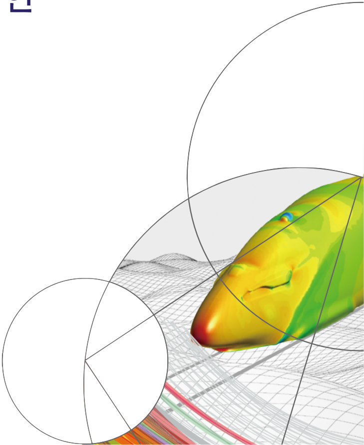
OpenFOAM 솔버의 문제점 및 해결 방안

길재흥

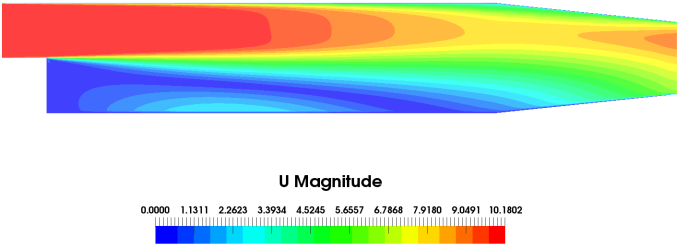
넥스트폼 기술연구소

2015년 9월 10일

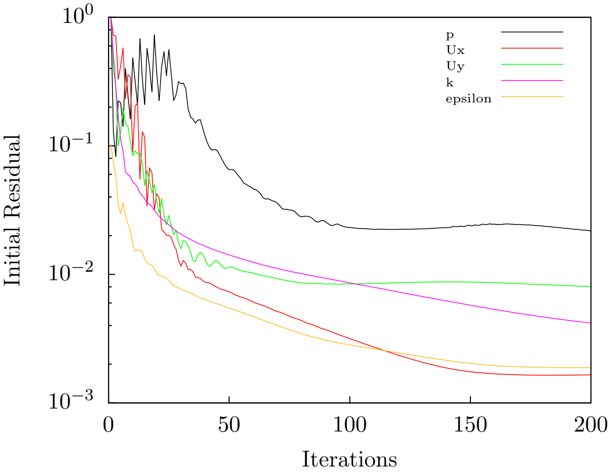


SIMPLE Algorithm

간단한 예제


U Magnitude
0.0000 1.1311 2.2623 3.3934 4.5245 5.6557 6.7868 7.9180 9.0491 10.1802

수렴성


10^0
10^{-1}
10^{-2}
10^{-3}
Initial Residual
0
50
100
150
200
Iterations
p
Ux
Uy
k
epsilon

OpenFOAM의 SIMPLE 알고리즘

1. Solve momentum equation

$$a_P\vec{U}_P + \sum_N a_N\vec{U}_N = -V_P(\nabla p)_P \tag{1}$$

$$a_P\vec{U}_P = H(\vec{U}) - V_P(\nabla p)_P$$

   - get $\vec{U}^*$ from solving equation (1)

2. Express velocity as:

$$\vec{U}_P^* = \frac{H(\vec{U}^*)}{a_P} - \frac{V_P}{a_P}(\nabla p)_P \tag{2}$$

3. Calculate pseudo mass flow rate at cell face

$$F^* = \left\{\frac{H(\vec{U}^*)}{a}\right\}_f \cdot \vec{S}_f \tag{3}$$

   - Collocated grid에서 pressure checker-board 현상을 피하기 위함
   - Rhie-Chow Interpolation or Delayed Pressure Discretization

OpenFOAM의 SIMPLE 알고리즘

4 Solve pressure equation

$$\nabla \cdot \left(\frac{V}{a} \nabla p \right) = \sum_f F^* \tag{4}$$

- get corrected pressure p^* from solving equation (4)

5 Correct mass flow rate

$$F^{new} = F^* - \left(\frac{V}{a} \right)_f |\vec{S}_f| \vec{n} \cdot (\nabla p^*)_f \tag{5}$$

6 Under-relax pressure

$$p^{new} = p^{old} + \alpha_p (p^* - p^{old}) \tag{6}$$

OpenFOAM의 SIMPLE 알고리즘

7. Correct cell velocity

$$\vec{U}_P^{new} = \frac{H(\vec{U}^*)}{a_P} - \frac{V_P}{a_P}(\nabla p^{new})_P \quad (7)$$

OpenFOAM의 SIMPLE 알고리즘

7. Correct cell velocity

$$\vec{U}_P^{new} = \frac{H(\vec{U}^*)}{a_P} - \frac{V_P}{a_P}(\nabla p^{new})_P \tag{7}$$

- p^{new}는 under-relaxation이 적용된 상태

OpenFOAM의 SIMPLE 알고리즘

7. Correct cell velocity

$$\vec{U}_P^{new} = \frac{H(\vec{U}^*)}{a_P} - \frac{V_P}{a_P}(\nabla p^{new})_P \tag{7}$$

- p^{new}는 under-relaxation이 적용된 상태
- mass flow rate correction과 velocity correction이 일관되지 않음

OpenFOAM의 SIMPLE 알고리즘

7 Correct cell velocity

$$\vec{U}_P^{new} = \frac{H(\vec{U}^*)}{a_P} - \frac{V_P}{a_P}(\nabla p^{new})_P \tag{7}$$

- p^{new}는 under-relaxation이 적용된 상태
- mass flow rate correction과 velocity correction이 일관되지 않음

"A continuity-satisfying velocity field is likely to be more reasonable than the starred velocities. ... Furthermore, the solution of the other scalar equations in every iteration can be based on a flow field that satisfies a mass balance. To realize these advantages, one precaution is necessary: The velocity corrections should not be underrelaxed."

— S. V. Patankar(1980), Numerical Heat Transfer and Fluid Flow, pp.128

OpenFOAM의 SIMPLE 알고리즘

7 Correct cell velocity

$$\vec{U}_P^{new} = \frac{H(\vec{U}^*)}{a_P} - \frac{V_P}{a_P}(\nabla p^{new})_P \qquad (7)$$

- p^{new}는 under-relaxation이 적용된 상태
- mass flow rate correction과 velocity correction이 일관되지 않음

> *"A continuity-satisfying velocity field is likely to be more reasonable than the starred velocities. ... Furthermore, the solution of the other scalar equations in every iteration can be based on a flow field that satisfies a mass balance. To realize these advantages, one precaution is necessary: The velocity corrections should not be underrelaxed."*

— S. V. Patankar(1980), Numerical Heat Transfer and Fluid Flow, pp.128

Under-relaxation 순서 수정

5. Correct mass flow rate

$$F^{new} = F^{*} - \left(\frac{V}{a}\right)_f |\vec{S}_f| \vec{n} \cdot (\nabla p^{*})_f$$

6. Correct cell velocity

$$\vec{U}_P^{new} = \frac{H(\vec{U}^{*})}{a_P} - \frac{V_P}{a_P} (\nabla p^{*})_P$$

7. Under-relax pressure

$$p^{new} = p^{old} + \alpha_p (p^{*} - p^{old})$$

Under-relaxation 순서 수정

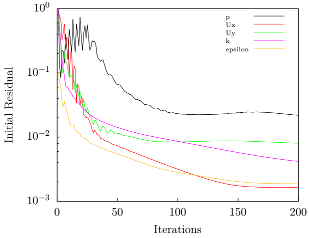
- simpleFoam 코드 수정

simpleFoam/pEqn.H

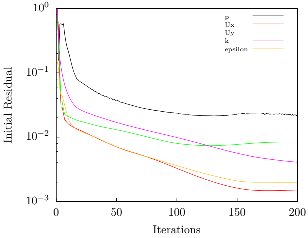
```
    .
    .
    .
        pEqn.solve();                              //solve pressure equation

        if (simple.finalNonOrthogonalIter())
        {
            phi = phiHbyA - pEqn.flux();           //correct mass flow rate
        }
    }

    #include "continuityErrs.H"

    //p.relax();

    // Momentum corrector
    U = HbyA - rAU*fvc::grad(p);                   //correct cell velocity
    U.correctBoundaryConditions();

    // Explicitly relax pressure for momentum corrector
    p.relax();                                     //under-relax pressure
    .
    .
    .
```

Under-relaxation 순서 수정

- 수정 효과 확인 #1- pitzDaily - residuals



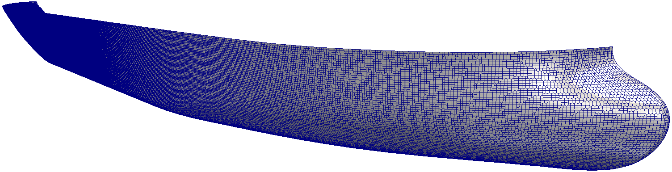
(a) original

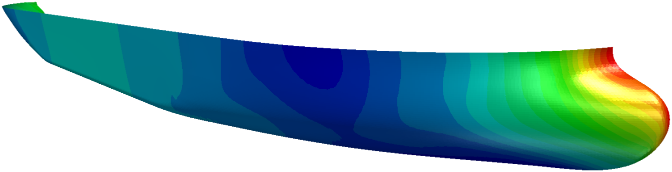


(b) modified

Under-relaxation 순서 수정

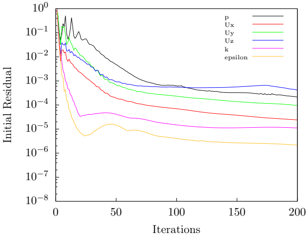
- 수정 효과 확인 #2- double-body ship



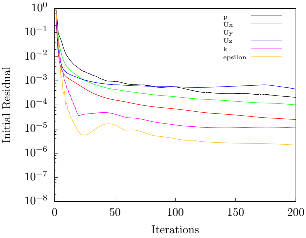


Under-relaxation 순서 수정

- 수정 효과 확인 #2 - double-body ship : residuals



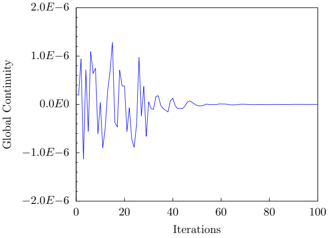
(a) original



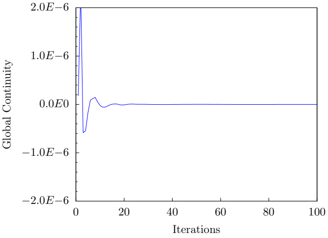
(b) modified

Under-relaxation 순서 수정

- 수정 효과 확인 #2 - double-body ship : continuity



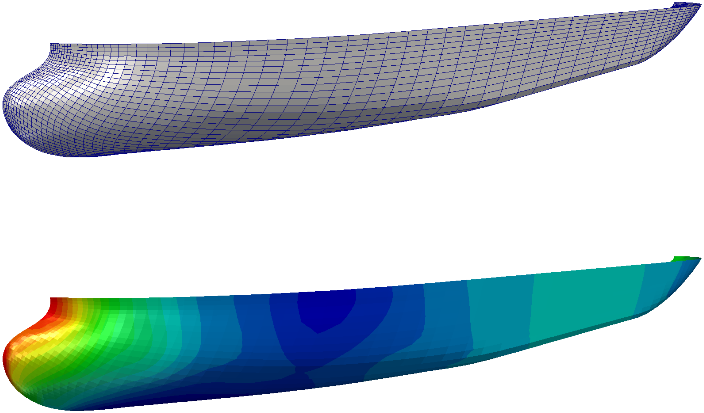
(a) original



(b) modified

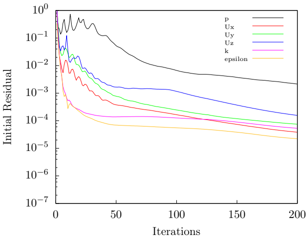
Under-relaxation 순서 수정

- 수정 효과 확인 #3- double-body ship(structured grid)

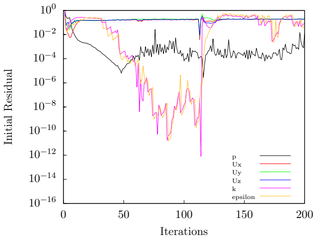


Under-relaxation 순서 수정

- 수정 효과 확인 #3 - double-body ship(structured grid) : residuals



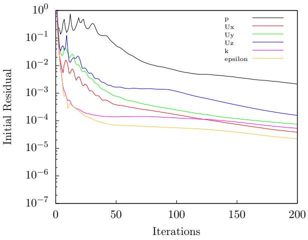
(a) original



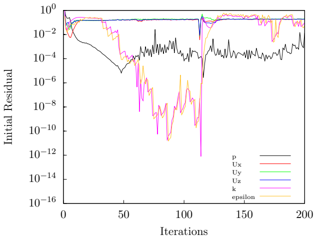
(b) modified

Under-relaxation 순서 수정

- 수정 효과 확인 #3 - double-body ship(structured grid) : residuals



(a) original



(b) modified

아; 왜 ?

Non-orthogonality

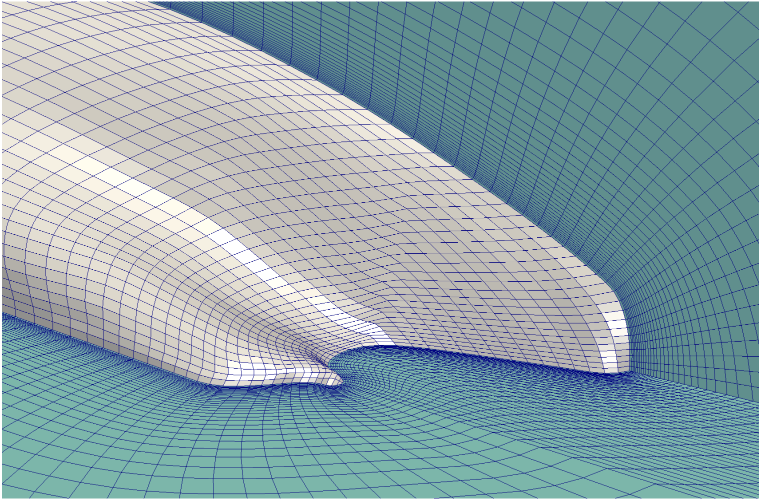
checkMesh

```
    Mesh (non-empty) directions (1 1 1)
    Boundary openness (9.61139e-17 1.28784e-15 5.83167e-16) OK.
    Max cell openness = 7.26193e-16 OK.
    Max aspect ratio = 137.654 OK.
    Minimum face area = 1.20105e-07. Maximum face area = 0.0062768.  Face area
         magnitudes OK.
    Min volume = 1.26073e-10. Max volume = 0.000212885.  Total volume = 2.49792.
         Cell volumes OK.
    Mesh non-orthogonality Max: 76.8778 average: 27.948
   *Number of severely non-orthogonal (> 70 degrees) faces: 549.
    Non-orthogonality check OK.
  <<Writing 549 non-orthogonal faces to set nonOrthoFaces
    Face pyramids OK.
 ***Max skewness = 26.0322, 123 highly skew faces detected which may impair the
       quality of the results
  <<Writing 123 skew faces to set skewFaces
    Coupled point location match (average 0) OK.

Failed 1 mesh checks.
```

NEXTfoam open source CFD consulting

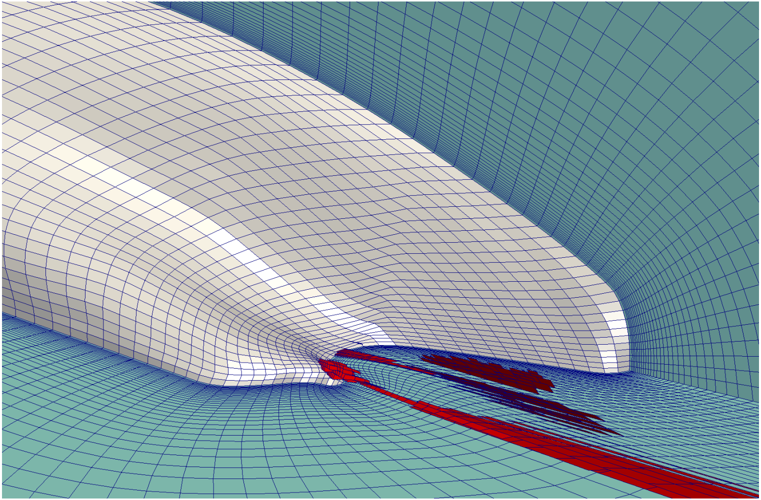
Non-orthogonality

- 선미부에 심하게 찌그러진 격자 다수 분포



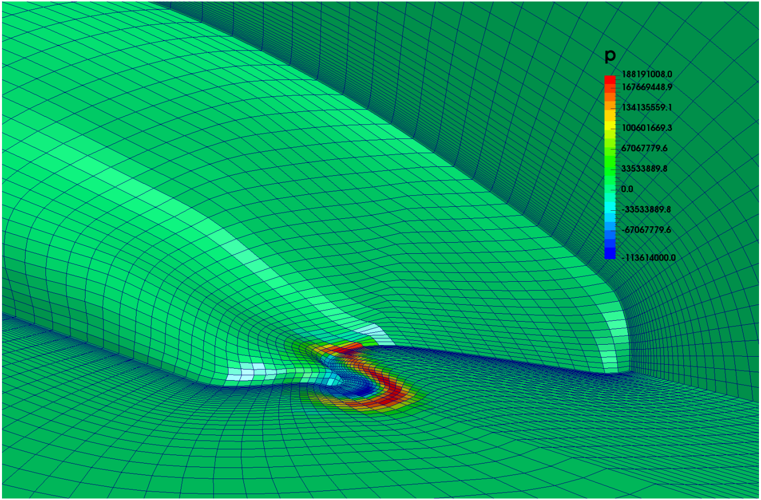
Non-orthogonality

- 선미부에 심하게 찌그러진 격자 다수 분포



Non-orthogonality

- 압력 분포



Under-relaxation 순서에 관한 결론

- 속도 수정(velocity correction)에 under-relaxation이 적용된 압력을 사용하는 것은 non-orthogonal mesh에서의 수렴에 도움.

$$\vec{U}_P^{new} = \frac{H(\vec{U}^*)}{a_P} - \frac{V_P}{a_P}(\nabla p^{new})_P$$

Under-relaxation 순서에 관한 결론

- 속도 수정(velocity correction)에 under-relaxation이 적용된 압력을 사용하는 것은 non-orthogonal mesh에서의 수렴에 도움.

$$\vec{U}_P^{new} = \frac{H(\vec{U}^*)}{a_P} - \frac{V_P}{a_P}(\nabla p^{new})_P$$

- 이대로 괜찮은가?

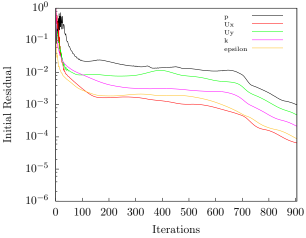
교과서대로 해보자

- Mass flow rate를 구하기 위한 velocity interpolation을 Rhie-Chow Interpolation으로 변경

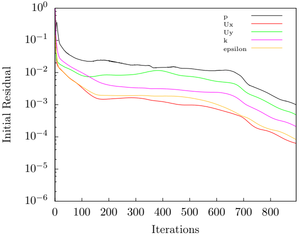
	original	modified
1. solve momentum equation and get $\vec{U}^*$	$a_P \vec{U}_P = H(\vec{U}) - V_P (\nabla p)_P$	$a_P \vec{U}_P = H(\vec{U}) - V_P (\nabla p)_P$
2. interpolate pseudo-velocity to get mass flow rate	$F^* = \left\{ \frac{H(\vec{U}^*)}{a} \right\}_f \cdot \vec{S}_f$	$F^* = \left\{ \vec{U}^* + \frac{V_P}{a_P} (\nabla p)_P \right\}_f \cdot \vec{S}_f$
3. solve pressure equation and get p^*	$\nabla \cdot \left(\frac{V}{a} \nabla p \right) = \sum_f F^*$	$\nabla \cdot \left(\frac{V}{a} \nabla p \right) = \sum_f F^*$
4. correct mass flow rate	$F^{new} = F^* - \left(\frac{V}{a} \right)_f \lvert\vec{S}_f\rvert \vec{n} \cdot (\nabla p^*)_f$	$F^{new} = F^* - \left(\frac{V}{a} \right)_f \lvert\vec{S}_f\rvert \vec{n} \cdot (\nabla p^*)_f$
5. original: under-relax pressure modified: correct velocity	$p^{new} = p^{old} + \alpha_p (p^* - p^{old})$	$\vec{U}_P^{new} = \vec{U}_P^* - \frac{V_P}{a_P} (\nabla p')_P$
6. original: correct velocity modified: under-relax pressure	$\vec{U}_P^{new} = \frac{H(\vec{U}^*)}{a_P} - \frac{V_P}{a_P} (\nabla p^{new})_P$	$p^{new} = p^{old} + \alpha_p (p^* - p^{old})$

교과서대로 해보자

- pitzDaily 수렴성



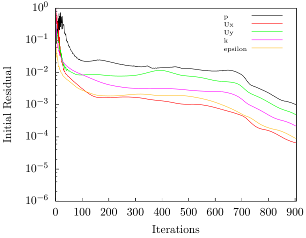
(a) original



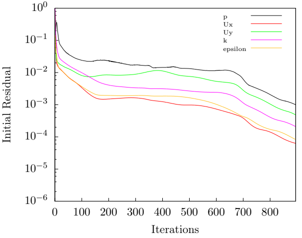
(b) modified

교과서대로 해보자

- pitzDaily 수렴성



(a) original



(b) modified

- double-body ship(structured grid) 케이스는 여전히 문제...

문제의 원인

- 확산항의 이산화

$$\frac{\partial(\rho\psi)}{\partial t} + \nabla\cdot(\rho\vec{U}\psi) - \nabla\cdot(\rho\Gamma_\psi\nabla\psi) = S_\psi(\psi)$$

$$\int_{V_P}\nabla\cdot(\rho\Gamma_\psi\nabla\psi)dV \Rightarrow \sum_f(\rho\Gamma_\psi)_f|\vec{S}_f|\vec{n}\cdot(\nabla\psi)_f$$

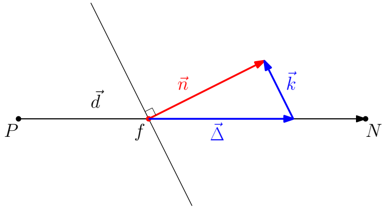
- 이산화된 확산항에서 surface normal gradient($\vec{n}\cdot(\nabla\psi)_f$)를 구하는 방법
  - Central differencing with non-orthogonal correction

system/fvSchemes

```
laplacianSchemes
{
    default                 Gauss linear corrected;
}

.
.
.
```

문제의 원인

- 격자의 orthogonality가 좋지 않을 경우 발산하게 되는 원인
  - non-orthogonal correction



- Split unit normal vector($\vec{n}$) into two parts:

$$\vec{n} = \vec{\Delta} + \vec{k}$$

then,

$$\vec{n} \cdot (\nabla\psi)_f = |\vec{\Delta}| \frac{\psi_N - \psi_P}{|\vec{d}|} + \underbrace{\vec{k} \cdot (\nabla\psi)_f}_{\text{non-orthogonal contribution}}$$

해결 방안 #1

해결 방안 #1

격자를 다시...

해결 방안 #1

격자를 다시... 잘...

해결 방안 #2

- uncorrected scheme

system/fvSchemes

```
.
.
.

laplacianSchemes
{
    default                    Gauss linear uncorrected;
}

snGradSchemes
{
    default                    uncorrected;
}

.
.
.
```

$$\vec{n} \cdot (\nabla \psi)_f = |\vec{\Delta}| \frac{\psi_N - \psi_P}{|\vec{d}|} + \cancel{\vec{k} \cdot (\nabla \psi)_f}$$

해결 방안 #2

- uncorrected scheme

system/fvSchemes

```
.
.
.

laplacianSchemes
{
    default                     Gauss linear uncorrected;
}

snGradSchemes
{
    default                     uncorrected;
}

.
.
.
```

$$\vec{n} \cdot (\nabla \psi)_f = |\vec{\Delta}| \frac{\psi_N - \psi_P}{|\vec{d}|} + \cancel{\vec{k} \cdot (\nabla \psi)_f}$$

- 수렴은 가능하지만 부정확한 해

해결 방안 #3

- limited scheme

system/fvSchemes

```
.
.
.

laplacianSchemes
{
    default             Gauss linear limited α;
}

snGradSchemes
{
    default             limited α;
}

.
.
.
```

$$\vec{n} \cdot (\nabla \psi)_f = |\vec{\Delta}| \frac{\psi_N - \psi_P}{|\vec{d}|} + \alpha \left\{ \vec{k} \cdot (\nabla \psi)_f \right\}$$

$$(0 < \alpha < 1)$$

해결 방안 #3

- limited scheme

system/fvSchemes

```
.
.
.

laplacianSchemes
{
    default             Gauss linear limited α;
}

snGradSchemes
{
    default             limited α;
}

.
.
.
```

$$\vec{n}\cdot(\nabla\psi)_f = |\vec{\Delta}|\frac{\psi_N-\psi_P}{|\vec{d}|} + \alpha\left\{\vec{k}\cdot(\nabla\psi)_f\right\}$$

$$(0<\alpha<1)$$

- α 값에 따라 다른 특성

해결 방안 #4

- under-relax non-orthogonal contribution

system/fvSchemes

```
.
.
.

laplacianSchemes
{
    default                     Gauss linear corrected;
}

snGradSchemes
{
    default                     corrected;
}

.
.
.
```

$$\vec{n} \cdot (\nabla\psi)_f = |\vec{\Delta}| \frac{\psi_N - \psi_P}{|\vec{d}|} + \left\{ \vec{k} \cdot (\nabla\psi)_f \right\}_{relax}$$

해결 방안 #4

- under-relax non-orthogonal contribution

simpleFoam/pEqn.H

```
.
.
.

// Non-orthogonal pressure corrector loop
while (simple.correctNonOrthogonal())
{
    fvScalarMatrix pEqn
    (
        fvm::laplacian(rAUf, p) == fvc::div(phi)
    );

    // under-relax non-orthogonal contribution in pressure laplacian
    #include "relaxPressureFaceFluxCorrection.H"

    pEqn.setReference(pRefCell, pRefValue);

    pEqn.solve();

    if (simple.finalNonOrthogonalIter())
    {
        phi = -= pEqn.flux();
    }
}

.
.
.
```

해결 방안 #4

- under-relax non-orthogonal contribution

simpleFoam/createFields.H

```
.
.
.

surfaceScalarField pFaceFluxCorrection
(
    IOobject
    (
        "pFaceFluxCorrection",
        runTime.timeName(),
        mesh,
        IOobject::READ_IF_PRESENT,
        IOobject::AUTO_WRITE
    ),
    mesh,
    dimensionedScalar("zero", dimVolume/dimTime, 0.0),
    calculatedFvPatchField<scalar>::typeName
);

.
.
.
```

해결 방안 #4

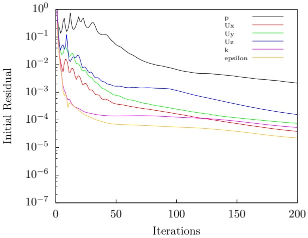
- under-relax non-orthogonal contribution

simpleFoam/relaxPressureFaceFluxCorrection.H

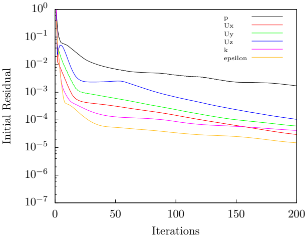
```
scalar pUrf = mesh.fieldRelaxationFactor("p");

surfaceScalarField oldFaceFluxCorrection = pFaceFluxCorrection;

pFaceFluxCorrection = *(pEqn.faceFluxCorrectionPtr());

surfaceScalarField relaxedFaceFluxCorrection
(
    oldFaceFluxCorrection
      + pUrf*(pFaceFluxCorrection - oldFaceFluxCorrection)
);

pEqn.source() += mesh.V()
    *fvc::div
    (
        pFaceFluxCorrection - relaxedFaceFluxCorrection
    )().internalField();

*(pEqn.faceFluxCorrectionPtr()) = relaxedFaceFluxCorrection;
```

해결 방안 #4

- 수정 결과



(a) original



(b) modified

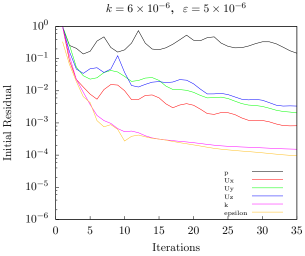
- Drag Coefficient

	original	modified	commercial
Cd	0.00417	0.00419	0.00420

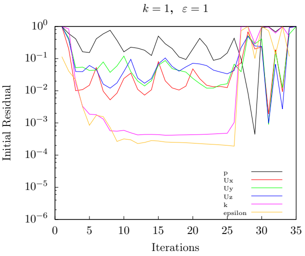
Turbulence Model

난류초기조건과 난류점성계수

- Double-body ship(structured grid)
  - original **`simpleFoam`**
  - standard $k-\varepsilon$ model



(a) good initial condition



(b) bad initial condition

난류초기조건과 난류점성계수

- 난류점성계수
  - Model equation

$$\nu_t = C_\mu \frac{k^2}{\varepsilon}$$

  - 난류초기조건이 수렴성에 영향을 주는것은 운동량보존방정식에 사용되는 난류점성계수 때문

- 수렴 안정화를 위한 장치
  - 난류값 제한

kEpsilon.C

```
solve(epsEqn);
bound(epsilon_, epsilonMin_);
.
.
.
solve(kEqn);
bound(k_, kMin_);
```

난류초기조건과 난류점성계수

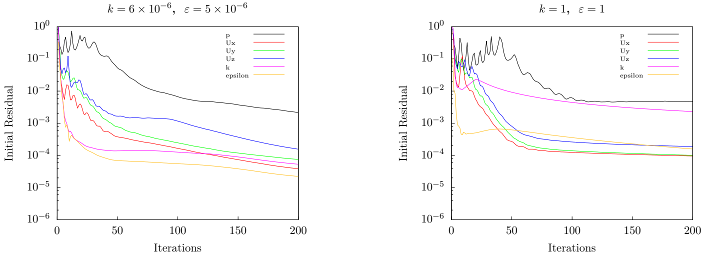
- 수렴 안정화를 위한 장치 개선
  - 난류값 제한 수정 및 난류점성계수 제한

standardKEpsilon.C

```
.
.
.

solve(epsEqn);
bound(epsilon_, epsilonMin_*1e-5);

.
.
.

solve(kEqn);
bound(k_, kMin_*10);

.
.
.

// Re-calculate viscosity
nut_ = min(Cmu_*sqr(k_)/epsilon_, 1e5*nu());

.
.
.
```

난류초기조건과 난류점성계수

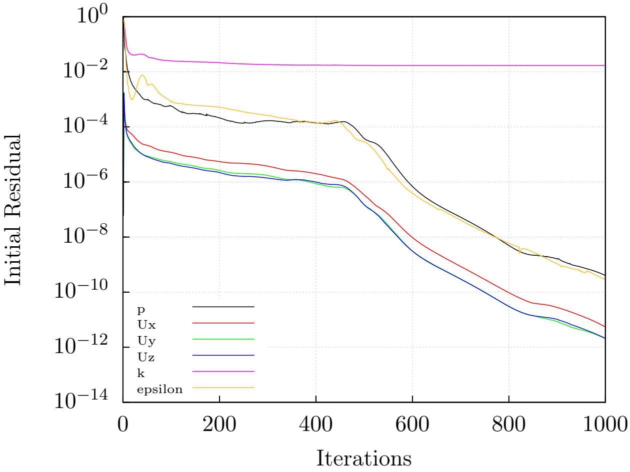
- 개선 효과



(a) good initial condition (b) bad initial condition

난류방정식의 생성항 선형화(Linearization)

- realizable $k - \varepsilon$ 모델의 기이한 residual...



난류방정식의 생성항 선형화(Linearization)

- Realizable $k-\varepsilon$ Model
  - k-방정식

$$\frac{\partial(\rho k)}{\partial t} + \nabla\cdot(\rho\vec{U}k) - \nabla\cdot\left\{\left(\mu + \frac{\mu_t}{\sigma_k}\right)\nabla k\right\} = G_k - \rho\varepsilon \qquad (8)$$

  - ε-방정식

$$\frac{\partial(\rho\varepsilon)}{\partial t} + \nabla\cdot(\rho\vec{U}\varepsilon) - \nabla\cdot\left\{\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\nabla\varepsilon\right\} = \rho C_1 S\varepsilon - \rho C_2\frac{\varepsilon^2}{k + \sqrt{\nu\varepsilon}} \qquad (9)$$

  - 계수들

$$\sigma_k = 1.0$$

$$\sigma_\varepsilon = 1.2$$

$$C_1 = max\left[0.43, \frac{\eta}{\eta+5}\right], \quad \eta = S\frac{k}{\varepsilon}, \quad S = \sqrt{2S_{ij}S_{ij}}$$

$$C_2 = 1.9$$

난류방정식의 생성항 선형화(Linearization)

- 생성항이 지배방정식의 종속변수 ψ에 대한 함수로 표현되는 경우에는 선형화 필요

$$S_\psi(\psi) = S_u - S_p\psi \tag{10}$$

  - S_p 는 반드시 양수가 되도록 해야한다.

- k-방정식의 생성항

$$S_k = G_k - \rho\varepsilon \tag{11}$$

- ε-방정식의 생성항

$$S_\varepsilon = \rho C_1 S\varepsilon - \rho C_2 \frac{\varepsilon^2}{k + \sqrt{\nu\varepsilon}} \tag{12}$$

난류방정식의 생성항 선형화(Linearization)

- OpenFOAM의 선형화
  - k-방정식

$$S_k = S_u - S_p k = G_k - \left(\frac{\rho\varepsilon^*}{k^*}\right) k$$

realizableKE.C

```
G - fvm::Sp(epsilon_/k_, k_)
```

  - ε-방정식

$$S_\varepsilon = S_u - S_p\varepsilon = \rho C_1 S\varepsilon^* - \left(\frac{\rho C_2\varepsilon^*}{k^* + \sqrt{\nu\varepsilon^*}}\right)\varepsilon$$

realizableKE.C

```
    C1*magS*epsilon_
  - fvm::Sp
    (
        C2_*epsilon_/(k_ + sqrt(nu()*epsilon_)),
        epsilon_
    )
```

- The symbol (*) is used to denote the value from previous iteration

난류방정식의 생성항 선형화(Linearization)

- 선형화 방법 변경
  - 일반화된 선형화 방법:

$$S_\psi(\psi) = S^* + \left(\frac{\partial S}{\partial \psi}\right)^* (\psi - \psi^*)$$
$$= \left\{S^* - \left(\frac{\partial S}{\partial \psi}\right)^* \psi^*\right\} + \left(\frac{\partial S}{\partial \psi}\right)^* \psi$$

  - 식 (10)의 형태로 표현하면,

$$S_u = \left\{S^* - \left(\frac{\partial S}{\partial \psi}\right)^* \psi^*\right\}$$
$$S_p = -\left(\frac{\partial S}{\partial \psi}\right)^* \tag{13}$$

난류방정식의 생성항 선형화(Linearization)

- k-방정식 생성항 선형화
  - 난류점성계수의 정의를 이용하여 생성항을 표현

$$\begin{aligned} S_k &= G_k - \rho\varepsilon \\ &= G_k - \rho^2 \frac{C_\mu}{\mu_t} k^2 \end{aligned} \tag{14}$$

그리고,

$$\frac{\partial S_k}{\partial k} = -2\rho^2 \frac{C_\mu}{\mu_t} k \tag{15}$$

  - 식 (13)을 이용하여 표현하면,

$$\begin{aligned} S_u &= G_k + \rho^2 \frac{C_\mu}{\mu_t} (k^*)^2 \\ S_p &= 2\rho^2 \frac{C_\mu}{\mu_t} k^* \end{aligned} \tag{16}$$

난류방정식의 생성항 선형화(Linearization)

- ε-방정식 생성항 선형화
  - η와 난류점성계수를 이용하여 생성항을 표현

$$
\begin{aligned}
S_\varepsilon &= \rho C_1 S\varepsilon - \rho C_2 \frac{\varepsilon^2}{k + \sqrt{\nu\varepsilon}} \\
&= \rho C_1 \eta \frac{\varepsilon^2}{k} - \rho C_2 \frac{\varepsilon^2}{k + \sqrt{\nu\varepsilon}} \\
&= \rho^{3/2} \left(\frac{C_1 \eta}{\sqrt{\mu_t / C_\mu}} - \frac{C_2}{\sqrt{\mu_t / C_\mu} + \sqrt{\mu}} \right) \varepsilon^{3/2}
\end{aligned} \tag{17}
$$

그리고,

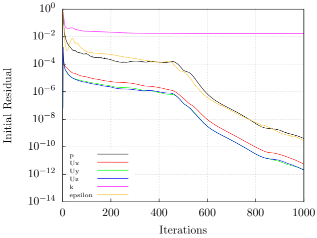
$$
\frac{\partial S_\varepsilon}{\partial \varepsilon} = \frac{3}{2} \rho^{3/2} \left(\frac{C_1 \eta}{\sqrt{\mu_t / C_\mu}} - \frac{C_2}{\sqrt{\mu_t / C_\mu} + \sqrt{\mu}} \right) \sqrt{\varepsilon} \tag{18}
$$

  - 식 (13)을 이용하여 표현하면,

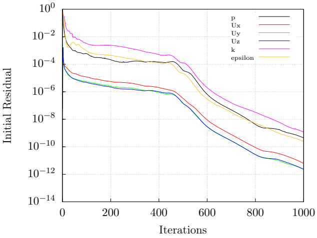
$$
\begin{aligned}
S_u &= \frac{1}{2} \rho^{3/2} \left(\frac{C_2}{\sqrt{\mu_t / C_\mu} + \sqrt{\mu}} - \frac{C_1 \eta}{\sqrt{\mu_t / C_\mu}} \right) (\varepsilon^*)^{3/2} \\
S_p &= \frac{3}{2} \rho^{3/2} \left(\frac{C_2}{\sqrt{\mu_t / C_\mu} + \sqrt{\mu}} - \frac{C_1 \eta}{\sqrt{\mu_t / C_\mu}} \right) \sqrt{\varepsilon^*}
\end{aligned} \tag{19}
$$

난류방정식의 생성항 선형화(Linearization)

- 변경 효과



(a) original



(b) modified

비압축성 난류모델의 divDevReff 함수

- **simpleFoam**의 운동량보존방정식 풀이

simpleFoam/UEqn.H

```
// Momentum predictor

tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U)
  + turbulence->divDevReff(U)
  ==
    fvOptions(U)
);

UEqn().relax();

fvOptions.constrain(UEqn());

solve(UEqn() == -fvc::grad(p));

fvOptions.correct(U);
```

비압축성 난류모델의 divDevReff 함수

- 운동량보존방정식
  - 압축성

$$\frac{\partial(\rho\vec{U})}{\partial t} + \nabla\cdot(\rho\vec{U}\vec{U}) - \nabla\cdot\left[\mu_{eff}\left\{\nabla\vec{U} + (\nabla\vec{U})^T\right\}\right] - \frac{2}{3}\mu_{eff}(\nabla\cdot\vec{U})\bar{\bar{I}}\Bigg\}\Bigg] = -\nabla p$$

비압축성 난류모델의 divDevReff 함수

- 운동량보존방정식
  - 압축성

$$\frac{\partial(\rho\vec{U})}{\partial t} + \nabla\cdot(\rho\vec{U}\vec{U}) - \nabla\cdot\left[\mu_{eff}\left\{\nabla\vec{U} + (\nabla\vec{U})^T\right\}\right] - \frac{2}{3}\mu_{eff}(\nabla\cdot\vec{U})\bar{\bar{I}}\Bigg\}\Bigg] = -\nabla p$$

  - 비압축성

$$\frac{\partial\vec{U}}{\partial t} + \nabla\cdot(\vec{U}\vec{U}) - \nabla\cdot\left[\nu_{eff}\left\{\nabla\vec{U} + (\nabla\vec{U})^T\right\}\right] - \frac{2}{3}\nu_{eff}(\nabla\cdot\vec{U})\bar{\bar{I}}\Bigg\}\Bigg] = -\nabla\hat{p}$$

비압축성 난류모델의 divDevReff 함수

- 운동량보존방정식
  - 압축성

$$\frac{\partial(\rho\vec{U})}{\partial t} + \nabla\cdot(\rho\vec{U}\vec{U}) - \nabla\cdot\left[\mu_{eff}\left\{\nabla\vec{U} + (\nabla\vec{U})^T\right\}\right] - \frac{2}{3}\mu_{eff}(\nabla\cdot\vec{U})\bar{\bar{I}}\Big\}\Big] = -\nabla p$$

  - 비압축성

$$\frac{\partial\vec{U}}{\partial t} + \nabla\cdot(\vec{U}\vec{U}) \underbrace{- \nabla\cdot\left[\nu_{eff}\left\{\nabla\vec{U} + (\nabla\vec{U})^T\right\}\right] - \frac{2}{3}\nu_{eff}(\nabla\cdot\vec{U})\bar{\bar{I}}\Big\}\Big]}_{\texttt{divDevReff(U)}} = -\nabla\hat{p}$$

비압축성 난류모델의 divDevReff 함수

- 운동량보존방정식
  - 압축성

$$\frac{\partial(\rho\vec{U})}{\partial t} + \nabla\cdot(\rho\vec{U}\vec{U}) - \nabla\cdot\left[\mu_{eff}\left\{\nabla\vec{U} + (\nabla\vec{U})^T\right\}\right] - \frac{2}{3}\mu_{eff}(\nabla\cdot\vec{U})\bar{\bar{I}}\Bigg\}\Bigg] = -\nabla p$$

  - 비압축성

$$\frac{\partial\vec{U}}{\partial t} + \nabla\cdot(\vec{U}\vec{U}) \underbrace{- \nabla\cdot\left[\nu_{eff}\left\{\nabla\vec{U} + (\nabla\vec{U})^T\right\}\right] - \frac{2}{3}\nu_{eff}(\nabla\cdot\vec{U})\bar{\bar{I}}\Bigg\}\Bigg]}_{\texttt{divDevReff(U)}} = -\nabla\hat{p}$$

kEpsilon.C

```
tmp<fvVectorMatrix> kEpsilon::divDevReff(volVectorField& U) const
{
    return
    (
      - fvm::laplacian(nuEff(), U)
      - fvc::div(nuEff()*dev(T(fvc::grad(U))))
    );
}
```

비압축성 난류모델의 `divDevReff` 함수

- 비압축성 운동량보존방정식

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot (\vec{U}\vec{U}) - \nabla \cdot \left[\nu_{eff} \left\{ \nabla \vec{U} + (\nabla \vec{U})^T \right\} \right] - \frac{2}{3} \nu_{eff} (\nabla \cdot \vec{U}) \bar{\bar{I}} \Big\} \Big] = -\nabla \hat{p}$$

비압축성 난류모델의 divDevReff 함수

- 비압축성 운동량보존방정식

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot (\vec{U}\vec{U}) - \nabla \cdot \left[\nu_{eff} \left\{ \nabla \vec{U} + (\nabla \vec{U})^T \right\} \right] - \cancel{\frac{2}{3}\nu_{eff}(\nabla \cdot \vec{U})\bar{\bar{I}}} \Big\} \Big] = -\nabla \hat{p}$$

비압축성 난류모델의 `divDevReff` 함수

- 비압축성 운동량보존방정식

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot (\vec{U}\vec{U}) - \nabla \cdot \left[\nu_{eff} \left\{ \nabla \vec{U} + (\nabla \vec{U})^T \right\} \right] = -\nabla \hat{p}$$

비압축성 난류모델의 divDevReff 함수

- 비압축성 운동량보존방정식

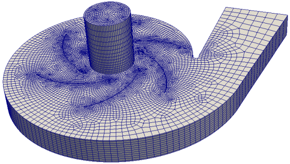
$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot (\vec{U}\vec{U}) - \nabla \cdot \left[\nu_{eff} \left\{ \nabla \vec{U} + (\nabla \vec{U})^T \right\} \right] = -\nabla \hat{p}$$

kEpsilon.C

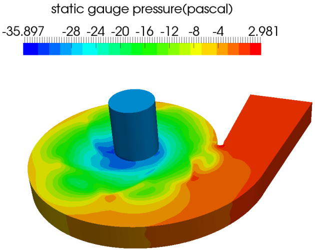
```
tmp<fvVectorMatrix> kEpsilon::divDevReff(volVectorField& U) const
{
    return
    (
      - fvm::laplacian(nuEff(), U)
      - fvc::div(nuEff()*dev(T(fvc::grad(U))))
      - fvc::div(nuEff()*T(fvc::grad(U)))
    );
}
```

비압축성 난류모델의 divDevReff 함수

- 변경 효과 검증 해석 대상
  - simple blower(1000rpm)



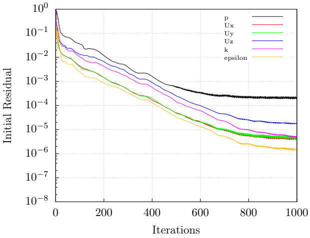
(a) grid



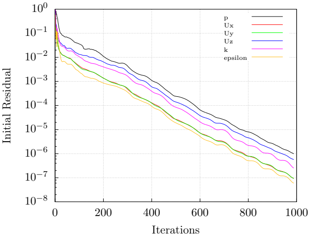
(b) solution

비압축성 난류모델의 divDevReff 함수

- 변경 효과



(a) original

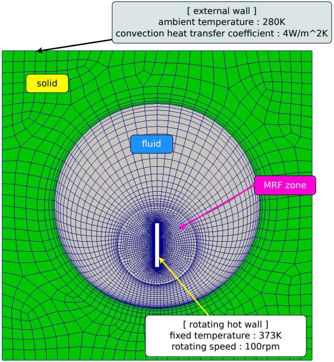


(b) modified

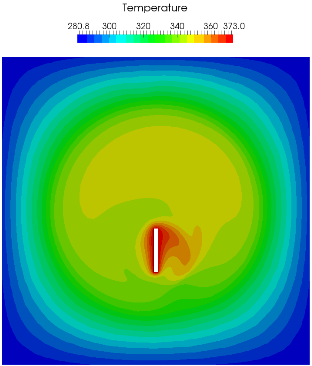
기타

복합열유동해석 솔버

- chtMultiRegionSimpleFoam



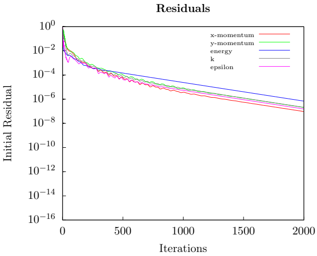
(a) grid & setup



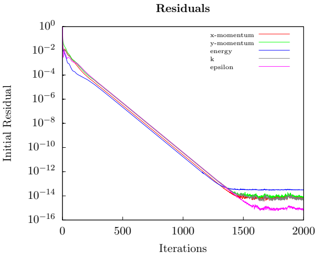
(b) solution

복합열유동해석 솔버

- chtMultiRegionSimpleFoam 성능 개선



(a) original



(b) modified

결론

- OpenFOAM은 잘 만들어진 Field Operation 라이브러리

결론

- OpenFOAM은 잘 만들어진 Field Operation 라이브러리
- 석연치 않은 솔버와 경계조건

결론

- OpenFOAM은 잘 만들어진 Field Operation 라이브러리
- 석연치 않은 솔버와 경계조건
- 솔버 및 경계조건들이 구현된 방식에 대한 세심한 점검 필요