLaplacianUncorrected function by adding

```
,
const volScalarField& alpha1,
const volScalarField& rho
```

after `const GeometricField<scalar, fvPatchField, volMesh>& vf`.

Construct interfaceJump object boolean list of surface faces by adding

```
interfaceJump intface(vf,alpha1);
const boolList& sFaces = intface.sFaces();
```

before `tmp<fvMatrix<scalar>> tfvm`.

Change dimension of system matrix by appending `/dimDensity` to line
`deltaCoeffs.dimensions()*gammaMagSf.dimensions()*vf.dimensions()`.

Start implementation of new matrix assembly by substituting

```
fvm.upper() = deltaCoeffs.primitiveField()*gammaMagSf.primitiveField();
fvm.negSumDiag();
```

with

```
//- Build system matrix for internal faces
for(label facei=0; facei<fvm.lduAddr().upperAddr().size(); facei++)
{
    label owner = fvm.lduAddr().upperAddr()[facei]; // P
    label neighbour = fvm.lduAddr().lowerAddr()[facei]; // N
    if (sFaces[facei])
    {// This is a special surface face.
        if (alpha1[owner] > 0.5)
        {// 1) P is wet:
        }
        else
        {// 2) P is dry:
        }
    }
    else
    {// This is regular internal face.
        if (alpha1[owner] > 0.5)
        {// 3) P & N are both wet:
        }
        else
        {// 4) P & N are both dry:
        }
    }
}
```

Inside brackets with `// 1) P is wet:`

```
// - Looking from P -> N
fvm.diag()[owner] -= deltaCoeffs.primitiveField()[facei]
                   * gammaMagSf.primitiveField()[facei]
                   * intface.betaPlus().value()
                   * intface.betaMinus().value()
                   / intface.betaBarWet(facei);
fvm.upper()[facei] += deltaCoeffs.primitiveField()[facei]
                    * gammaMagSf.primitiveField()[facei]
                    * intface.betaPlus().value()
                    * intface.betaMinus().value()
                    / intface.betaBarWet(facei);
fvm.source()[owner] += deltaCoeffs.primitiveField()[facei]
                     * gammaMagSf.primitiveField()[facei]
                     * intface.betaPlus().value()
                     * intface.betaMinus().value()
                     / intface.betaBarWet(facei)
                     * intface.H(facei).value();
// - Looking from N -> P
fvm.diag()[neighbour] -= deltaCoeffs.primitiveField()[facei]
                       * gammaMagSf.primitiveField()[facei]
                       * intface.betaPlus().value()
                       * intface.betaMinus().value()
                       / intface.betaBarWet(facei);
fvm.source()[neighbour] -= deltaCoeffs.primitiveField()[facei]
                         * gammaMagSf.primitiveField()[facei]
                         * intface.betaPlus().value()
                         * intface.betaMinus().value()
                         / intface.betaBarWet(facei)
                         * intface.H(facei).value();
```

Inside brackets with `// 2) P is dry:`

```
// - Looking from P -> N
fvm.diag()[owner] -= deltaCoeffs.primitiveField()[facei]
                   * gammaMagSf.primitiveField()[facei]
                   * intface.betaPlus().value()
                   * intface.betaMinus().value()
                   / intface.betaBarDry(facei);
fvm.upper()[facei] += deltaCoeffs.primitiveField()[facei]
                    * gammaMagSf.primitiveField()[facei]
                    * intface.betaPlus().value()
                    * intface.betaMinus().value()
                    / intface.betaBarDry(facei);
fvm.source()[owner] -= deltaCoeffs.primitiveField()[facei]
                     * gammaMagSf.primitiveField()[facei]
                     * intface.betaPlus().value()
                     * intface.betaMinus().value()
                     / intface.betaBarDry(facei)
                     * intface.H(facei).value();
// - Looking from N -> P
fvm.diag()[neighbour] -= deltaCoeffs.primitiveField()[facei]
                       * gammaMagSf.primitiveField()[facei]
                       * intface.betaPlus().value()
                       * intface.betaMinus().value()
                       / intface.betaBarDry(facei);
fvm.source()[neighbour] += deltaCoeffs.primitiveField()[facei]
                         * gammaMagSf.primitiveField()[facei]
                         * intface.betaPlus().value()
                         * intface.betaMinus().value()
                         / intface.betaBarDry(facei)
                         * intface.H(facei).value();
```

Inside brackets with `// 3) P & N are both wet:`

```
// Assign contributions to the diagonal matrix coefficient:
// - Looking from P -> N
fvm.diag()[owner] -= deltaCoeffs.primitiveField()[facei]
                   * gammaMagSf.primitiveField()[facei]
                   * intface.betaPlus().value();

// - Looking from N -> P
fvm.diag()[neighbour] -= deltaCoeffs.primitiveField()[facei]
                       * gammaMagSf.primitiveField()[facei]
                       * intface.betaPlus().value();

// Assign the matrix coefficient in the upper triangle:
fvm.upper()[facei] += deltaCoeffs.primitiveField()[facei]
                    * gammaMagSf.primitiveField()[facei]
                    * intface.betaPlus().value();
```

Inside brackets with `// 4) P & N are both dry:`

```
// Assign contributions to the diagonal matrix coefficient:
// - Looking from P -> N
fvm.diag()[owner] -= deltaCoeffs.primitiveField()[facei]
                   * gammaMagSf.primitiveField()[facei]
                   * intface.betaMinus().value();

// - Looking from N -> P
fvm.diag()[neighbour] -= deltaCoeffs.primitiveField()[facei]
                       * gammaMagSf.primitiveField()[facei]
                       * intface.betaMinus().value();

// Assign the matrix coefficient in the upper triangle:
fvm.upper()[facei] += deltaCoeffs.primitiveField()[facei]
                    * gammaMagSf.primitiveField()[facei]
                    * intface.betaMinus().value();
```

The boundary conditions are modified by adding

```
const fvPatchScalarField& pRho = rho.boundaryField()[patchi];
```

after

```
const fvsPatchScalarField& pDeltaCoeffs =
    deltaCoeffs.boundaryField()[patchi];
```

and substitute `pGamma` with `pGamma/pRho` in

```
if (pvf.coupled())
{
    fvm.internalCoeffs()[patchi] =
        pGamma*pvf.gradientInternalCoeffs(pDeltaCoeffs);
    fvm.boundaryCoeffs()[patchi] =
       -pGamma*pvf.gradientBoundaryCoeffs(pDeltaCoeffs);
}
else
{
    fvm.internalCoeffs()[patchi] = pGamma*pvf.gradientInternalCoeffs();
    fvm.boundaryCoeffs()[patchi] = -pGamma*pvf.gradientBoundaryCoeffs();
}
```

In `fvcLaplacian(vf)` replace

```
tmp<GeometricField<scalar, fvPatchField, volMesh>> tLaplacian
(
    fvc::div(this->tsnGradScheme_().snGrad(vf)*mesh.magSf())
);
```

with

```
const volScalarField betavf(vf/rho_);
tmp<GeometricField<scalar, fvPatchField, volMesh>> tLaplacian
(
    fvc::div(this->tsnGradScheme_().snGrad(betavf)*mesh.magSf())
);
```

In `fvcLaplacian(gamma,vf)` replace

```
tmp<GeometricField<scalar, fvPatchField, volMesh>> tLaplacian
(
    fvc::div(gamma*this->tsnGradScheme_().snGrad(vf)*mesh.magSf())
);
```

with

```
const volScalarField betavf(vf/rho_);
tmp<GeometricField<scalar, fvPatchField, volMesh>> tLaplacian
(
    fvc::div(gamma*this->tsnGradScheme_().snGrad(betavf)*mesh.magSf())
);
```

In the function `fvmLaplacian` add new input variables to `fvmLaplacianUncorrected` by replacing

```
tmp<fvMatrix<scalar>> tfvm = fvmLaplacianUncorrected
(
    gammaMagSf,
    this->tsnGradScheme_().deltaCoeffs(vf),
    vf
);
```

with

```
tmp<fvMatrix<scalar>> tfvm = fvmLaplacianUncorrected
(
    gammaMagSf,
    this->tsnGradScheme_().deltaCoeffs(vf),
    vf,
    alpha1_,
    rho_
);
```

To account for density in non orthogonal correction add

```
    const volScalarField betavf(vf/rho_);
```

after the lines given by

```
if (this->tsnGradScheme_().corrected())
{
```

and continues on next slides ...

Replace `vf` with the new variable `betavf` in non orthogonal correction by replacing

```
gammaMagSf*this->tsnGradScheme_().correction(vf)
```

with

```
gammaMagSf*this->tsnGradScheme_().correction(betavf)
```

at 2 instances. Now save and exit the file.

We already included `interfaceJump` in the library so we can compile:

```
wmake $WM_PROJECT_USER_DIR/src/finiteVolume
```

Key features of laplacianVOFFoam

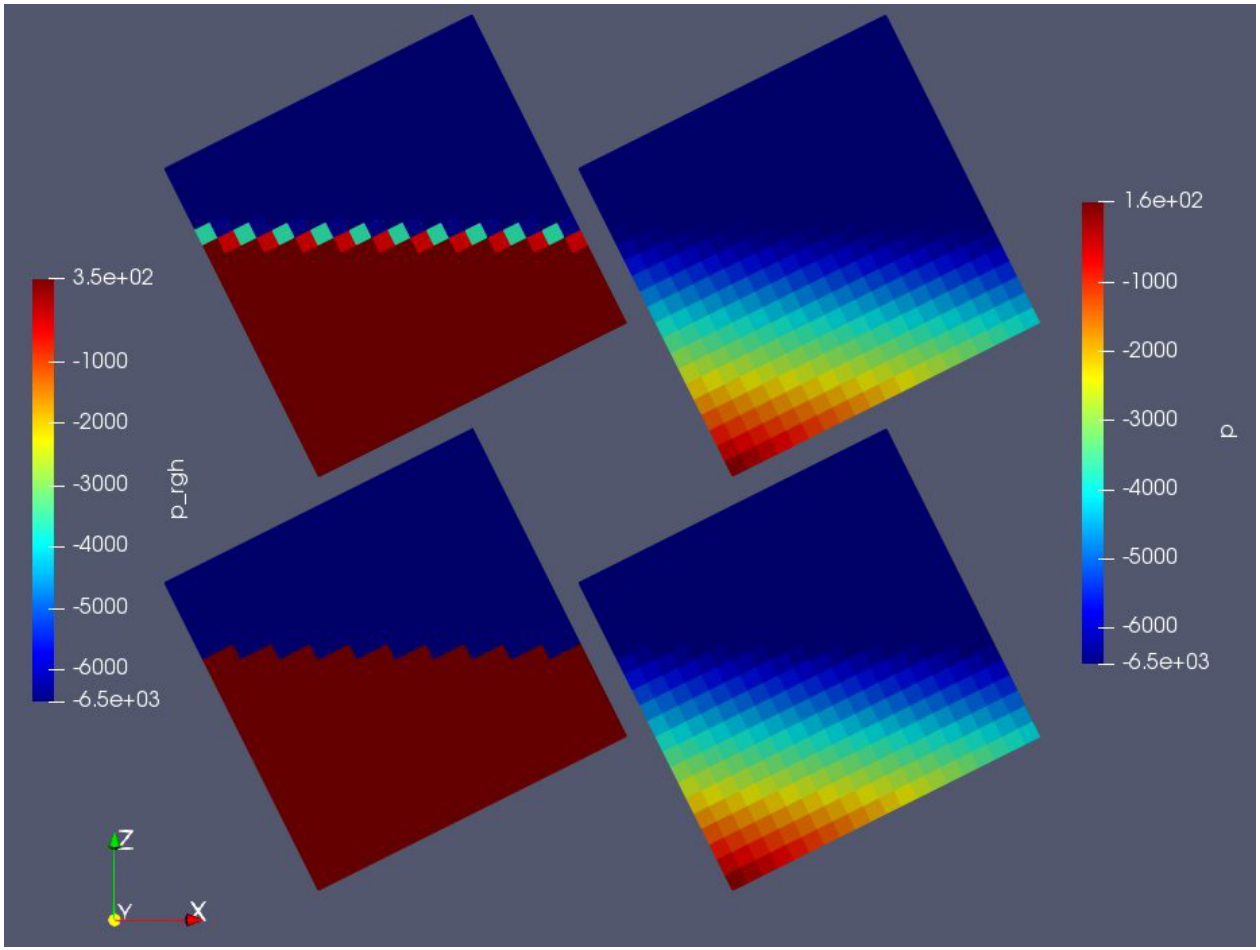
Self study:

Study laplacianVOFFoam.C solver and createFields.H, where the step interpolation of the density is performed.

Note:

The density step interpolation is not enforced by the GFM laplacian scheme. It must be performed by the application in which the scheme is used!

Still water level with GFM-method in Gauss Laplacian operator



Case:

Still free surface

Solver:

Row 1: interIsoFoam

Row 2: laplacianVOFFoam

Laplacian scheme:

Row 1: Gauss linear corrected

Row 2: GFMGauss linear corrected

Left column:

Dynamic pressure (p_rgh)

Right column:

Total pressure (p)