愚公移山

Shoveling Chronicle
(Shoveling Oriented Presentation)

2019 OKUCC

COPYLEFT 2019 All wrongs reserved

Get Started

Insanity

Insanity is doing the same thing over and over again and expecting different results

Albert Einstein

Agenda

오늘은 그 동안 CFD를 하면서 열심히 삽질한 경험에 대해 공유해 보고자 합니다. 때로는 삽질이 좋은 결과로 이어지기도 했지만, 대부분의 삽질은 아무런 결과나 소득 없이 허무하게 끝나버리고 말았습니다.

본 발표를 통해 조금이라도 시행착오를 줄일 수 있었으면 하는 바람입니다.

- **삽질의 3요소**
- **삽질 1. 코드 개발**
- **삽질 2. 해석 안정화**
- **삽질 3. Two-phase flows**
- **결론 및 제언**


CFD
개노답
삼형제다!
개노답
삼형제!
속도
정확도
안정성


계산이
자꾸
터진다
고요?
왼손으로
엔터 쳐서
그래요.
오른손으로
쳐봐요.
(그냥 CCM
쓰세요)
아니,
무슨
답이
3초만에
나오는
줄
아세요?
임신부
10명이
있다고
애
한달만에
안나와요
1+1이
얼만지
계산해
달라
고요?
얼마에
맞춰
드리면
되죠?

삽질의 3요소

* 본 요소 구분은 지극히 주관적인 의견입니다

無我之境무아지경

모진 풍파와 외란에도 아랑곳 않고
본디 목표한 방향으로 정진하는 것.
Insanity 그거슨 destiny...

#물아일체 #무한루프

自我省察자아성찰

뒤 늦게 찾아낸 사소한 실수에
"나는 왜 이 모양인가?" 하고 자기
자신을 돌아보는 마음.

#여긴어디 #나는누구 #현자타임

溫故知新온고지신

과거의 학문과 실수의 기록들로부터
해법을 찾아가는 과정.
같은 실수는 되풀이 하지 말아야
(하지만 또 함)

#창의무한 #지식유한 #망각의동물

Previous Slide

Next Slides

삽질 연대기

삽질
허용기

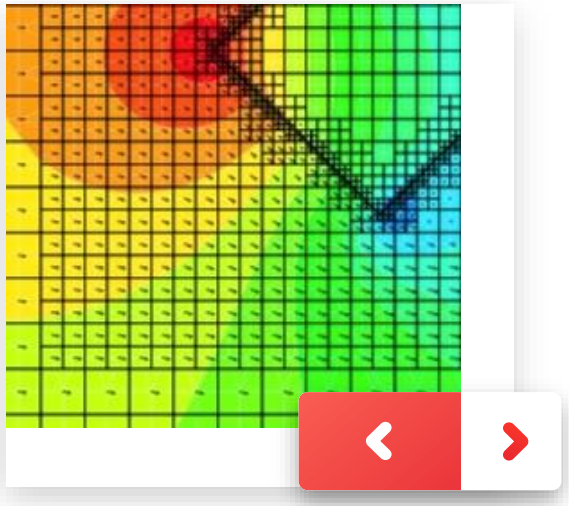
코드 개발

Object Oriented
Programming(OOP)
정렬 격자계 solver 개발
- Local refinement & IBM 적용

2010

OpenFOAM

잘못된 만남(?)



2013

중공업
입사

안정성 개선

OpenFOAM을 이용한 선박
저항 및 자항 해석 프로그램의
안정성을 개선하기 위한
프로젝트

Previous Slide

Next Slides

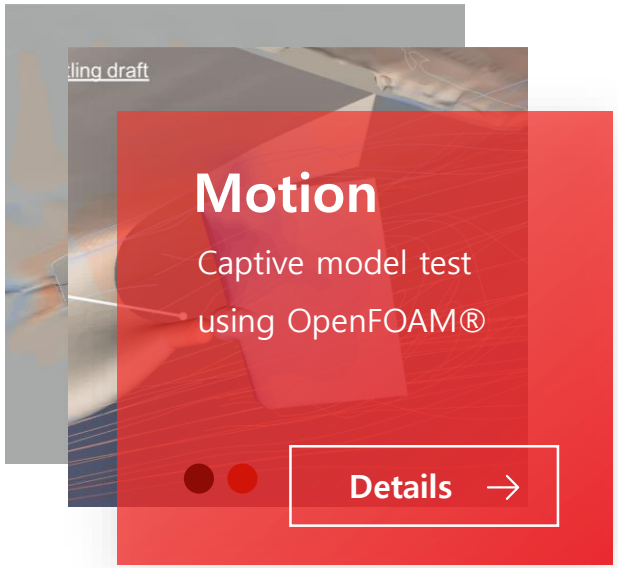
삽질 연대기

2014

NUMAP-FOAM

Zagreb 대학에서 개최되는
OpenFOAM summer
school.
쇄빙선 선저로 전개되는
VoF에 대한 review project

2017



Flux Limiter

안정적인 유동장 예측과
속도 개선을 위한 VoF field
flux limiter 개발 논문 발표

To be continued...

2019

- CFD + 기계학습
  이제 CFD는 기계가!
  설계도 기계가!

We are the world!

Previous Slide

Next Slides

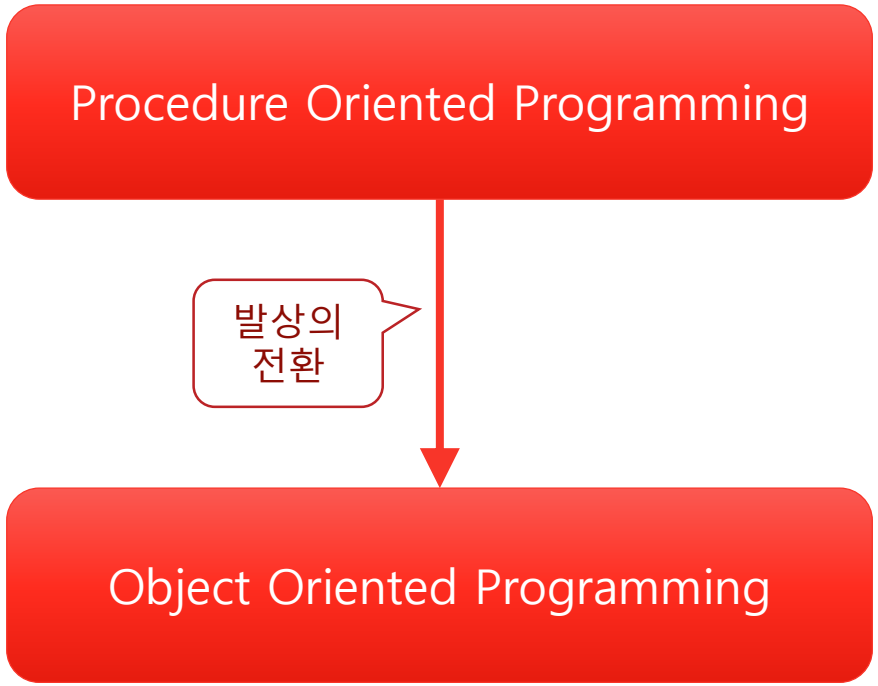
Shoveling on OpenFOAM Code Study

Description

2010년 이전까지는 FORTRAN을 이용한 [순차 지향 프로그래밍] 으로 연구나 해석을 수행하고 있었습니다. 특히 우리 연구실에서 전통적으로 구전되어온 Coupled LU-SGS Solver와 Dr. Peric 의 CAFFA Code를 이용하여 압축성, 비압축성 해석 연구와 개발을 진행하고 있었습니다.

그러다 우연히 연구실 선배에 의해 OpenFOAM을 접하고 공부를 하게 되는데, 완전히 다른 철학으로 작성된 코드는 신세계였습니다. 특히, 자료는 없고 물어볼 선배는 더더욱이 없고...그나마 절차 지향의 코드들은 들여다 보면 이해라도 되지만, 객체 지향은 이해조차 되지 않더군요.

결국, 삽(?)을 들었습니다.



Previous Slide
Next Slides

점진적 삽질

삽질에도 전략이 필요합니다.
무턱대고 아스팔트에 삽을 꽂아봤자
삽자루만 부러질 뿐. 여기저기
찔러보면서 삽이 들어갈만한 자리를
찾아야 합니다.

"여기다" 싶은 곳이 있으면 열심히
파 봅니다.

OOP의 개념 파악

일단 OOP가 무엇인지, 용어는 어떻게 다른지, 그리고 코드 구조와 철학은 어떻게 구성되는지 알아봅니다.

FORTRAN으로 OOP 구현

FORTRAN의 FREE FORMAT과 MODULE을 이용하여 [구전]되어 온 FORTRAN Solver의 OOP 버전을 만들어 봅니다.

CODE TRANSFORMATION

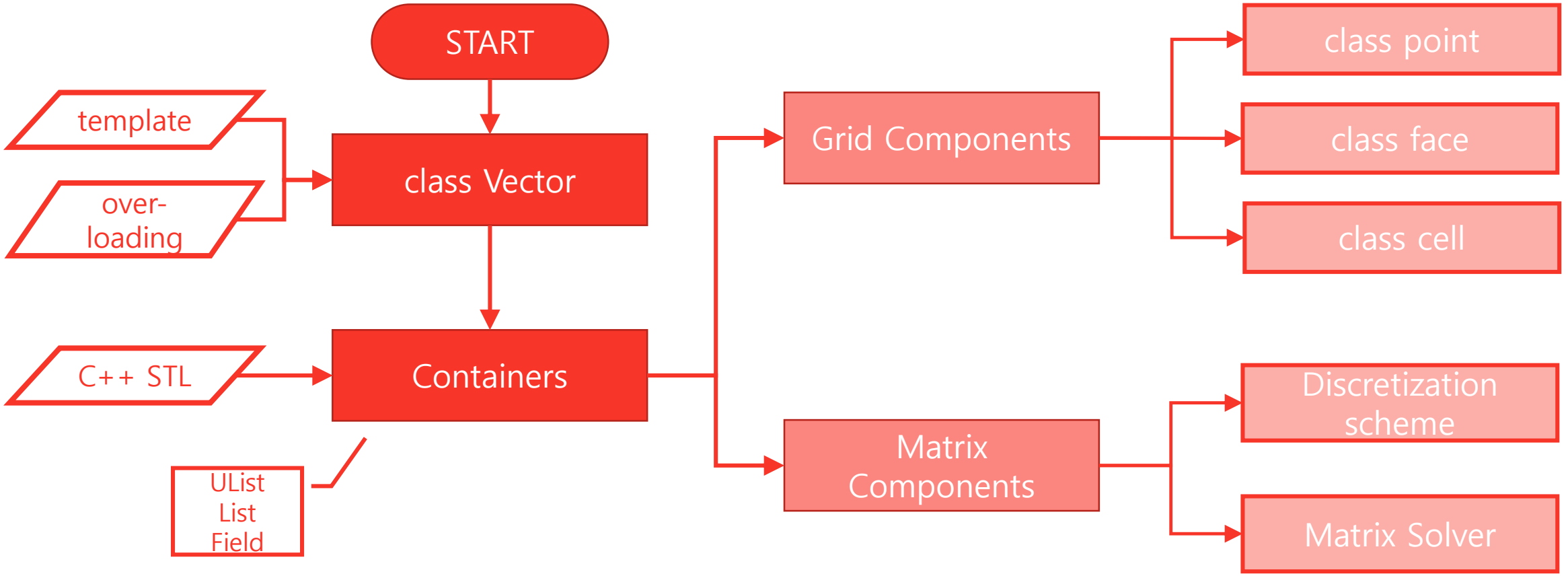
직접 C++ 또는 C# 을 이용해서 코드를 아예 새로 짜봅니다. 그리고 새로 만들어진 코드에 IBM을 심어서 논문도 한번 써 봅니다.

Previous Slide

Next Slides

CODE.TRANSFORMATION

Project ADLIB* (AeroDynamic Lab Immersed Boundary)



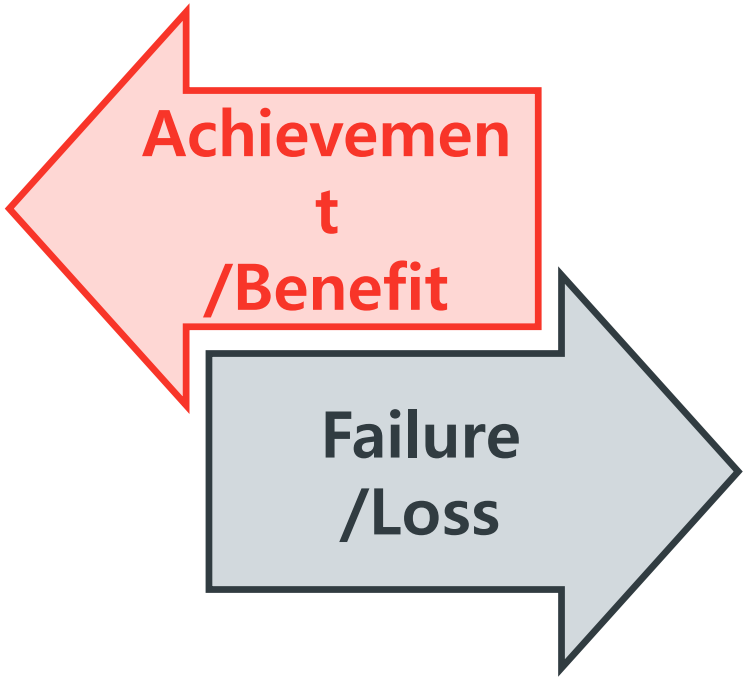
Previous Slide

* https://github.com/geonhong/adlib
github에 공개한 adlib(AeroDynamics Library)과는 다릅니다

Next Slides

LESSONS

- SIMPLE 알고리즘 기반 비압축성 비정렬 격자계 Solver 개발
- Local Refinement 및 Immersed Boundary Method 적용
- Non-conformal grid의 flux & pressure treatment
- Object Oriented Programming의 개념 이해
- OpenFOAM의 코드와 데이터 구조 파악
- 학회 2건, 학술지 1건



- Parallelization
- Runtime selection table
- Auto pointer 등 고급 skill
- 해석 성능
- 코드 작성하느라 허비한 시간 (쓰지도 못할 코드를...)
- ADLIB v1.0 (처음이자 마지막)

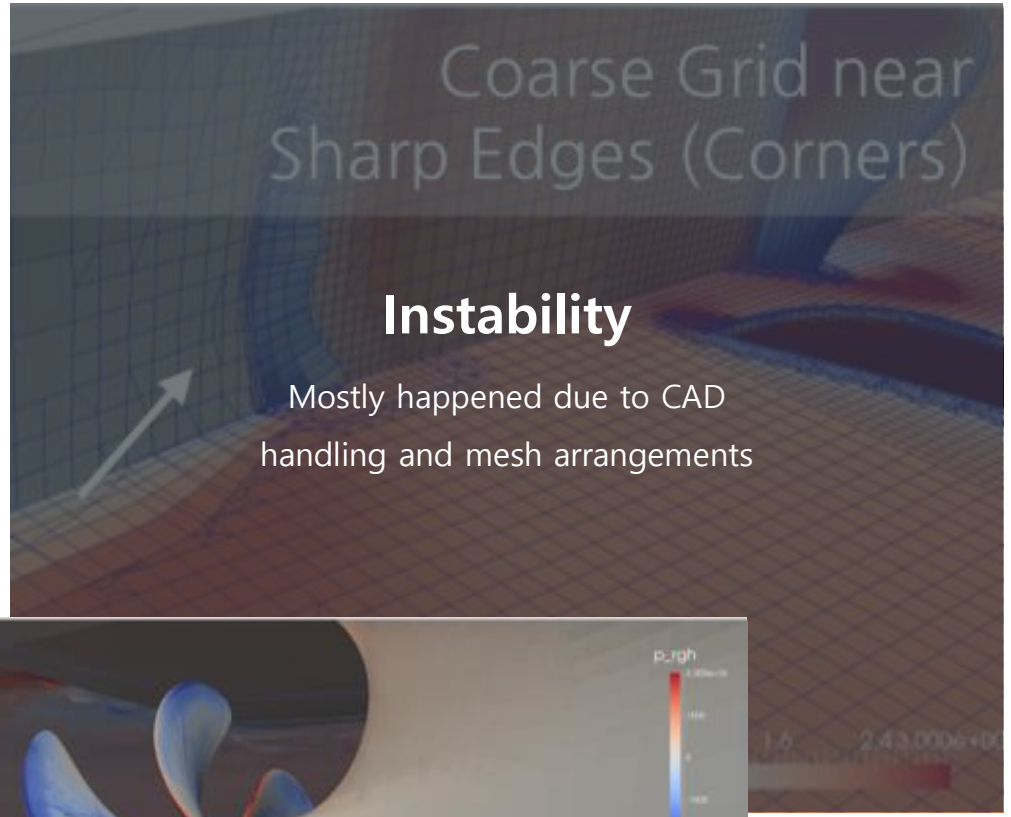
Previous Slide
Next Slides

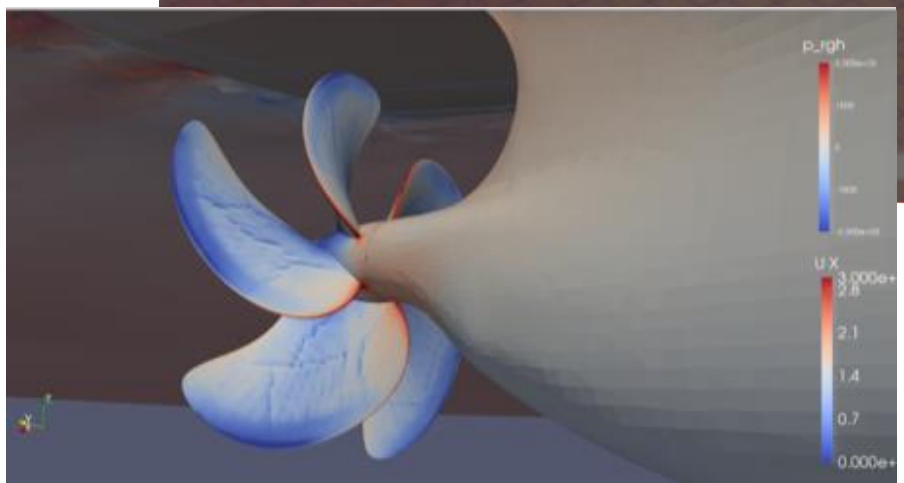
삽질 2. 해석 안정화

Description

2013년 7월 현대중공업에 입사하여 처음 받은 숙제가 안정적으로 선박의 저항 해석을 수행하는 것이었습니다. 당시 해석에 있어서 상당부분 해석이 발산하는 문제들이 발생하였는데, 이를 개선하는 것이 목적이었습니다.

이를 위해서 코드를 보다 본격적으로 뜯어보는, [심도 있는 삽질]이 시작되었습니다.





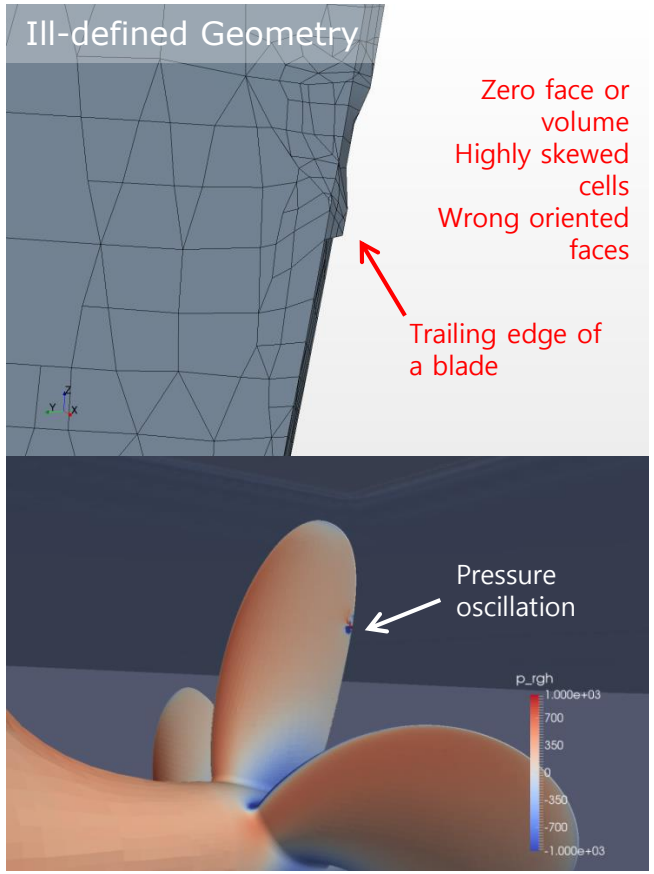
격자, 이산화 기법, 안정화 기법 등

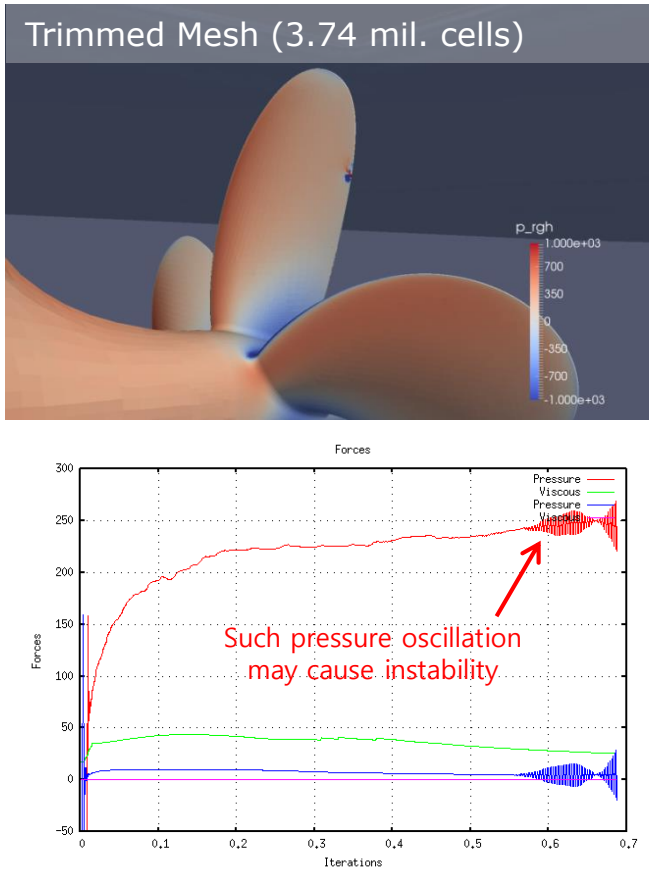
Previous Slide

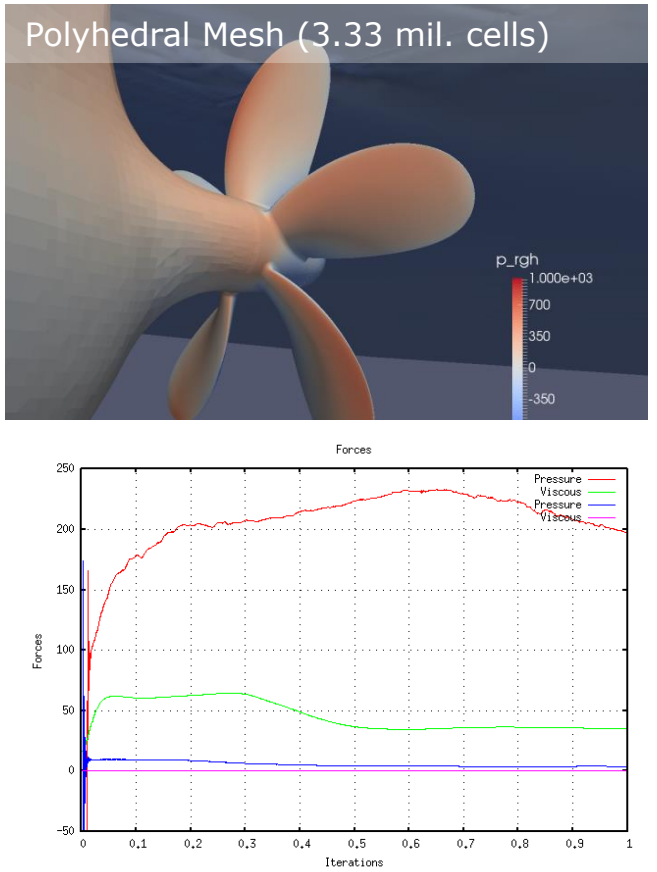
Next Slides

삽질 2. 해석 안정화

형상 또는 격자에서 야기되는 불안정성 - Easy





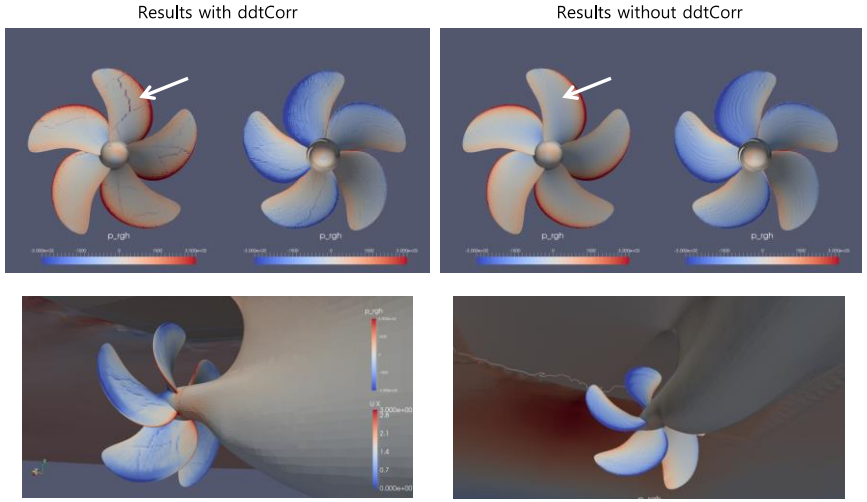


Previous Slide
Next Slides

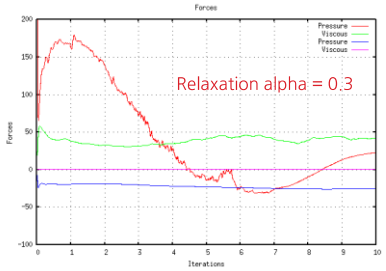
삽질 2. 해석 안정화

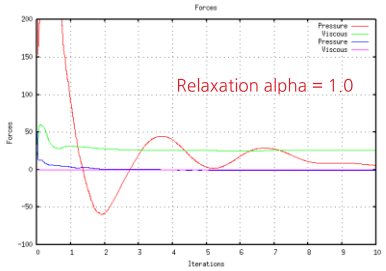
해석 기법 또는 파라미터에 의한 불안정성 - Normal

Issue 3. Effects of ddt Corrections

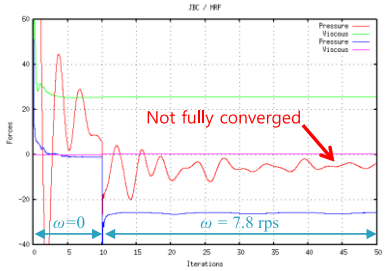


Issue 4. Effects of Relaxation alpha





- Solving VoF field without relaxation shows more smooth variation and rapid convergence of forces
- However the solution is not fully converged and keeps oscillating with small amplitude if relaxation is not applied



현상에 대한 이해를 바탕으로 해결책 모색 - 역지사지의 자세 견지 필요

Previous Slide
Next Slides

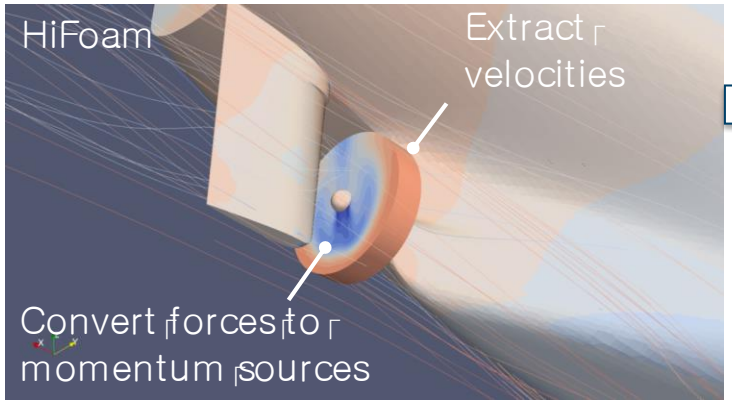
삽질 2. 해석 안정화

여러가지 요인이
복합적으로 작용하여
발생하는 불안정성
- Hard

NS 유동 해석을 위한 OpenFOAM과 프로펠러 해석을 위한 potential flow solver인 KPA 코드의 결합으로 자항 솔버 개발. 하지만 안정성이 형편없이 안좋았습니다.

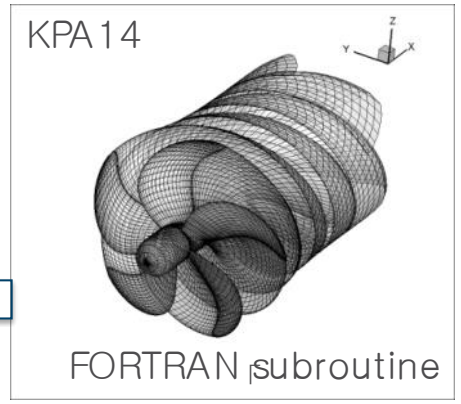
알고 있는 다양한 안정화 기법을 총동원하여 어떻게든 계산하게 만들었습니다.

The effect of the propeller was reflected by means of momentum source method where the momentum sources were estimated from KPA14 code.



Wake info.

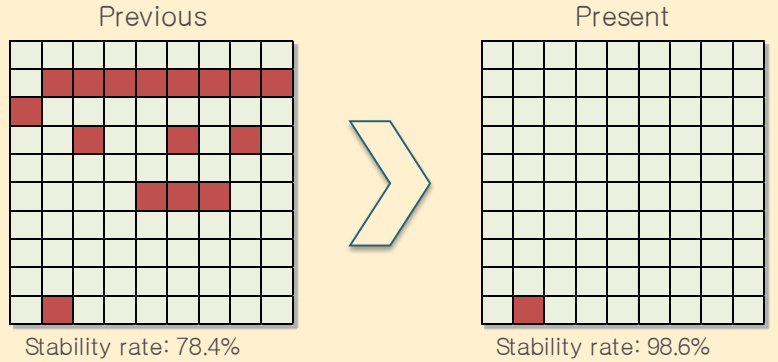
Force info.



- Enhanced accuracy
- Inflow model
- Viscous correction

- Hsin's method
- Relaxing potential for Kutta condition

Enhanced robustness



- Single case run in an hour with 108 cores (3 nodes)
- 2,880 jobs can be submitted at once

삽질 3. Two-phase Flow

Description

선박 성능 예측에 있어서 이상(two-phase) 유동의 정확하고 빠른 해석은 매우 중요합니다. 특히, OpenFOAM을 이용하여 STAR-CCM+ 와 같은 상용 도구를 대체하기 위해서는 필수적으로 점검해야 하는 요소입니다. 이상 유동에 있어서 아래와 같은 몇 가지 이슈가 있었습니다.

Air Jet near Freesurface

자유수면 부근의 공기가 급격히 가속되는 현상

Air ventilation into the water

공기가 물 속으로 선체를 따라 빨려 들어가는 현상

Limited time step size

cAlpha < 1.0 의 낮은 deltaT로 오랜 computation time

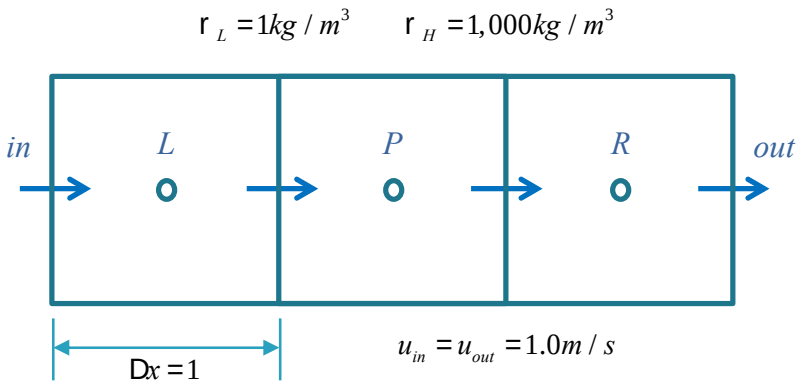
Previous Slide

Next Slides

TwoPhaseFlow::AirJet

간단한 사고 삽질실험을 해봅시다

To improve the stability of implicit VoF solver,
the induced flow of light fluid near the free surface has been considered.



For a simplicity, it is assumed that

- 1-D configuration with uniform grid
- pressure gradient is 0
- linear interpolation scheme
- upwind scheme for divergence operator
- Surface tension is negligible

A momentum equation can be simply written as,

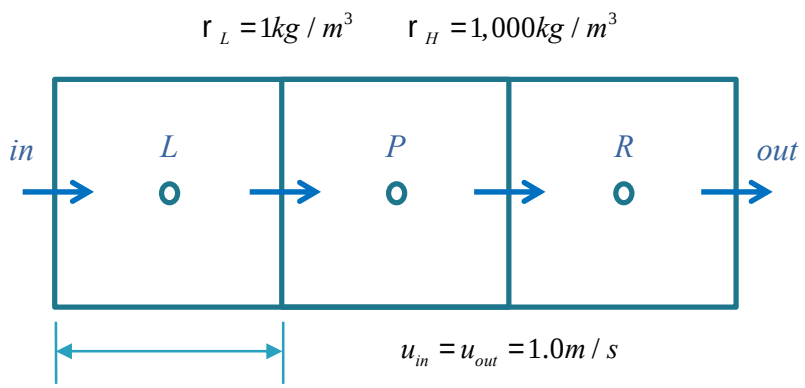
$$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho uu) = 0$$

Solve IMPLICITLY

Previous Slide

Next Slides

TwoPhaseFlow::AirJet



Solving for U yields,

Very small velocity may cause very high pressure fluctuation

$$\{u_L, u_P, u_R\}^{n+1} = \{0.0422, 0.8715, 0.9654\}$$

Dry cell next to the free surface is highly affected

Limiting face flux between L-P cells is required

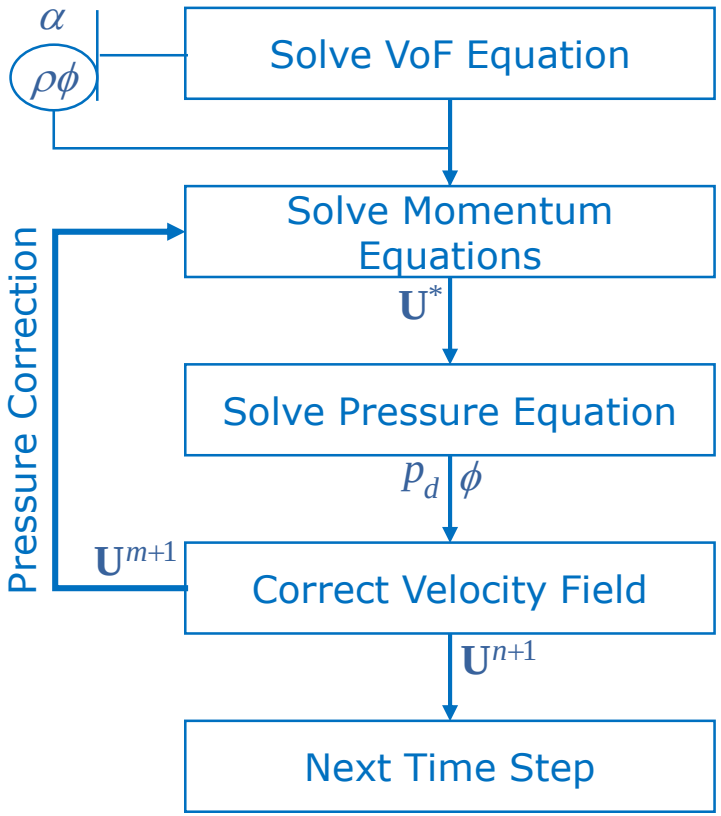
Pressure-correction 과정을 거치며 mass balance를 찾아 가지만, 많은 경우에 이러한 현상이 instability의 source로 작용하기도 합니다. Scheme과 VoF Solver를 적절히 선정하여 완화시킬 수는 있으나 완벽한 solution은 아니죠. 그래서 flux limiter를 적용하기로 마음먹고 열심히 팠습니다.

Previous Slide

Next Slides

TwoPhaseFlow::AirJet

- MULES solves for alpha and corrects phiAlpha, which is used to predict rhoPhi
- The rhoPhi is used to solve the momentum equations instead of phi
- Consequently, phiAlpha is the only quantity that affects to a solution in interFoam-like solvers of OpenFOAM, thus this should be limited to be bounded within the range of [0, phi] as well
- Limiting alpha solely is not a good choice

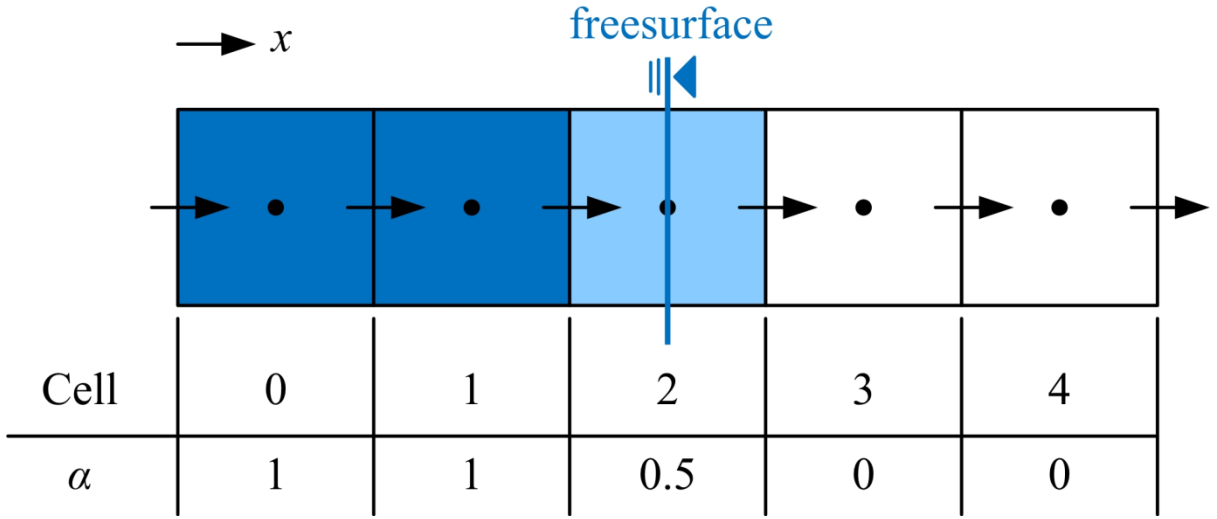


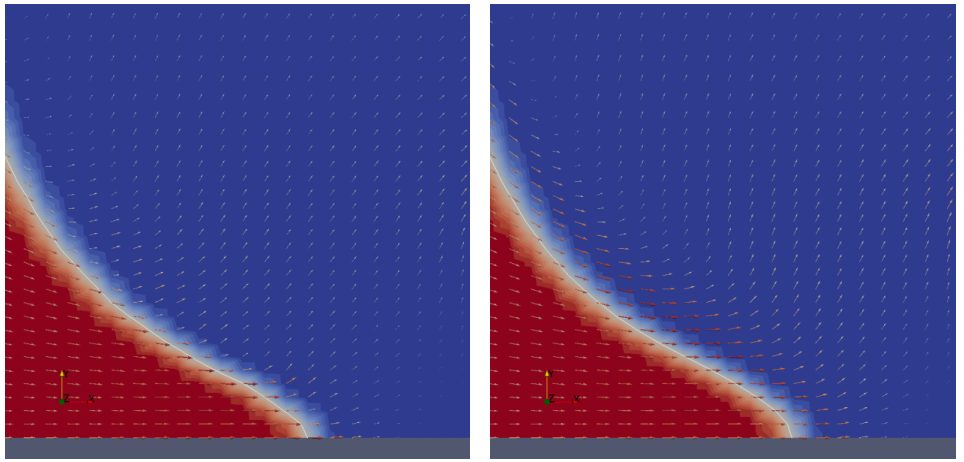
Cell centre에서 alpha값의 limiting 보다
Face 에서의 phiAlpha에 대한 limiting이 중요합니다.
특히, deltaT size가 커질수록 limiter에 크게 영향 받습니다.

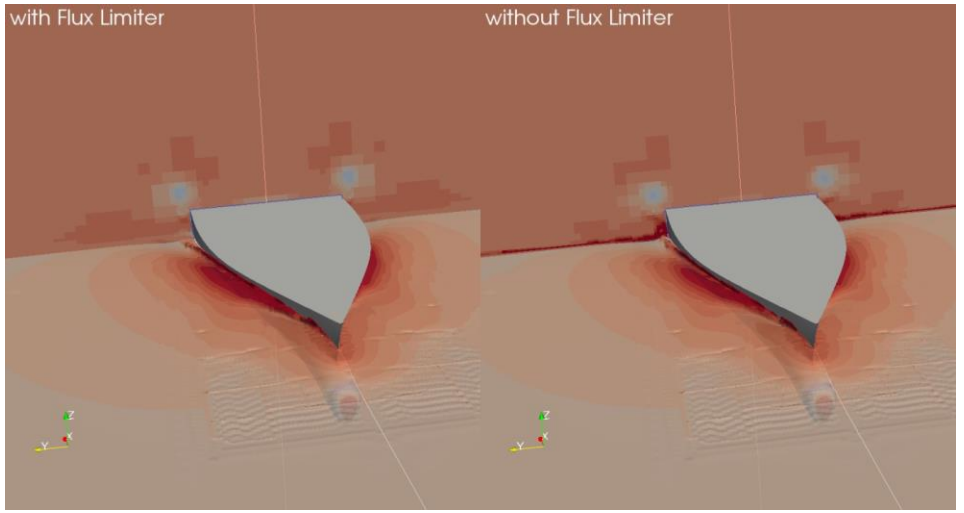
TwoPhaseFlow::AirJet

Simple flux limiter on cell faces

$$(\phi\alpha)_f = \phi_f \cdot pos(\bar{\alpha}_f - 0.5)$$







Face에서의 flux인 phiAlpha 값을 bounding 시키는 역할을 합니다.
단점은 volume을 잘라 먹습니다 (또 삽을 들고...).
Mass conserve가 중요한 유동에서는 비추입니다.

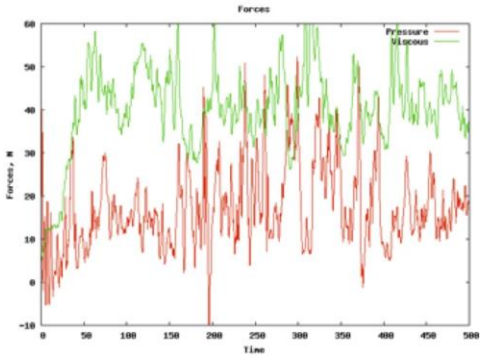
Previous Slide

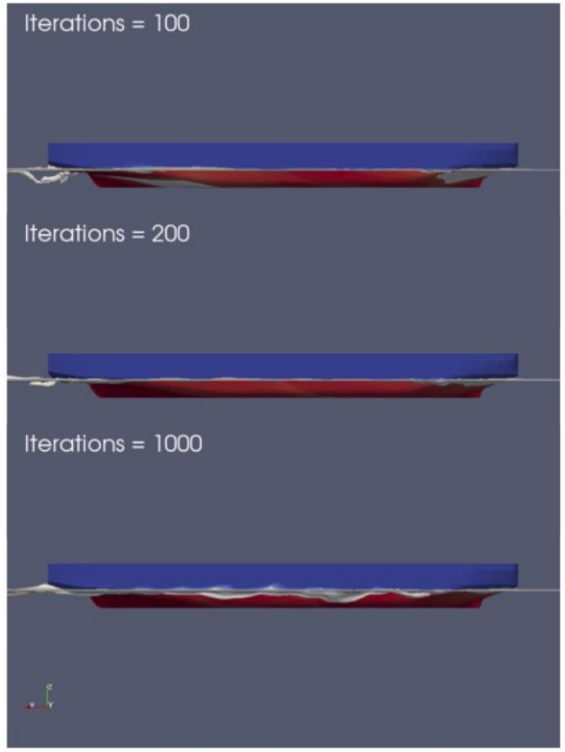
Next Slides

TwoPhaseFlow::AirVentilation

Validation of steadyNavalFoam

- steadyNavalFoam helped me to suppress the ventilation problem
- However the first try of running steady VoF simulation failed to get the converged solution
- So I began to complain to Vuko





매년 여름과 겨울 Zagreb 대학에서 주최하는 OpenFOAM School 인 NUMAP-FOAM(2014)에 참가했습니다. 약 2주간 하나의 주제를 가지고 short-term project를 진행하는데, 쇄빙선의 선저로 유입되는 공기 유동 문제(ventilation)을 해결하고자 하였습니다.

그런데, steady 상태에서의 해석조차 제대로 되지 않는 것입니다.

...그래서 어쩔 수 없이 또 팠습니다.

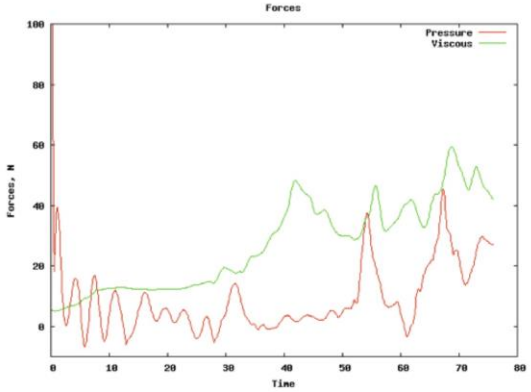
Previous Slide

Next Slides

TwoPhaseFlow::AirVentilation

Test simulation step by step :: case1000

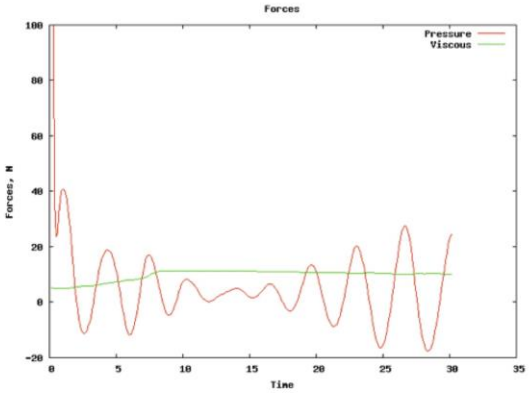
- Change the grad scheme only
- `leastSquare` to `cellLimited leastSquare 1`
- The forces are not converged with the grad scheme solely changed

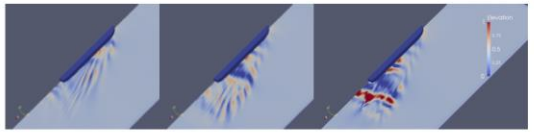


Scheme도 의심해 보고...

Test simulation step by step :: case1100

- Based on the case1000, reduced `limitMagU` from 10 to 6
- This option limits maximum magnitude of the velocity U. If the magnitude of velocity of a cell is larger than the specified limit value, then it scales the velocity components by the factor of `lmitMagU/mag(U)`
- It shows more stable behavior than case1000. Before Time = 15, it seems to be converged but later the pressure force begins to oscillate and the amplitude increases





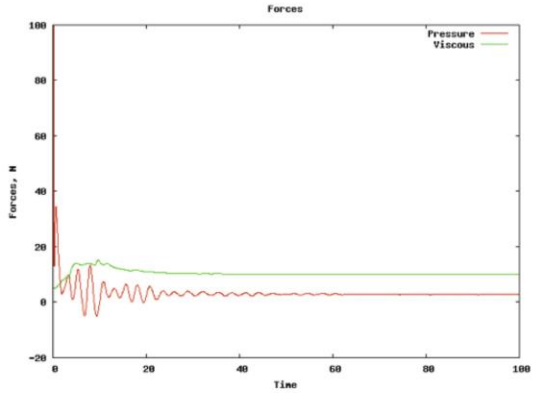
Velocity limiter도 적용해 보고

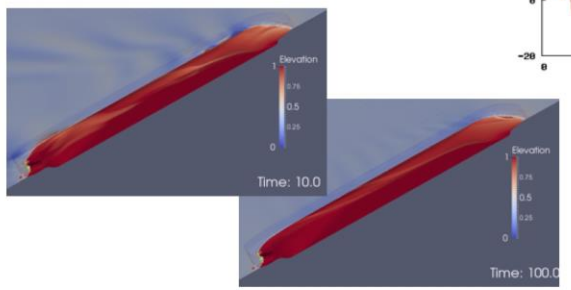
Previous Slide
Next Slides

TwoPhaseFlow::AirVentilation

Test simulation step by step :: case1110

- Grad scheme is changed, limitMagU is reduced, and the time step size is decreased form 0.1 to 0.05.
- The computation converged after Time = 60 where the computation iterates 120 steps.



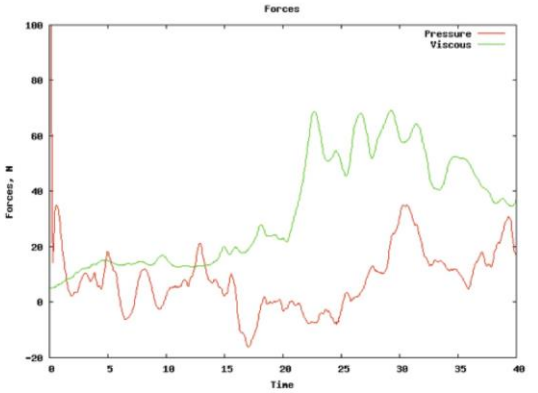


The ventilation has been suppressed, but still air mixture is found in the forward and corner of the hull where surface curvature is relatively large.

deltaT도 줄여 보니
#성공적

Test simulation step by step :: case0010

- Then, is `deltaT` the most important factor affecting the convergence?
- If `deltaT` is reduced only, the computation does not converged.
- We need to set up appropriate `limitMagU` value as well as `deltaT`, which may be connected to the Courant number



하지만, Main factor 찾기는
실패

Previous Slide

Next Slides

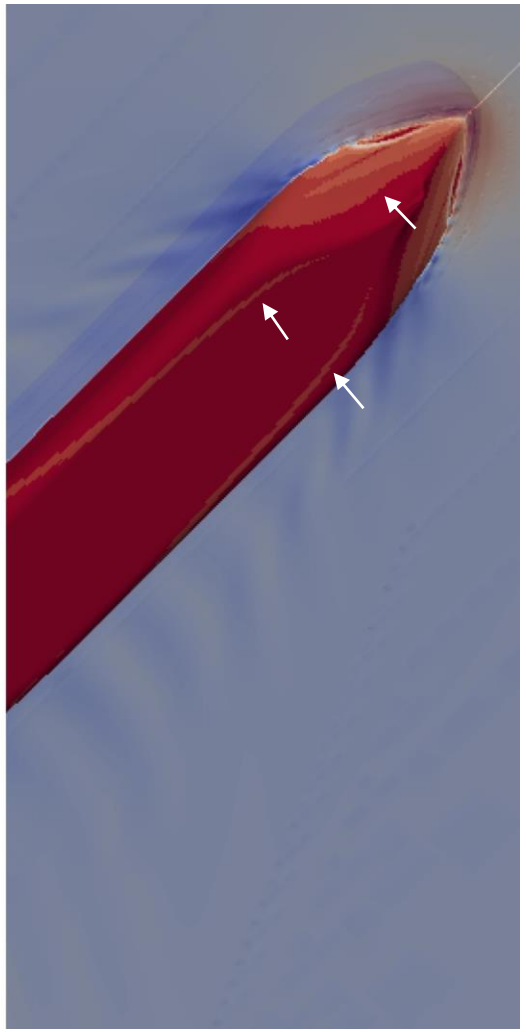
TwoPhaseFlow::AirVentilation

Trace cells with air mixture

- In the source code,

```
// Trace boundary cells on the hull immersed in the water
// with air mixture
const fvBoundaryMesh& boundary = mesh.boundary();
forAll(boundary, i)
{
    if (boundary[i].name() == "hull")
    {
        forAll(boundary[i].faceCells(), faceI)
        {
            const label& cellI(boundary[i].faceCells()[faceI]);

            if
            (
                alpha1[cellI] < 0.95
             && mesh.C()[cellI].z() < -0.332
             && mesh.C()[cellI].y() < 0.25
            )
            {
                Pout<< "Cell " << cellI << "\t"
                    << "y = " << mesh.C()[cellI].y() << "\t"
                    << "z = " << mesh.C()[cellI].z() << "\t"
                    << "alpha1 = " << alpha1[cellI]
                    << endl;
            }
        }
    }
}
```



어쨌든 계산이 돌아가게 만들었으니,

선저에 형성되는 공기층이 "왜" 생기며, "어떻게" 하면 없앨 수 있을지 고민해 봅니다.

여기서 선택한 방법은,

"한 놈만 팬다"

문제가 되는 cell 하나를 선택하고, 이 cell에서 VoF가 결정되는 과정을 추적해 봅시다.

TwoPhaseFlow::AirVentilation

Decompose the VoF equation and monitor values of each components

- In the `alphaImplicit.H`,

```
fvScalarMatrix alpha1Eqn
(
    fvm::ddt(alpha1)
  + fvm::div(phi, alpha1, alphaScheme)
  + fvm::SuSp(-fvc::div(phi), alpha1)
  + fvm::div
    (
        -fvc::flux(-phir, scalar(1) - alpha1, alpharScheme),
        alpha1,
        alpharScheme
    )
);
```

This can be separated into sub-operations as,

```
// Surface scalar field components
volScalarField fvcDiv_phi(fvc::div(phi));
surfaceScalarField Ur
(
    fvc::flux(-phir, scalar(1)-alpha1, alpharScheme)
);

// Matrix components
fvScalarMatrix ddt_alpha1(fvm::ddt(alpha1));
fvScalarMatrix div_phi_alpha1(fvm::div(phi, alpha1, alphaScheme));
fvScalarMatrix SuSp_divPhi_alpha1(fvm::SuSp(-fvcDiv_phi, alpha1));
fvScalarMatrix divUrAlpha1(fvm::div(-Ur, alpha1, alpharScheme));
```

지배 방정식을 각 component(ddt, div, source 등) 별로 분해합니다.

Decompose the VoF equation and monitor values of each components

```
### Cell info. ###
Cell 16597 consists of 6
List of cell faces :6(48404 48405 4307606 48391 48398 48400)

### Volume field info. ###
alpha1 = 0.9484678462
fvc::div(phi) = 0.0007579478242
--- Matrix coefficients ---
Operation       Diagonal        source
ddt(alpha1)                     1.680870318e-05   1.59425145e-05
div(phi, alpha1)                1.63924365e-05    -9.20592076e-08
SuSp(-fvc::div(phi),alpha1)     0                 6.041797088e-10
div(-Ur x (1-alpha1), alpha1)   3.434479221e-07   0
alpha1Eqn                       3.35445876e-05    1.585105947e-05

### Surface field info. ###
--- Face field values ---
faceI   isOwnFace       phi     phir    Ur      lduAddr.lower   alpha[nextCell]
48404   1       -5.594932724e-06        4.07823265e-06  -7.026082638e-08        16597   0.9848908049
48405   1       6.097555705e-06         6.43006981e-06          -1.2519581e-07  16597   0.9830495798
48391   0       1.079686677e-05         -1.983362978e-06        6.180325739e-08         16593   0.9712266591
48398   0       -1.242039896e-06        -3.024125633e-07        9.244846093e-09         16595   0.9718214391
48400   0       -9.052840899e-06        -2.742931873e-06        1.106468799e-07         16596   0.9614490235

--- lower coefficients ---
faceI   isOwnFace       ddt(alpha1)     div(phi, alpha1)        SuSp(-fvc::div(phi, alpha1))    div(-Ur x (1-alpha1), alpha1)   alpha1Eqn
48404   1       0       0       -6.623312461e-08        -6.623312461e-08
48405   1       0       -6.097555705e-06        0       -1.181857182e-07        -6.215741423e-06
48391   0       0       -1.079686677e-05        0       6.561276622e-09         -1.079030549e-05
48398   0       0       0       0       9.675932122e-10         9.675932122e-10
48400   0       0       0       0       1.513703424e-08         1.513703424e-08

--- upper coefficients ---
faceI   isOwnFace       ddt(alpha1)     div(phi, alpha1)        SuSp(-fvc::div(phi, alpha1))    div(-Ur x (1-alpha1), alpha1)   alpha1Eqn
48404   1       0       -5.594932724e-06        0       4.027701766e-09         -5.590905022e-06
48405   1       0       0       0       7.010091763e-09         7.010091763e-09
48391   0       0       0       0       -5.524198077e-08        -5.524198077e-08
48398   0       0       -1.242039896e-06        0       -8.277252881e-09        -1.250317149e-06
48400   0       0       -9.052840899e-06        0       -9.550984565e-08        -9.148350745e-06

Terminate program
```

그리고 cell centre와 face에서 분해된 각 matrix의 component들의 값들을 일일이 비교해 봅니다.

TwoPhaseFlow::AirVentilation

Decompose the VoF equation and monitor values of each components

$$\mathbf{A}_P\alpha_P + \sum_L \mathbf{A}_L\alpha_L = \mathbf{R}_P \quad \text{or} \quad \alpha_P = \frac{1}{\mathbf{A}_P}\left(\mathbf{R}_P - \sum_L \mathbf{A}_L\alpha_L\right)$$

Face Index	next alpha1	A	A x alpha1
391	0.97123	-1.07903E-05	-1.04798E-05
398	0.97182	9.67593E-10	9.40328E-10
400	0.96145	1.51370E-08	1.45535E-08
404	0.98489	-5.59091E-06	-5.50643E-06
405	0.98305	7.01009E-09	6.89127E-09
			-1.59639E-05

Diagonal	3.35446E-05
Source	1.58511E-05

R - sum(A x alp)	3.18149E-05
alpha1	**0.94843729**

이제, Excel로 다음 step에서의 alpha 값을 계산해 봅시다.

Decompose the VoF equation and monitor values of each components

Volume Field Info

alpha1	0.948467846
fvc::div(phi)	0.000757948

--- Matrix Coefficients ---

Operations	Diagonal	Source
ddt(alpha1)	1.68087E-05	1.59425E-05
div(phi,alpha1)	1.63924E-05	-9.2059E-08
SuSp(-fvc::div(phi),alpha1)	0	6.04180E-10
div(-Urx(1-alpha1),alpha1)	3.43448E-07	0
alpha1Eqn	**3.35446E-05**	**1.58511E-05**

Reduce $\mathbf{A}_P$

Increase $\mathbf{R}_P - \sum_L \mathbf{A}_L\alpha_L$

--- coefficients ---

facel	isOwnFace	ddt(alpha1)	div(phi,alpha1)	SuSp(-fvc···)	div(-Ur···)	alpha1Eqn
48404	1	0	-5.5949E-06	0	4.02770E-09	-5.5909E-06
48405	1	0	0	0	7.01009E-09	7.01009E-09
48391	0	0	-1.0797E-05	0	6.56128E-09	-1.0790E-05
48398	0	0	0	0	9.67593E-10	9.67593E-10
48400	0	0	0	0	1.51370E-08	1.51370E-08

- `div(phi, alpha1)` may be limited further than it really is required
- `div(-Ur, alpha1)` has small values but acts in the way to reduce the alpha1. `Ur` can be controlled by `cAlpha` and therefore it can be inferred that the reduced `cAlpha` may help to suppress the air ventilation further

div(phi,alpha1) 값이 main factor가 되겠군요. 아마도 relative compression term으로 control 가능하겠군요 (호오~)
가변 cAlpha면 문제를 해결할 수도 있어 보입니다.

TwoPhaseFlow::AirVentilation

이렇게,
SFD (Shoveling Fluid Dynamics)
해보면 문제를
해결할 수 있답니다.

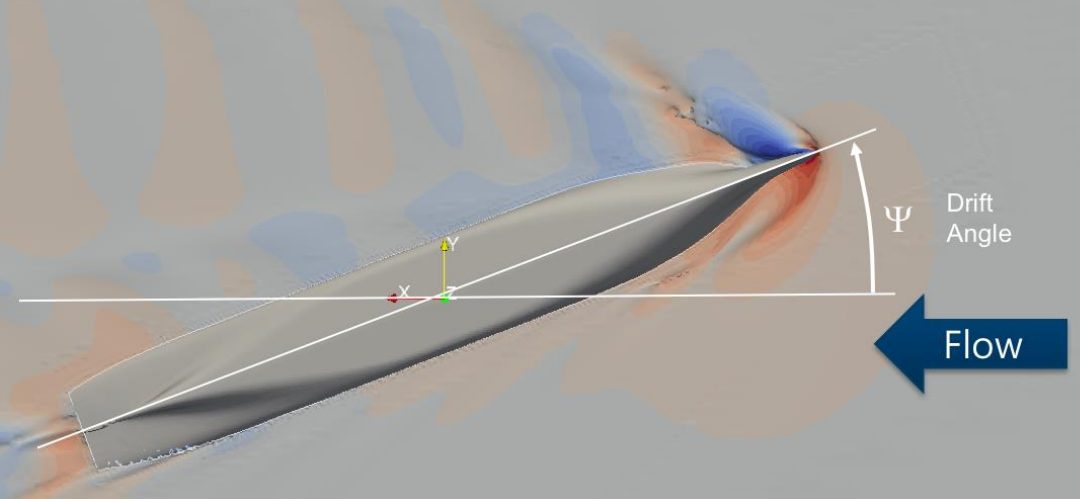
아,
물론 저는 이제
하지 않을 겁니다.
더이상은 NAVER...

Previous Slide
Next Slides

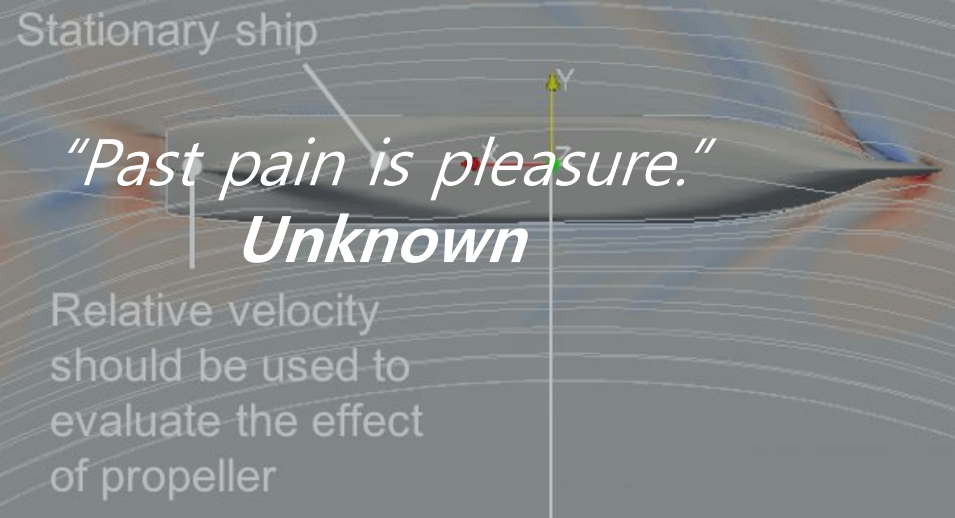
Limited Time Step Size

NEED FOAM SPEED

안정적이고 빠른 해석을 위해서는 더 큰 time step size를 사용하는 것이 필수적입니다. 하지만 OpenFOAM에서 deltaT를 키우는 것은 쉬운 일이 아니죠.



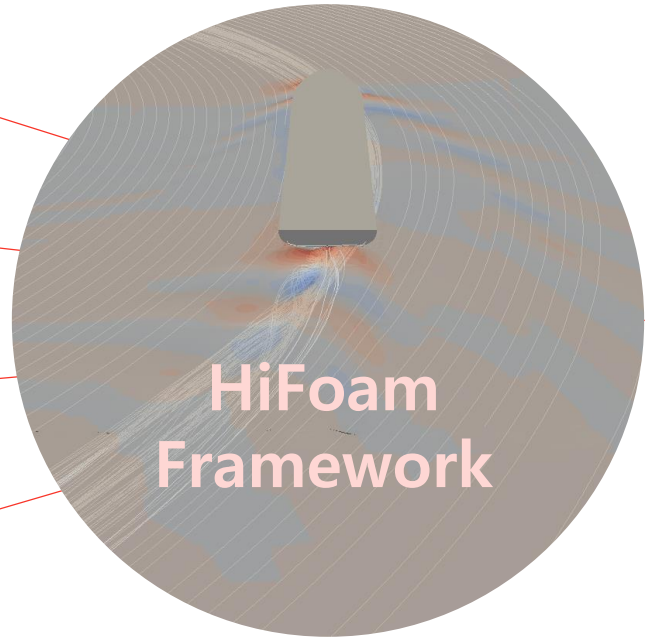
- *It is essential to apply large and fixed time step size for performing efficient manoeuvring simulations*



"Past pain is pleasure."
Unknown

LimitedTimeStepSize

- Flux Limiter
- Velocity Limiter
- QS-FBI
- Mesh cleanup



+ Viscous-inviscid coupling
+ swense

- Resistance simulations in ~30 minutes
  (with 72 cores)
- Self-propulsion simulations in an hour
- 99% stable
- 2~5% accuracy in average
- Resistance, self-propulsion, manoeuvring, added resistance etc

Previous Slide

Next Slides

Concluding Remarks

Next Slides

2019

EVERY DAY I'M SHOVELING

SMART SHOVELING

in OpenFOAM perspective

계산이 터졌습니다.
어떻게 하시겠습니까?

boundary condition을 바꾼다

격자를 다시 짠다 (prism layer, refinement 등)

fvSolution을 연다 (nOuterCorrectors 등)

간절하게 빌어본다

간절히 바라는 일은 이루어지지 않습니다. 이루어질 일이었다면 애초에 간절히 바라지도 않았겠죠

"근본적인 고민 없는 삽질은 맨손으로 콘크리트를 파겠다는 것과 같다"

Some Lessons from Shoveling

원론적인, 그러나 중요한

실패의 기록이 필요하다

- 인간은 망각의 동물이라 한번 한 실수는 반드시 다시 합니다 (역사는 반복된다)
- 실패를 줄이기 위해서는 실패했던 기록들이 정리되어야 하겠습니다
- 무엇을 실패했는가? 어떻게 실패했는가? 왜때문인가?

Computer-like한 사고가 필요하다

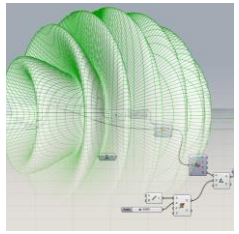
- 왜 문제가 발생했는지 파악하기 위해서는 컴퓨터가 생각하는 방식을 고민할 필요가 있습니다
- 어떠한 삽질을 할 때 요행을 바라며 그냥 이루어지기를 기대하는 경우가 많더군요
- 컴퓨터는 인간처럼 생각하고(인공지능), 인간은 컴퓨터처럼 생각하고(기계지능)

문제 없이 돌아가는 계산은 두 번 의심하라

- 계산이 돌아간다고? 왜? "제대로" 돌아가고 있는 건가?
- 엔지니어로써 CFD 결과를 책임질 수 있는가?
- 구체적이고 명확한 근거 없는 해석은 오히려 독이 될 수 있습니다

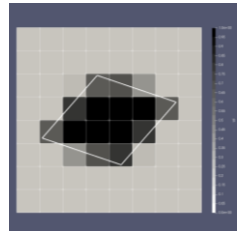
FUTURE SHOVELING

For a shape design and optimization…

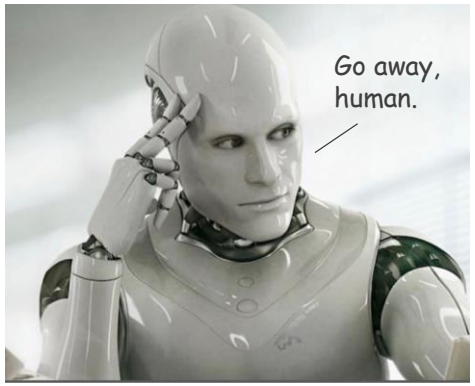


OpenFOAM

TensorFlow



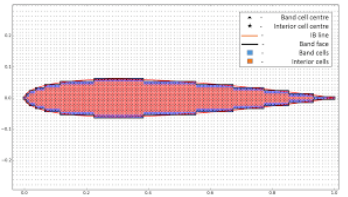
User Dependency

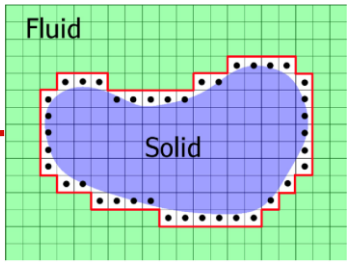


◀ They will work for us.
Veteran designer will live in a machine forever…

Difficult to Parameterize

Employing the concept of the Immersed Boundary Method (IBM)
- Especially, the cut-cell method





Optimal Hull Form

Machine Learning

Pixelated Volume Fraction

FUTURE SHOVELING

I am trying to develop the techniques to FIRE designers

Design Anyone

Expert designer's experience has already melted into the system

Design Anywhere

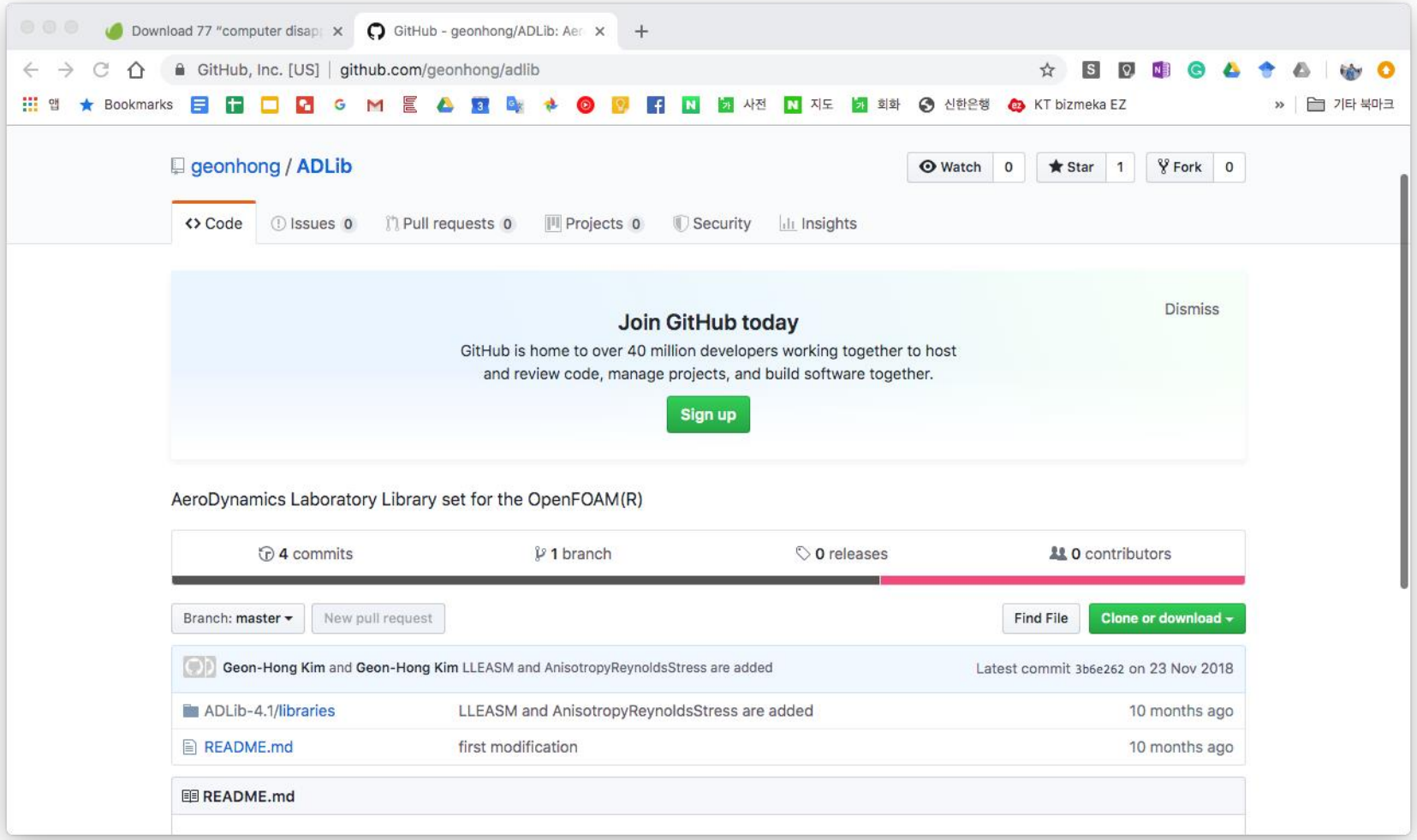
No professional and high-cost design tool is required

Quality Guarantee

The same output whoever design an object

FUTURE SHOVELING

Private project



project

ADLib

AeroDynamics lab
Library

KAIST 공기역학연구실 선후배들의 연구 내용이 사장되는 것이 아까워 OpenFOAM Library로 converting 시키는 프로젝트.
Github을 통해 공개

에필로그

내가 CFD 생활을 2010년에 시작했다
그 때 CFD 시작한 현중 동기가 백명이라 치면
지금 나 만큼 쓰는 놈은 나 하나 뿐이야
나는 어떻게 여기까지 왔냐
터지는 격자 제끼고
안 맞는 놈 보내고
adjustableTimeStep쓰는 속도 느린 계산들, 다 죽였다

(라이센스가 막힐 것 같습니다)

OpenFOAM은 무너졌냐?

Thank You

Geon-Hong Kim
Hyundai Heavy Industries Co., Ltd.

2019