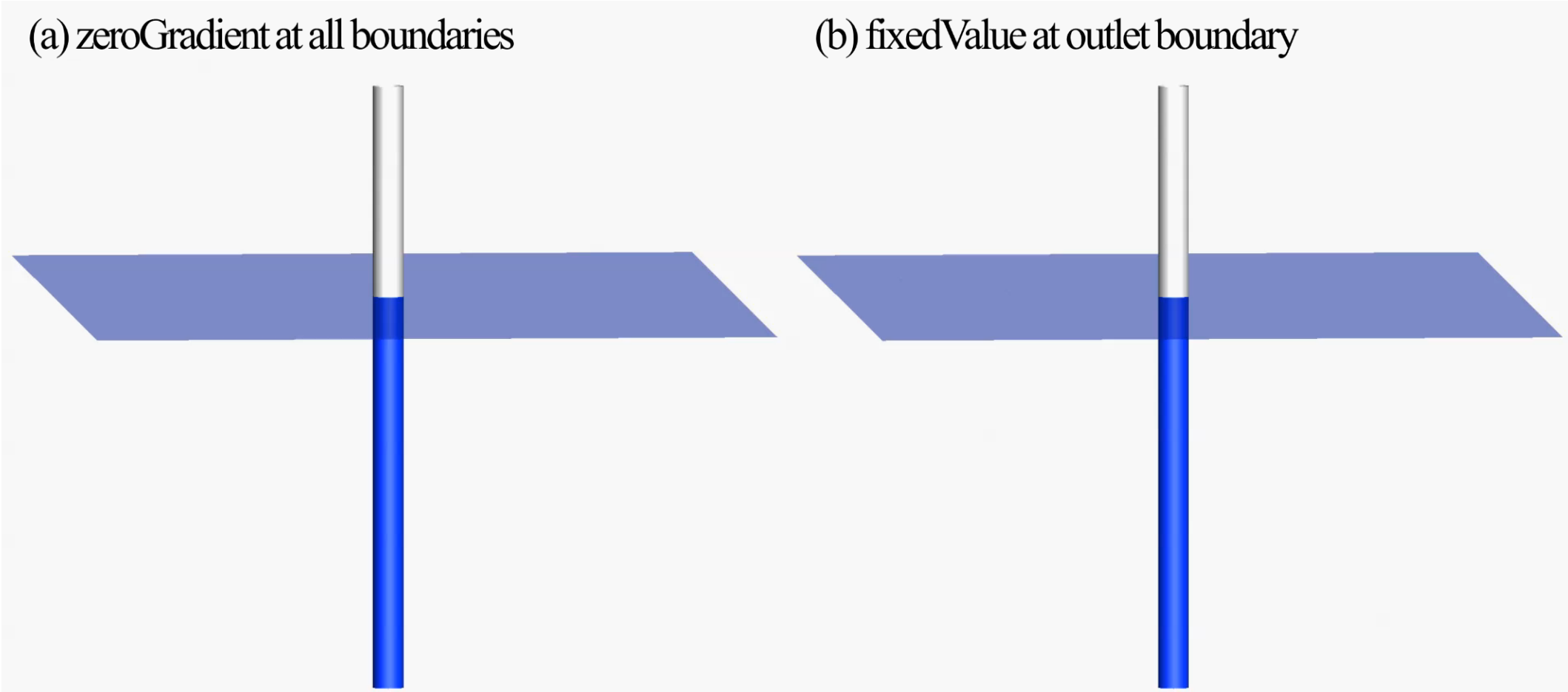
경계 조건에 따른 자유 수면의 이상 거동

Y.J. Kim and S.B. Lee

INDEX 목차 目次

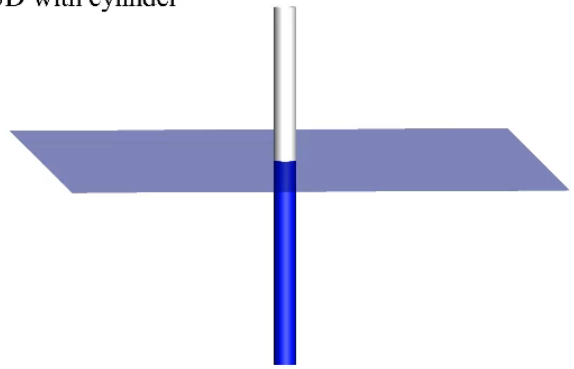
Problem statement



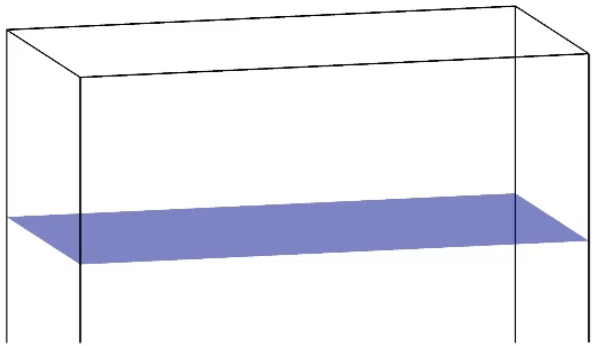
- 경계 조건에 따른 자유 수면의 이상 거동 발생
- 이에 대한 원인 분석과 개선 필요

Simplification

(a) 3D with cylinder



(b) 3D without cylinder



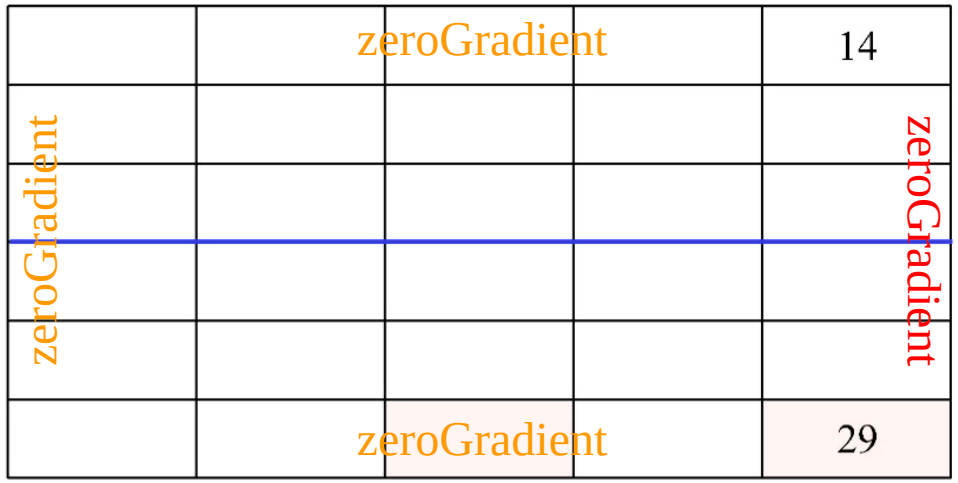
(c) 2D without cylinder

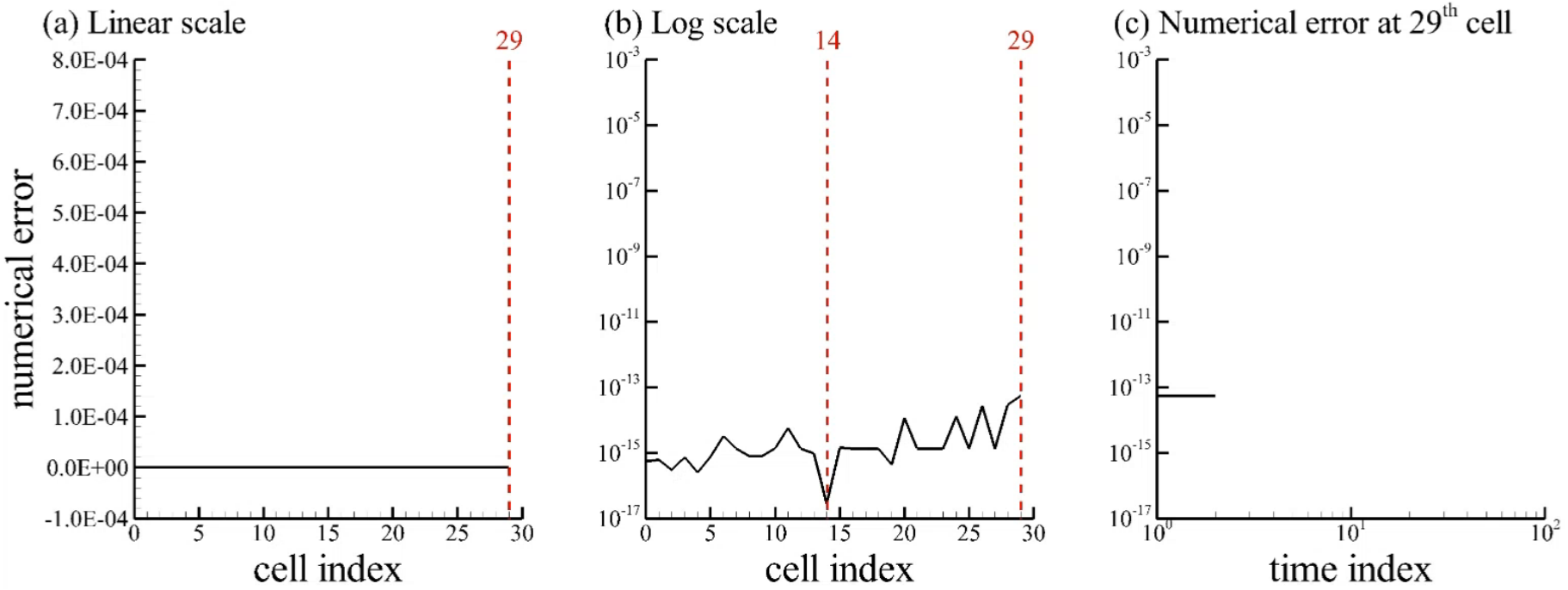
➢ 3D with cylinder case의 자유 수면 이상 거동

➢ 자유 수면의 이상 거동이 cylinder와 무관

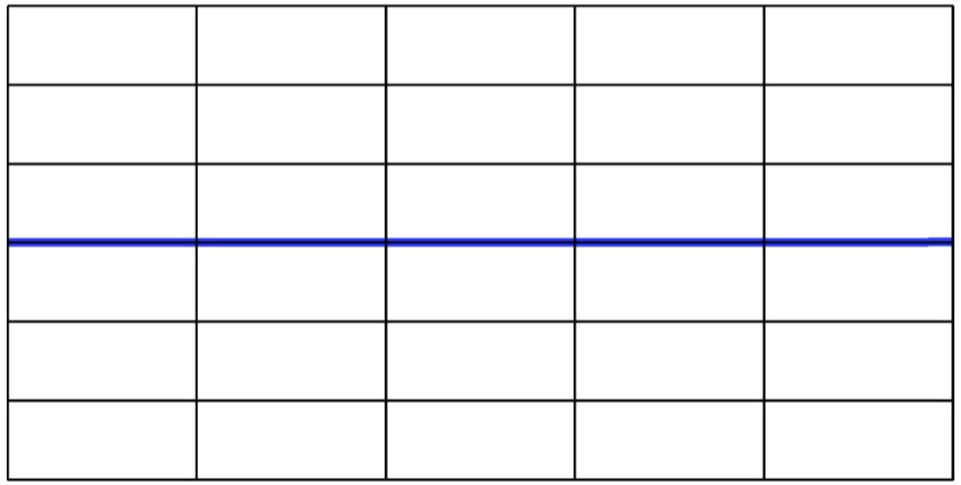
➢ 2D without cylinder case에서 3D 결과와 동일 오류

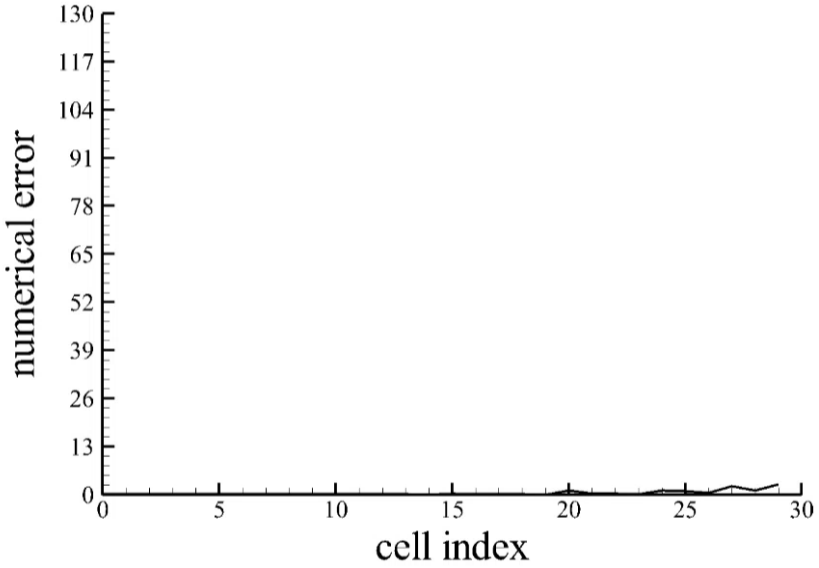
Inherent effect of significant digit





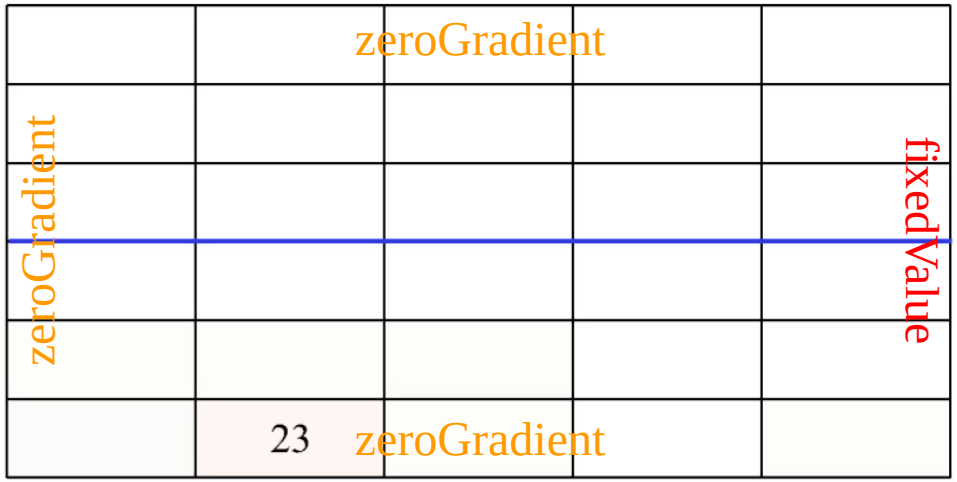
Inherent effect of significant digit

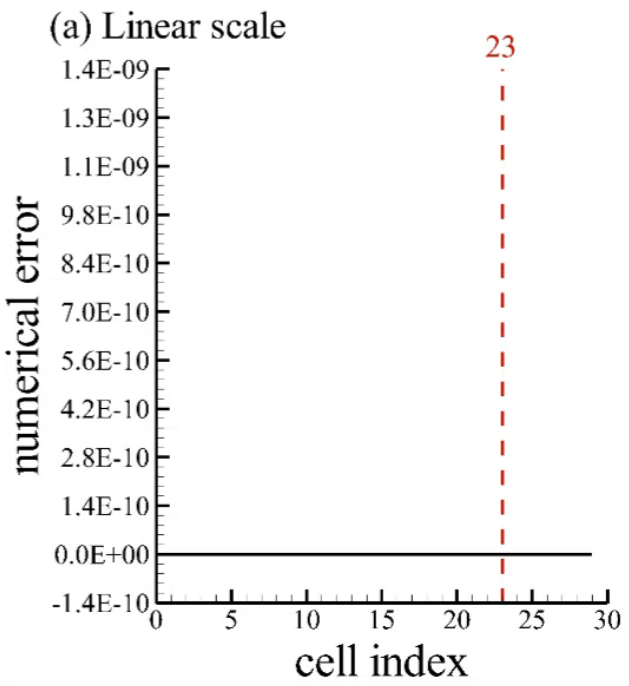


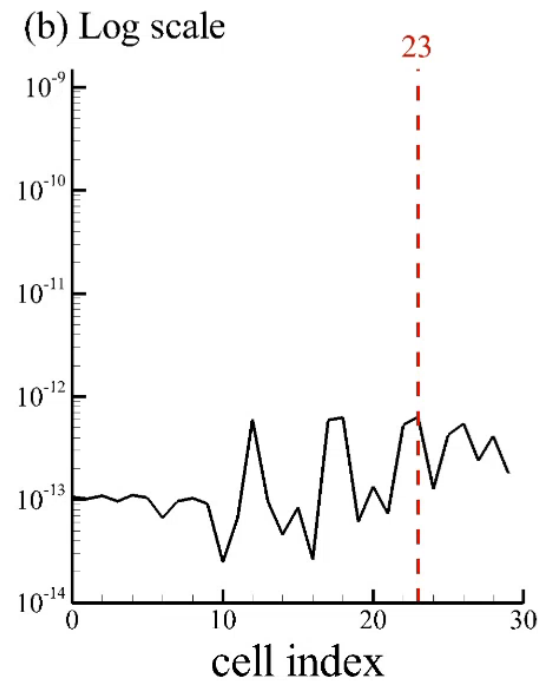


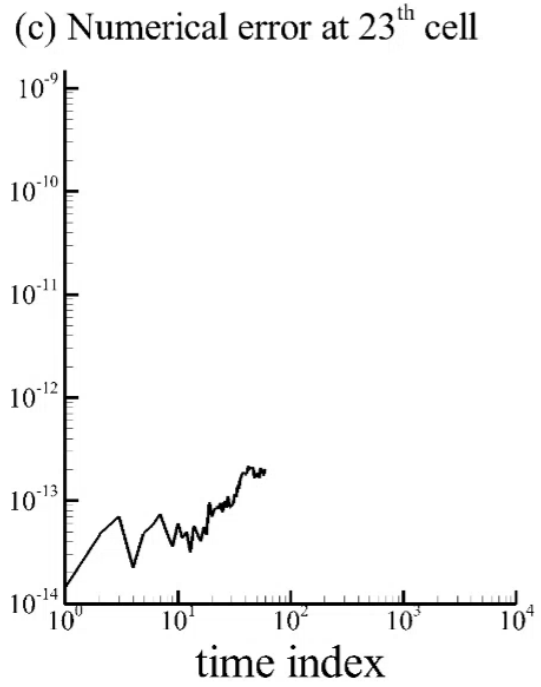
- double precision(10^{-15}) 이하의 수치 오차 누적
- 모든 경계에 zeroGradient 적용시 비 물리적 거동 발생
- 격자가 조밀한 경우, 자유 수면 이상 거동 지연

Inherent effect of significant digit

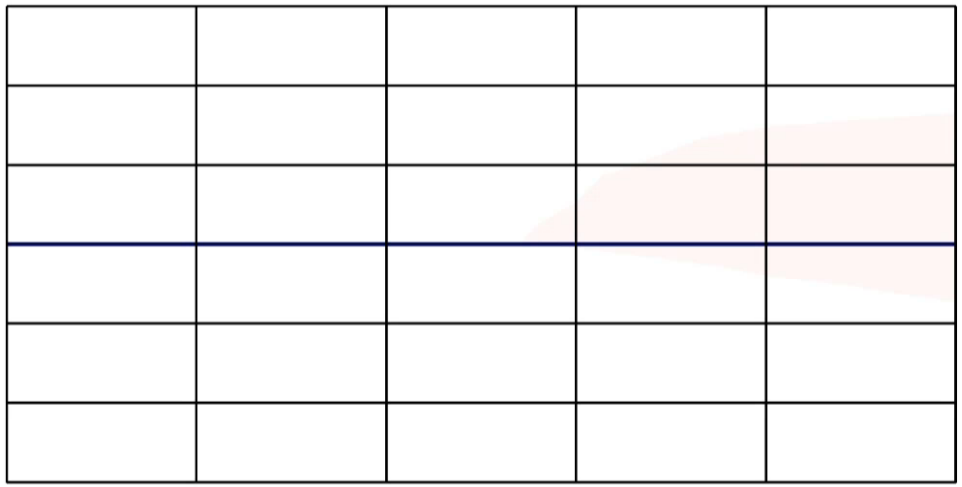


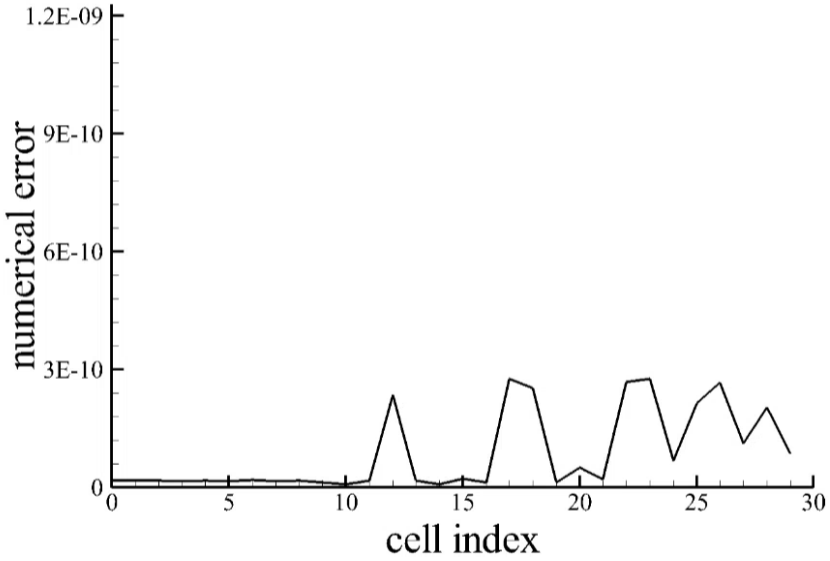






Inherent effect of significant digit

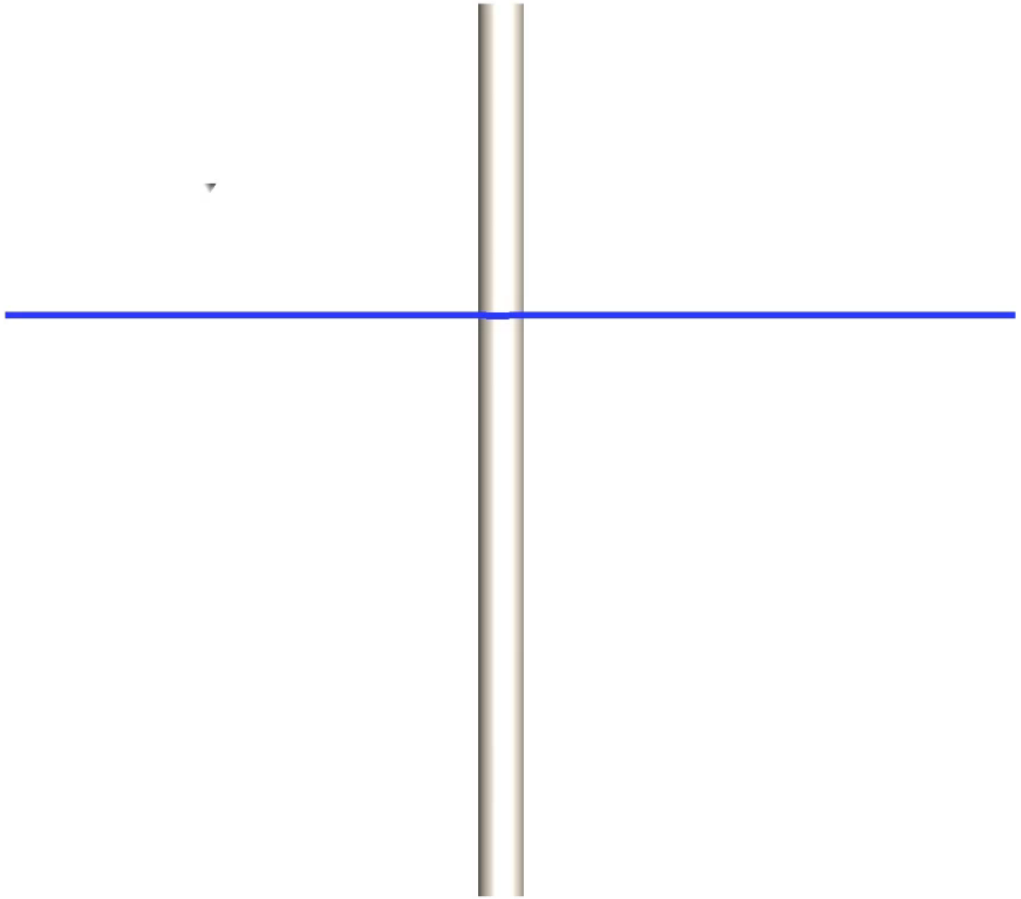




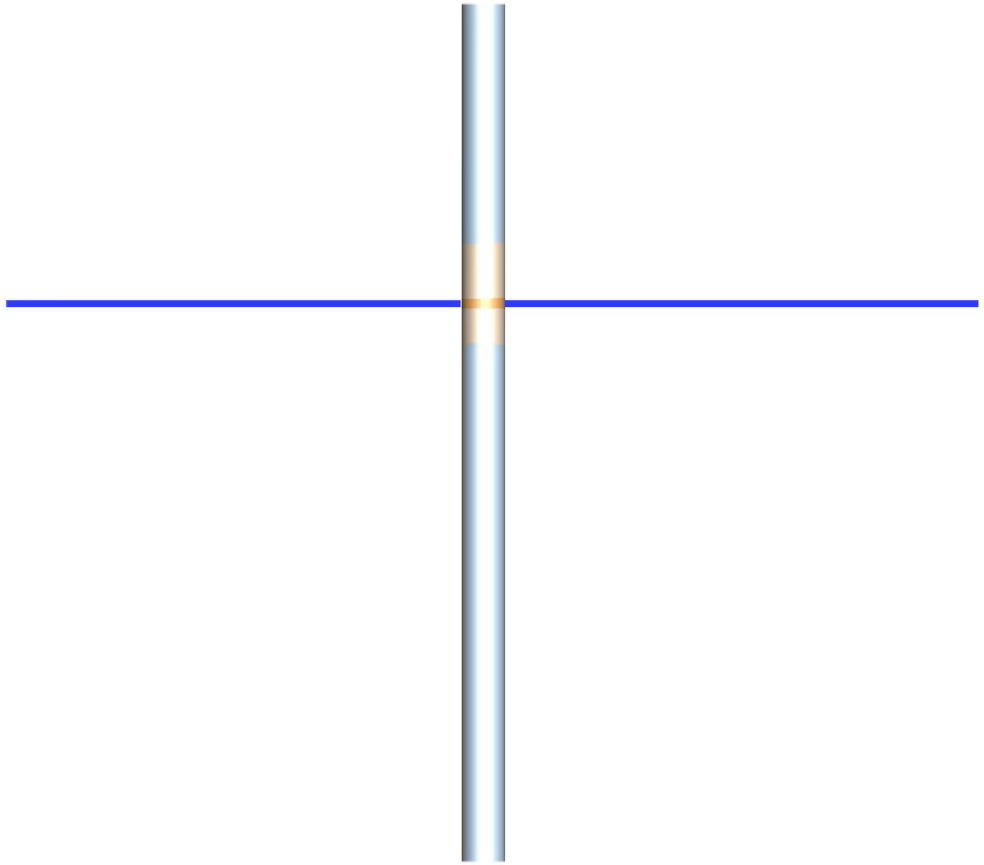
- 수치 오차가 10^{-9} 에 수렴
- Dirichlet 경계 조건이 수치 오차 제한
- 압력에 대한 fixedValue 경계 조건의 유효성 고찰 필요

Inherent effect of significant digit

(a) zeroGradient at all boundaries

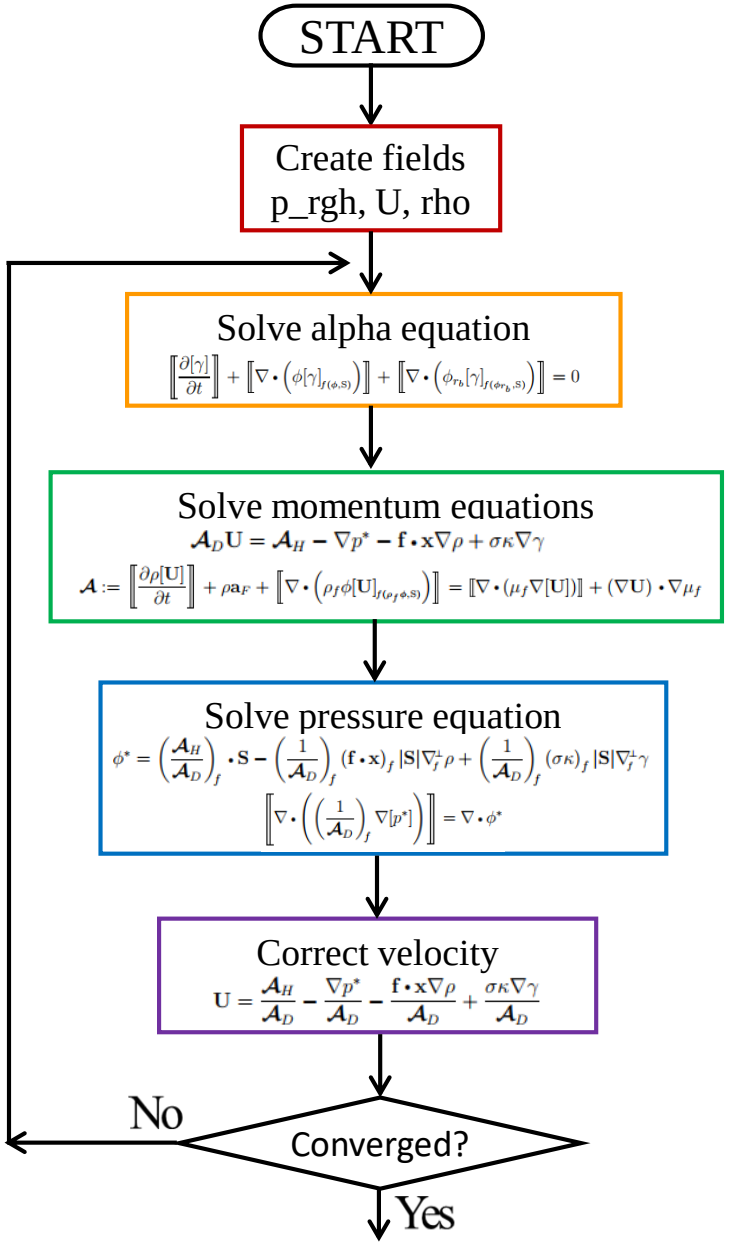


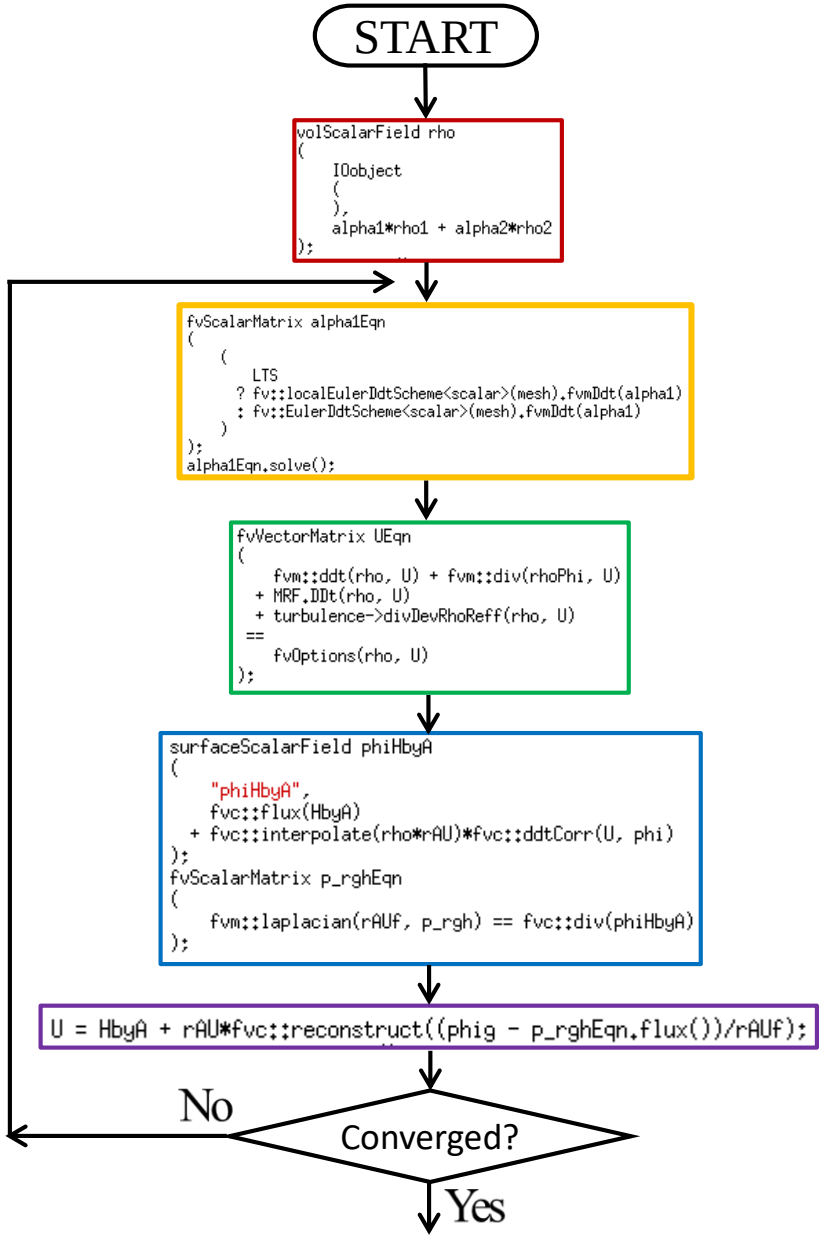
(b) fixedValue at outlet boundary



➢ 2D simulation의 수치 오차가 3D with cylinder simulation에서도 발생함.

Code analysis





Code analysis

createFields.H

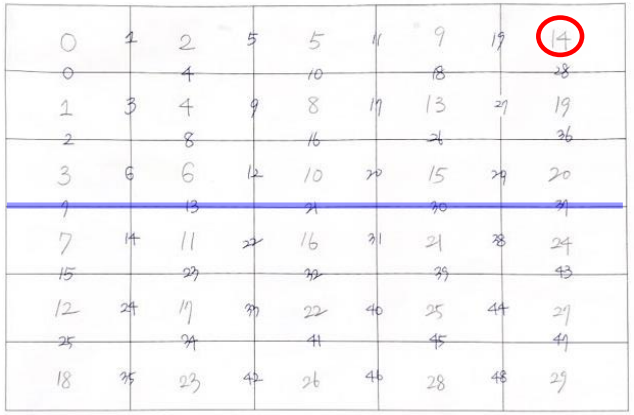
```
volScalarField p_rgh
(
    IOobject
    (
        "p_rgh",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

```
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    p_rgh + rho*gh
);

label pRefCell = 0;
scalar pRefValue = 0.0;

setRefCell
(
    p,
    p_rgh,
    pimple.dict(),
    pRefCell,
    pRefValue
);

if (p_rgh.needReference())
{
    p += dimensionedScalar
    (
        "p",
        p.dimensions(),
        pRefValue - getRefCellValue(p, pRefCell)
    );
    p_rgh = p - rho*gh;
}
```



1) p_rgh = 0.0

2) p = p_rgh+rgh
->value of each rho & cell center

3) p -= pRefValue(C_{14})
(pRefValue=1x-9.81x0.025=-0.24525)

4) p_rgh = p – rgh

⇒ p_rgh=p_rgh+rgh-pRefValue-rgh

⇒ p_rgh -= pRefValue

```
volScalarField rho
(
    IOobject
    (
        "rho",
        runTime.timeName(),
        mesh,
        IOobject::READ_IF_PRESENT
    ),
    alpha1*rho1 + alpha2*rho2
);
```

- rho1 : water(999.05)
- rho2 : air(1)
-alpha1 : water ratio
-alpha2 : 1-alpha1

celli	init_p_rgh[celli]
0	0.24524999999999968025576890795
1	0.24524999999999968025576890795
2	0.24524999999999968025576890795
3	0.24524999999999968025576890795
4	0.24524999999999968025576890795
5	0.24524999999999968025576890795
6	0.24524999999999968025576890795
7	0.24524999999999863575794734060 8
8	0.24524999999999968025576890795
9	0.24524999999999968025576890795
10	0.24524999999999968025576890795
11	0.24524999999999863575794734060 8
12	0.24524999999999863575794734060 8
13	0.24524999999999968025576890795
14	0.24524999999999968025576890795
15	0.24524999999999968025576890795
16	0.24524999999999863575794734060 8
17	0.24524999999999863575794734060 8
18	0.24524999999999863575794734060 8
19	0.24524999999999968025576890795
20	0.24524999999999968025576890795
21	0.24524999999999863575794734060 8
22	0.24524999999999863575794734060 8
23	0.24524999999999863575794734060 8
24	0.24524999999999863575794734060 8
25	0.24524999999999863575794734060 8
26	0.24524999999999863575794734060 8
27	0.24524999999999863575794734060 8
28	0.24524999999999863575794734060 8
29	0.24524999999999863575794734060 8

celli	init_rho[celli]
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	999.04999999999840838427189738
8	1
9	1
10	1
11	999.04999999999954525264911354
12	999.04999999999840838427189738
13	1
14	1
15	1
16	999.04999999999954525264911354
17	999.04999999999954525264911354
18	999.04999999999954525264911354
19	1
20	1
21	999.04999999999954525264911354
22	999.04999999999954525264911354
23	999.04999999999840838427189738
24	999.04999999999954525264911354
25	999.04999999999954525264911354
26	999.04999999999840838427189738
27	999.04999999999954525264911354
28	999.04999999999954525264911354
29	999.04999999999954525264911354

OpenCFD

Code analysis

목차 INDEX

START

```
volScalarField rho
(
    IOobject
    (
    ),
    alpha1*rho1 + alpha2*rho2
);
```

```
fvScalarMatrix alpha1Eqn
(
    (
        LTS
      ? fv::localEulerDdtScheme<scalar>(mesh).fvmDdt(alpha1)
      : fv::EulerDdtScheme<scalar>(mesh).fvmDdt(alpha1)
    )
);
alpha1Eqn.solve();
```

```
fvVectorMatrix UEqn
(
    fvm::ddt(rho, U) + fvm::div(rhoPhi, U)
  + MRF.DDt(rho, U)
  + turbulence->divDevRhoReff(rho, U)
 ==
    fvOptions(rho, U)
);
```

```
surfaceScalarField phiHbyA
(
    "phiHbyA",
    fvc::flux(HbyA)
  + fvc::interpolate(rho*rAU)*fvc::ddtCorr(U, phi)
);
fvScalarMatrix p_rghEqn
(
    fvm::laplacian(rAUf, p_rgh) == fvc::div(phiHbyA)
);
```

```
U = HbyA + rAU*fvc::reconstruct((phig - p_rghEqn.flux())/rAUf);
```

Converged?

No

Yes

UEqn.H

```
MRF.correctBoundaryVelocity(U);

fvVectorMatrix UEqn
(
    fvm::ddt(rho, U) + fvm::div(rhoPhi, U)
  + MRF.DDt(rho, U)
  + turbulence->divDevRhoReff(rho, U)
 ==
    fvOptions(rho, U)
);

UEqn.relax();

fvOptions.constrain(UEqn);

if (pimple.momentumPredictor())
{
    solve
    (
        UEqn
     ==
        fvc::reconstruct_DAU
        (
            (
                mixture.surfaceTensionForce()
              - ghf*fvc::snGrad(rho)
              - fvc::snGrad(p_rgh)
            ) * mesh.magSf()
        )
    );

    fvOptions.correct(U);
}
```

UEqn.coef()

```
1.305402e-05 -3.552e-08 9.93833333333e-07 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-3.552e-08 1.308954e-05 0 -3.552e-08 9.93833333333e-07 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-1.00616666667e-06 0 1.20478533333e-05 0 -3.552e-08 9.93833333333e-07 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -3.552e-08 0 1.44384804e-05 0 0 9.93833333333e-07 -1.3844604e-06 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -1.00616666667e-06 -3.552e-08 0 1.20833733333e-05 0 -3.552e-08 0 9.93833333333e-07 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -1.00616666667e-06 0 0 1.20478533333e-05 0 0 -3.552e-08 9.93833333333e-07 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 -1.00616666667e-06 -3.552e-08 0 1.34323137333e-05 0 0 0 9.93833333333e-07 -1.3844604e-06 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 -1.3844604e-06 0 0 0 0.0129931915074 0 0 0 0.00099857545125 -2.7334008e-06 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 -1.00616666667e-06 -3.552e-08 0 0 1.20833733333e-05 0 -3.552e-08 0 0 9.93833333333e-07 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 -1.00616666667e-06 0 0 0 1.20478533333e-05 0 0 0 -3.552e-08 9.93833333333e-07 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 -1.00616666667e-06 0 -3.552e-08 0 1.34323137333e-05 0 0 0 0 9.93833333333e-07 -1.3844604e-06 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 -1.3844604e-06 -0.00099952454875 0 0 0 0.0119936669587 0 0 0 0 0.00099857545125 -2.7334008e-06 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 -2.7334008e-06 0 0 0 0 0.0129945404478 0 0 0 0 0.00099857545125 -2.7334008e-06 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 -1.00616666667e-06 -3.552e-08 0 0 0 1.20833733333e-05 0 -3.552e-08 0 0 0 9.93833333333e-07 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 -1.00616666667e-06 0 0 0 0 1.30416866667e-05 0 0 0 0 -3.552e-08 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 -1.00616666667e-06 0 0 -3.552e-08 0 1.34323137333e-05 0 0 0 0 9.93833333333e-07 -1.3844604e-06 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 -1.3844604e-06 -0.00099952454875 0 0 0 0 0.0119936669587 0 0 0 0 0.00099857545125 -2.7334008e-06 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 -2.7334008e-06 -0.00099952454875 0 0 0 0 0.0119950158991 0 0 0 0 0.00099857545125 -2.7334008e-06 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 -2.7334008e-06 0 0 0 0 0 0.012991807047 0 0 0 0 0.00099857545125 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 -1.00616666667e-06 -3.552e-08 0 0 0 0 1.30772066667e-05 -3.552e-08 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1.00616666667e-06 0 0 0 -3.552e-08 1.44261470667e-05 0 0 0 -1.3844604e-06 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1.3844604e-06 -0.00099952454875 0 0 0 0 0.0119936669587 0 0 0.00099857545125 -2.7334008e-06 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2.7334008e-06 -0.00099952454875 0 0 0 0 0.0119950158991 0 0 0.00099857545125 -2.7334008e-06 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2.7334008e-06 -0.00099952454875 0 0 0 0 0.0119922824983 0 0 0.00099857545125 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1.3844604e-06 -0.00099952454875 0 0 0.0129922424099 0 0 -2.7334008e-06 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2.7334008e-06 -0.00099952454875 0 0 0.0119950158991 0 0.00099857545125 -2.7334008e-06 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2.7334008e-06 -0.00099952454875 0 0 0.0119922824983 0 0.00099857545125 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2.7334008e-06 -0.00099952454875 0 0.0129935913503 0 -2.7334008e-06
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2.7334008e-06 -0.00099952454875 0 0.0119922824983 0.00099857545125
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2.7334008e-06 -0.00099952454875 0.0129908579495
```

UEqn.source()

```
0 (2.80246666666666644293507207331e-06 0 0)
1 (2.80246666666666644293507207331e-06 0 0)
2 (2.39999999999999989139546838213e-06 0 0)
3 (2.80246666666666644293507207331e-06 0 0)
4 (2.39999999999999989139546838213e-06 0 0)
5 (2.39999999999999986208460475068e-06 0 0)
6 (2.40000000000000031491194200928e-06 0 0)
7 (0.00279752981949999996597044926716 0 0)
8 (2.399999999999999904436252112783e-06 0 0)
9 (2.40000000000000031491194200928e-06 0 0)
10 (2.399999999999999904436252112783e-06 0 0)
11 (0.00239772000000000003198008524663 0 0)
12 (0.00279752981950000039965131826136 0 0)
13 (2.400000000000000158546136289073e-06 0 0)
14 (2.399999999999999777381310024638e-06 0 0)
15 (2.40000000000000031491194200928e-06 0 0)
16 (0.00239771999999999994524391144779 0 0)
17 (0.00239772000000000003198008524663 0 0)
18 (0.00279752981949999996597044926716 0 0)
19 (2.399999999999999904436252112783e-06 0 0)
20 (2.39999999999999986208460475068e-06 0 0)
21 (0.00239772000000000003198008524663 0 0)
22 (0.002397719999999999858507737648949 0 0)
23 (0.00239771999999999994524391144779 0 0)
24 (0.00239771999999999994524391144779 0 0)
25 (0.00239772000000000003198008524663 0 0)
26 (0.002397719999999999858507737648949 0 0)
27 (0.002397719999999999815139650749529 0 0)
28 (0.00239772000000000003198008524663 0 0)
29 (0.002397719999999999858507737648949 0 0)
```

Code analysis

UEqn.H

```
MRF.correctBoundaryVelocity(U);

fvVectorMatrix UEqn
(
    fvm::ddt(rho, U) + fvm::div(rhoPhi, U)
  + MRF.DDt(rho, U)
  + turbulence->divDevRhoReff(rho, U)
 ==
    fvOptions(rho, U)
);

UEqn.relax();

fvOptions.constrain(UEqn);

if (pimple.momentumPredictor())
{
    solve
    (
        UEqn
     ==
        fvc::reconstruct_DAU
        (
            (
                mixture.surfaceTensionForce()
              - ghf*fvc::snGrad(rho)
              - fvc::snGrad(p_rgh)
            ) * mesh.magSf()
        )
    );

    fvOptions.correct(U);
}
```

UEqn == reconstruct

```
template<class Type>
tmp<GeometricField<Type, fvsPatchField, surfaceMesh>>
snGradScheme<Type>::snGrad
(
    const GeometricField<Type, fvPatchField, volMesh>& vf
) const
{
    tmp<GeometricField<Type, fvsPatchField, surfaceMesh>> tsf
    (
        snGrad(vf, deltaCoeffs(vf))
    );

    if (corrected())
    {
        tsf.ref() += correction(vf);
    }

    return tsf;
}
```

```
template<class Type>
tmp<GeometricField<Type, fvsPatchField, surfaceMesh>>
snGradScheme<Type>::snGrad
(
    const GeometricField<Type, fvPatchField, volMesh>& vf,
    const tmp<surfaceScalarField>& tdeltaCoeffs,
    const word& snGradName
)
                                        ⋮
    forAll(owner, facei)
    {
        ssf[facei] =
            deltaCoeffs[facei]*(vf[neighbour[facei]] - vf[owner[facei]]);
    }
```

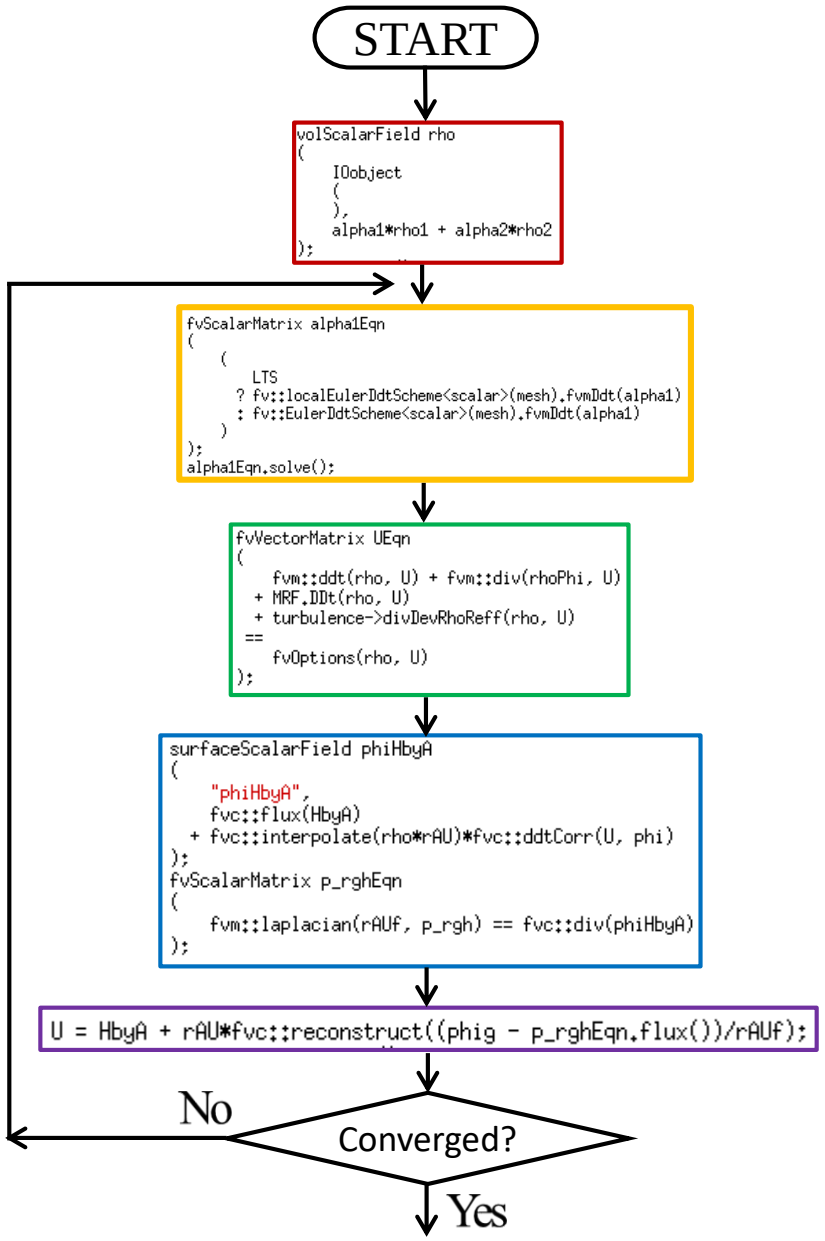
```
template<>
Foam::tmp<Foam::surfaceScalarField>
Foam::fv::correctedSnGrad<Foam::scalar>::correction
(
    const volScalarField& vsf
) const
{
    return fullGradCorrection(vsf);
}
```

UEqn.source ( )

```
0 (2.8024666666666664429350720733 1e-06 0 0)
1 (2.8024666666666664429350720733 1e-06 0 0)
2 (2.3999999999999989139546838213e-06 0 0)
3 (2.8024666666666664429350720733 1e-06 0 0)
4 (2.3999999999999989139546838213e-06 0 0)
5 (2.3999999999999986208460475006 8e-06 0 0)
6 (2.4000000000000003149119420092 8e-06 0 0)
7 (0.0027975298194999996597044926716 0 0)
8 (2.3999999999999990443625211278 3e-06 0 0)
9 (2.4000000000000003149119420092 8e-06 0 0)
10 (2.3999999999999990443625211278 3e-06 0 0)
11 (0.0023977200000000003198008524663 0 0)
12 (0.0027975298195000003996513182613 6 0 0)
13 (2.4000000000000015854613628907 3e-06 0 0)
14 (2.3999999999999977738131002463 8e-06 0 0)
15 (2.4000000000000003149119420092 8e-06 0 0)
16 (0.0023977199999999994524391144779 0 0)
17 (0.0023977200000000003198008524663 0 0)
18 (0.0027975298194999996597044926716 0 0)
19 (2.3999999999999990443625211278 3e-06 0 0)
20 (2.3999999999999986208460475006 8e-06 0 0)
21 (0.0023977200000000003198008524663 0 0)
22 (0.0023977199999999985850773764894 9 0 0)
23 (0.0023977199999999994524391144779 0 0)
24 (0.0023977199999999994524391144779 0 0)
25 (0.0023977200000000003198008524663 0 0)
26 (0.0023977199999999985850773764894 9 0 0)
27 (0.0023977199999999981513965074952 9 0 0)
28 (0.0023977200000000003198008524663 0 0)
29 (0.0023977199999999985850773764894 9 0 0)
```

```
0 (2.8024666666666664429350720733 1e-06 0 0)
1 (2.8024666666666664429350720733 1e-06 0 0)
2 (2.3999999999999989139546838213e-06 0 0)
3 (2.8024666666666655855023231394 55e-06 -1.5987211554602254587123363325 1e-20 0)
4 (2.3999999999999989139546838213e-06 0 0)
5 (2.3999999999999986208460475006 8e-06 0 0)
6 (2.4000000000000003149119420092 8e-06 -1.5987211554602263614919977640 3e-20 0)
7 (0.0027975298194999996597044926716 -1.5987211554602254587123363325 1e-20 0)
8 (2.3999999999999990443625211278 3e-06 0 0)
9 (2.4000000000000003149119420092 8e-06 0 0)
10 (2.4000000000000202201862024853 4e-06 -1.5987211554602251577857825220 1e-20 0)
11 (0.0023977200000000003198008524663 -1.5987211554602263614919977640 3e-20 0)
12 (0.0027975298195000003996513182613 6 2.6766429073177266748364885502 4e-19 0)
13 (2.4000000000000015854613628907 3e-06 0 0)
14 (2.3999999999999977738131002463 8e-06 0 0)
15 (2.4000000000000109028237826880 3e-06 -1.5987211554602251577857825220 1e-20 0)
16 (0.0023977199999999994524391144779 -1.5987211554602251577857825220 1e-20 0)
17 (0.0023977200000000003198008524663 -2.6766429073177261933540024534 3e-19 0)
18 (0.0027975298194999996597044926716 2.6766429073177261933540024534 3e-19 0)
19 (2.3999999999999990443625211278 3e-06 0 0)
20 (2.3999999999999986208460475006 8e-06 -1.5987211554602248568922871150 2e-20 0)
21 (0.0023977200000000003198008524663 -1.5987211554602251577857825220 1e-20 0)
22 (0.0023977199999999985850773764894 9 -2.6766429073177271563189746470 5e-19 0)
23 (0.0023977199999999994524391144779 -2.6766429073177252303890302598 2e-19 0)
24 (0.0023977199999999994524391144779 -1.5987211554602251577857825220 1e-20 0)
25 (0.0023977200000000003198008524663 1.1808357971524236146325223673 6e-34 0)
26 (0.0023977199999999985850773764894 9 -2.6766429073177271563189746470 5e-19 0)
27 (0.0023977199999999981513965074952 9 0 0)
28 (0.0023977200000000003198008524663 1.1808357971524238284536991747 4e-34 0)
29 (0.0023977199999999985850773764894 9 0 0)
```

Code analysis



pEqn.H

```
volScalarField rAU("rAU", 1.0/UEqn.A());
surfaceScalarField rAUf("rAUf", fvc::interpolate(rAU));

volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p_rgh));

surfaceScalarField phiHbyA
(
    "phiHbyA",
    fvc::flux(HbyA)
  + fvc::interpolate(rho*rAU)*fvc::ddtCorr(U, phi)
);
MRF.makeRelative(phiHbyA);
adjustPhi(phiHbyA, U, p_rgh);

surfaceScalarField phig
(
    (
        mixture.surfaceTensionForce()
      - ghf*fvc::snGrad(rho)
    )*rAUf*mesh.magSf()
);

phiHbyA += phig;

// Update the pressure BCs to ensure flux consistency
constrainPressure(p_rgh, U, phiHbyA, rAUf, MRF);

while (pimple.correctNonOrthogonal())
{
    fvScalarMatrix p_rghEqn
    (
        fvm::laplacian(rAUf, p_rgh) == fvc::div(phiHbyA)
    );

    p_rghEqn.setReference(pRefCell, getRefCellValue(p_rgh, pRefCell));

    p_rghEqn.solve(mesh.solver(p_rgh.select(pimple.finalInnerIter())));

    if (pimple.finalNonOrthogonalIter())
    {
        phi = phiHbyA - p_rghEqn.flux();

        p_rgh.relax();

        U = HbyA + rAU*fvc::reconstruct((phig - p_rghEqn.flux())/rAUf);
        U.correctBoundaryConditions();
        fvOptions.correct(U);
    }
}

#include "continuityErrs.H"

p == p_rgh + rho*gh;

if (p_rgh.needReference())
{
    p += dimensionedScalar
    (
        "p",
        p.dimensions(),
        pRefValue - getRefCellValue(p, pRefCell)
    );
    p_rgh = p - rho*gh;
}
```

$$rAU = \frac{1}{UEqn.A()} = \frac{mesh.V()}{UCoeff.Diag()}$$

$$HbyA = rAU \times \frac{(source - coeff \times phi')}{mesh.V()} = \frac{(source - coeff \times phi')}{UCoeff.Diag()}$$

$$phig = rAUf \times UEqn.snGrad(rho)$$

p_rghEqn.flux()

```
for (label face=0; face<l.size(); face++)
{
    faceHpsi[face] =
        Upper[face]*psi[u[face]]
      - Lower[face]*psi[l[face]];
}
```

Visualization

UEqn.H

```
MRF.correctBoundaryVelocity(U);

fvVectorMatrix UEqn
(
    fvm::ddt(rho, U) + fvm::div(rhoPhi, U)
  + MRF.DDt(rho, U)
  + turbulence->divDevRhoReff(rho, U)
 ==
    fvOptions(rho, U)
);

UEqn.relax();

fvOptions.constrain(UEqn);

if (pimple.momentumPredictor())
{
    solve
    (
        UEqn
     ==
        fvc::reconstruct_DAU
        (
            (
                mixture.surfaceTensionForce()
              - ghf*fvc::snGrad(rho)
              - fvc::snGrad(p_rgh)
            ) * mesh.magSf()
        )
    );

    fvOptions.correct(U);
}
```

$$U_{Error} = U - U_{init}$$
$$p\_rgh_{Error} = p\_rgh - p\_rgh_{init}$$

pEqn.H

```
volScalarField rAU("rAU", 1.0/UEqn.A());
surfaceScalarField rAUf("rAUf", fvc::interpolate(rAU));

volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p_rgh));

surfaceScalarField phiHbyA
(
    "phiHbyA",
    fvc::flux(HbyA)
  + fvc::interpolate(rho*rAU)*fvc::ddtCorr(U, phi)
);
MRF.makeRelative(phiHbyA);
adjustPhi(phiHbyA, U, p_rgh);

surfaceScalarField phig
(
    (
        mixture.surfaceTensionForce()
      - ghf*fvc::snGrad(rho)
    )*rAUf*mesh.magSf()
);

phiHbyA += phig;

// Update the pressure BCs to ensure flux consistency
constrainPressure(p_rgh, U, phiHbyA, rAUf, MRF);

while (pimple.correctNonOrthogonal())
{
    fvScalarMatrix p_rghEqn
    (
        fvm::laplacian(rAUf, p_rgh) == fvc::div(phiHbyA)
    );

    p_rghEqn.setReference(pRefCell, getRefCellValue(p_rgh, pRefCell));

    p_rghEqn.solve(mesh.solver(p_rgh.select(pimple.finalInnerIter())));

    if (pimple.finalNonOrthogonalIter())
    {
        phi = phiHbyA - p_rghEqn.flux();

        p_rgh.relax();

        U = HbyA + rAU*fvc::reconstruct((phig - p_rghEqn.flux())/rAUf);
        U.correctBoundaryConditions();
        fvOptions.correct(U);
    }
}

#include "continuityErrs.H"

p == p_rgh + rho*gh;

if (p_rgh.needReference())
{
    p += dimensionedScalar
    (
        "p",
        p.dimensions(),
        pRefValue - getRefCellValue(p, pRefCell)
    );
    p_rgh = p - rho*gh;
}
```