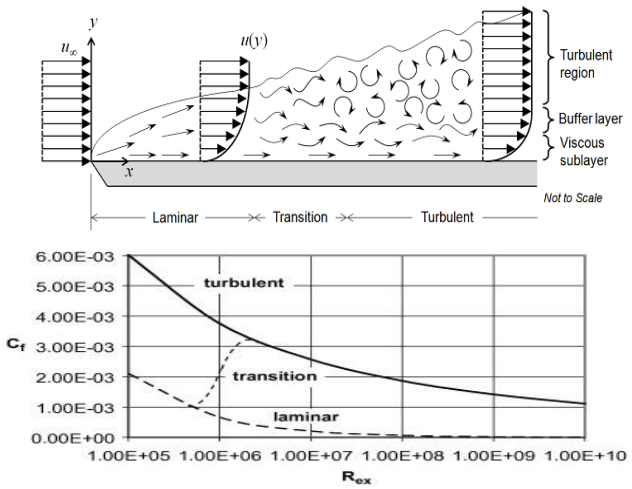
조선해양 분야의 OpenFOAM 활용 사례 및 코드 개발 과정

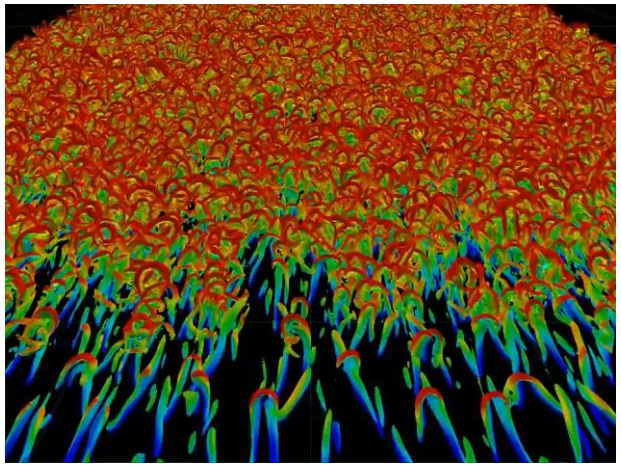
이상봉

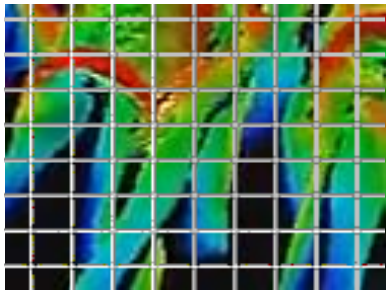
동아대학교 조선해양공학과

OpenFOAM을 이용한 CFD

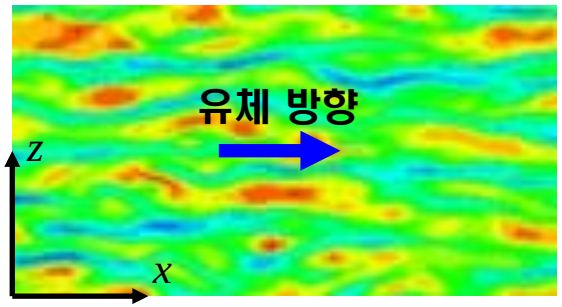
✓ **전산유체해석의 종착역**







격자 크기 기준?



$\Delta x^+ = \Delta x \frac{V}{\nu}\sqrt{0.5 c_f}$

Δx^+=15

Δy^+<0.2 with SR~1.1

Δz^+=5

DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM을 이용한 CFD

✓ **선박에 대한 전산유체해석의 종착역**

모형선 KCS 기준 (LBP=7.3m, Vm=2.1962m/s)

- Nx=30,000개 (Δx=0.25mm)
- Ny=128개 (Δy=0.25μm)
- Nz=30,000개 (Δz=0.06mm)

총 1,150억개의 격자 구성

실선 KCS 기준 (LBP=230m, Vs=24knots)

- Nx=1,000,000개 (Δx=0.25mm)
- Ny=320개 (Δy=0.25μm)
- Nz=1,000,000개 (Δz=0.06mm)

총 320조개의 격자 구성

코어당 10k개의 격자 사용 : 1,150만 코어, 320억 코어를 갖춘 HPC

LES를 사용한다면 2-4배의 성긴 격자 사용 가능 (3차원 기준 8-64배의 격자 감소 효과)

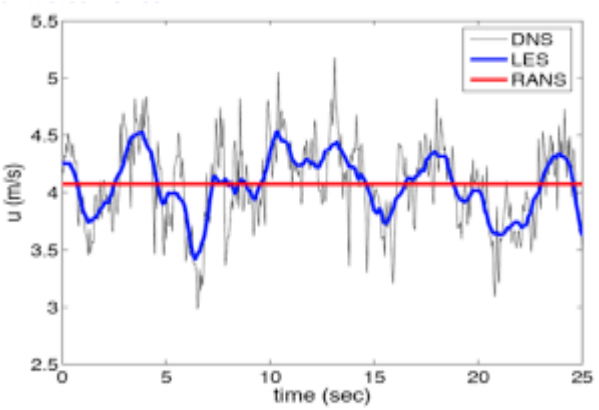
총 18억개의 격자 구성

총 5조개의 격자 구성

여전히 비현실적인 해석 비용 발생 → Reynolds averaged Navier-Stokes (RANS) 방정식

OpenFOAM을 이용한 CFD

✓ 전산유체해석의 현실적인 타협 – 난류모델항



$$\rho\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)=0$$

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-\frac{\partial p}{\partial x}+\mu\left(\frac{d^2u}{dx^2}+\frac{d^2u}{dy^2}+\frac{d^2u}{dz^2}\right)$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial y}+\mu\left(\frac{d^2v}{dx^2}+\frac{d^2v}{dy^2}+\frac{d^2v}{dz^2}\right)$$

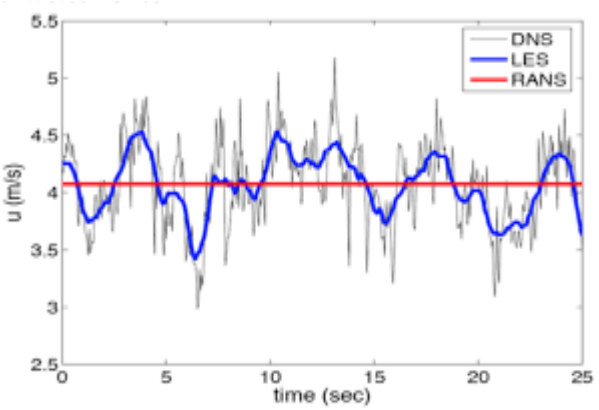
$$\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=-\frac{\partial p}{\partial z}+\mu\left(\frac{d^2w}{dx^2}+\frac{d^2w}{dy^2}+\frac{d^2w}{dz^2}\right)$$

+ TM

- ✓ **DNS**
  - TM=0에 해당하는 경우
  - 조밀한 격자에서는 정확한 유동 특성(속도, 압력, 속도 변동량 등등)을 제공
  - 성긴 격자에서는 평균 속도 분포에서도 큰 오차 발생 (저항 오차 야기)
- ✓ **LES**
  - 난류모델항에 격자 간격에 대한 정보 포함
  - DNS 수준으로 조밀한 격자에서는 난류모델항이 0에 수렴 (DNS와 동일한 결과)
  - 성긴 격자에서는 DNS 정도는 아니지만, 어느 정도의 오차 발생 (저항 오차 야기)

OpenFOAM을 이용한 CFD

✓ **전산유체해석의 현실적인 타협 – 난류모델항**



$$\rho\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)=0$$

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-\frac{\partial p}{\partial x}+\mu\left(\frac{d^2u}{dx^2}+\frac{d^2u}{dy^2}+\frac{d^2u}{dz^2}\right)$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial y}+\mu\left(\frac{d^2v}{dx^2}+\frac{d^2v}{dy^2}+\frac{d^2v}{dz^2}\right)$$

$$\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=-\frac{\partial p}{\partial z}+\mu\left(\frac{d^2w}{dx^2}+\frac{d^2w}{dy^2}+\frac{d^2w}{dz^2}\right)$$

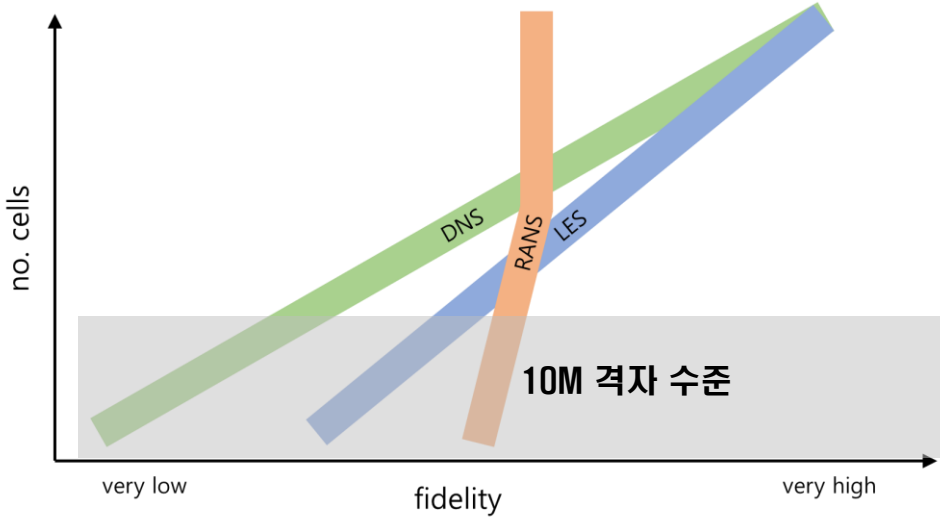
+ TM

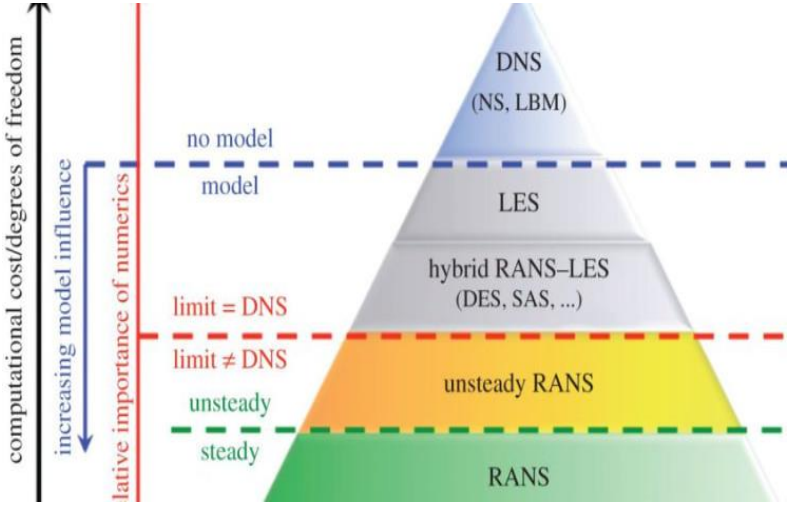
✓ **RANS 해석**

- 난류모델항에 격자 크기 정보 불포함 (one-point model)
- 조밀한 격자에서도 난류모델항의 크기가 존재하므로 오차 발생
- 매우 성긴 격자에서도 난류모델항이 어느 정도 유효하게 작동

OpenFOAM을 이용한 CFD

✓ **전산유체해석의 개선 방향 (격자 조밀도 개선)**



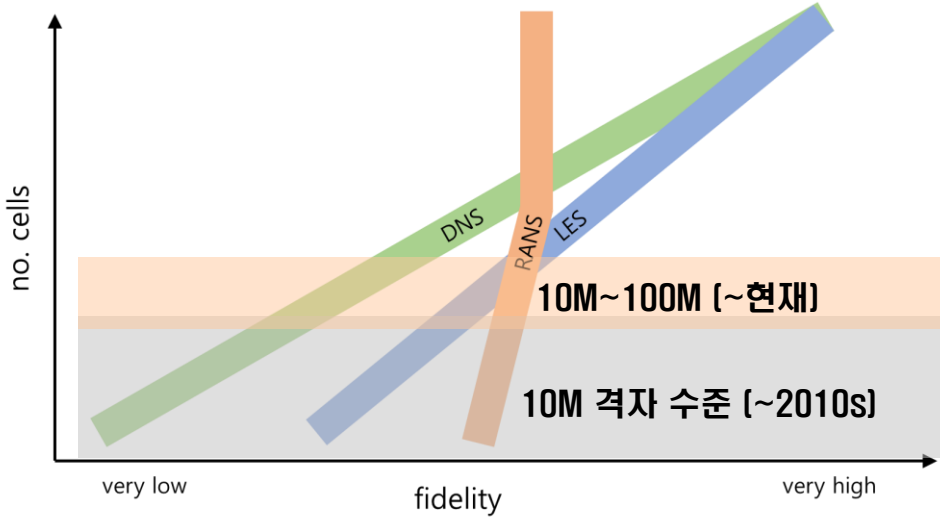


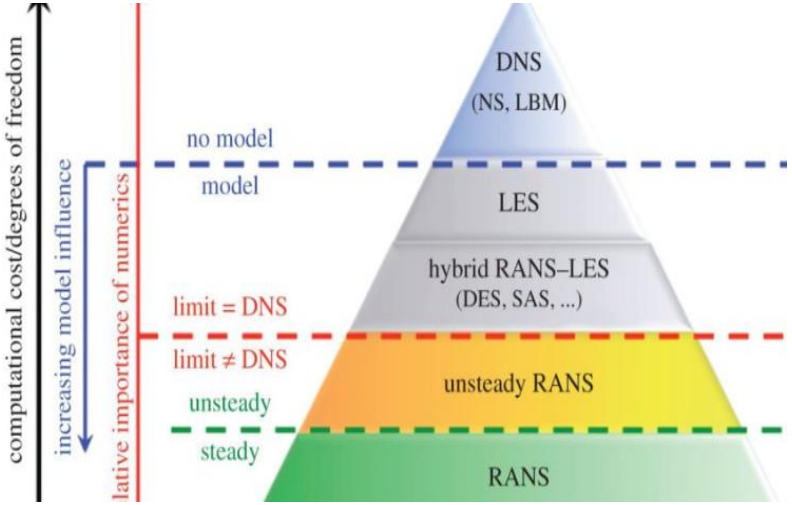
✓ **RANS를 이용한 CFD 해석 결과**

- **격자수를 증가시킬수록 오차가 감소하고…**
- **설계 형상 변경에 따른 영향이 반영되는 것 같으니…**
- **H/W 용량 대폭 확대 및 격자 수(천만개 수준의 격자 목표)를 증가시켰는데…**

OpenFOAM을 이용한 CFD

✓ **전산유체해석의 개선 방향**



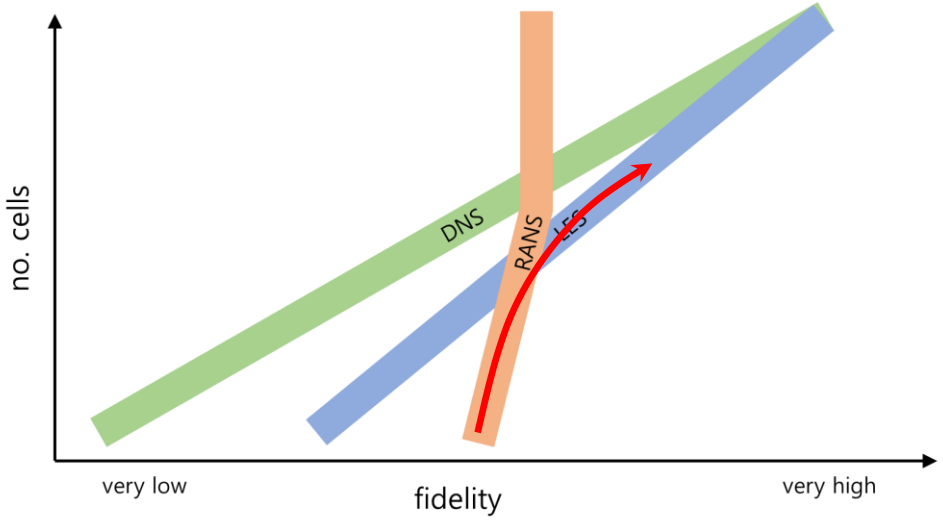


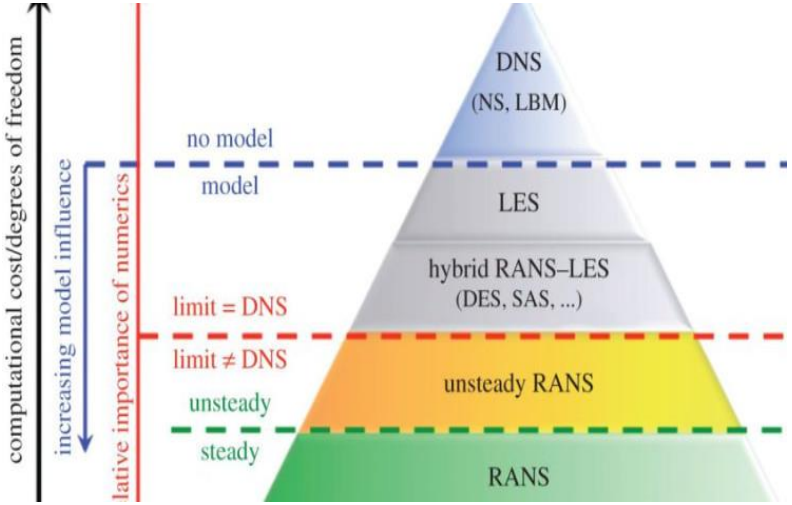
✓ **RANS를 이용한 CFD 해석 결과**

- 3천만개 이상 격자를 사용하더라도 오차는 더 이상 감소하지 않고…
- LES를 할 수 있는 수준(10억개 이상)은 안되고…
- 격자 조밀도 외에 해석의 정도를 개선할 수 있는 방법은 무엇일까???

OpenFOAM을 이용한 CFD

✓ **전산유체해석의 개선 방향 – 차분모델, 난류모델 등등**





✓ **난류 모델의 수정/개선**

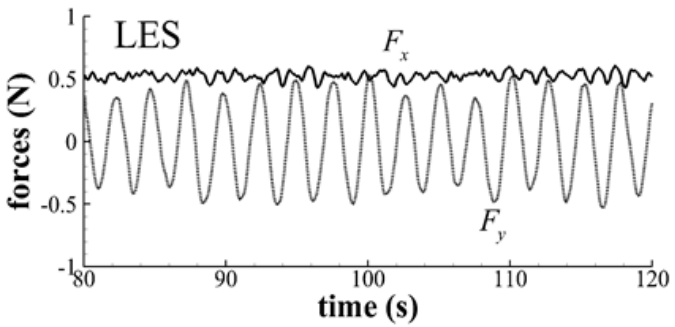
- **수치적 안정성 미검증**
- **격자 의존성 발생에 따른 해석 결과의 신뢰도 하락 우려**
- **해석 문제 특성에 따른 수치 기법 의존성 우려**

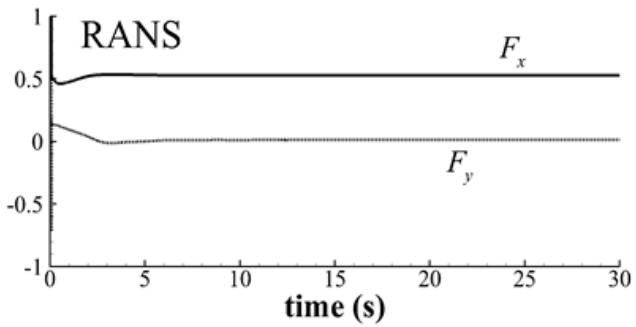
→ 상용 S/W에서 기술 지원하지 않는 영역

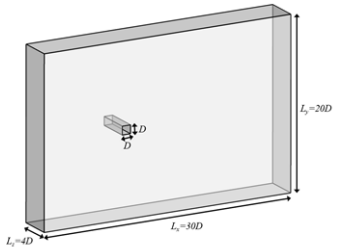
(문제 특화형으로 개발은 가능하나, 개발비용 발생 및 기술 유출 우려 가능성)

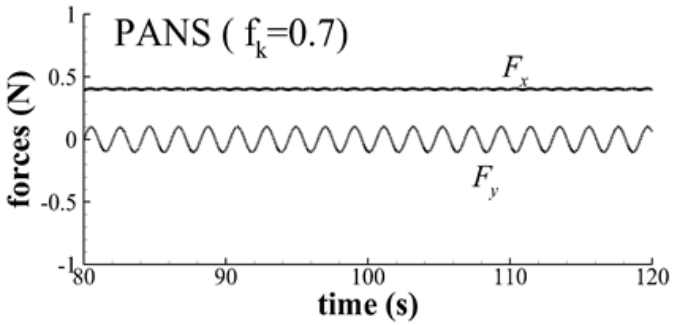
OpenFOAM의 개발 사례

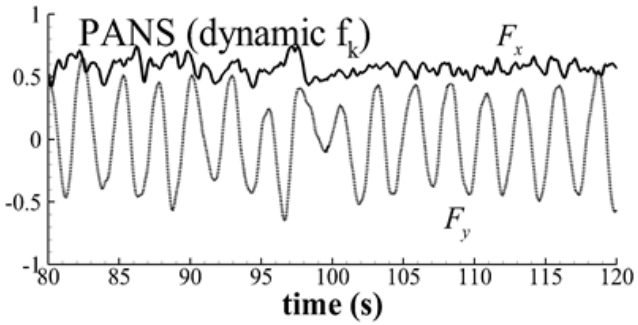
- ✓ **새로운 난류 모델에 대한 평가**
  - PANS (partially averaged Navier-Stokes) 난류 모델 평가 및 개선
  - RANS와 DNS의 혼합 모델 (이론상, 조밀 격자에서 TM→0으로 수렴 가능)







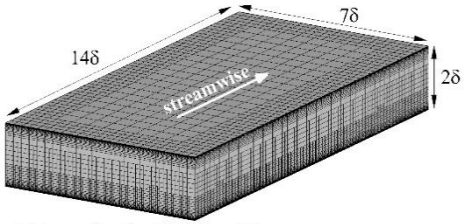




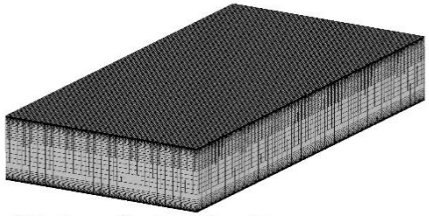
DONG-A UNIVERSITY
조선해양플랜트공학과

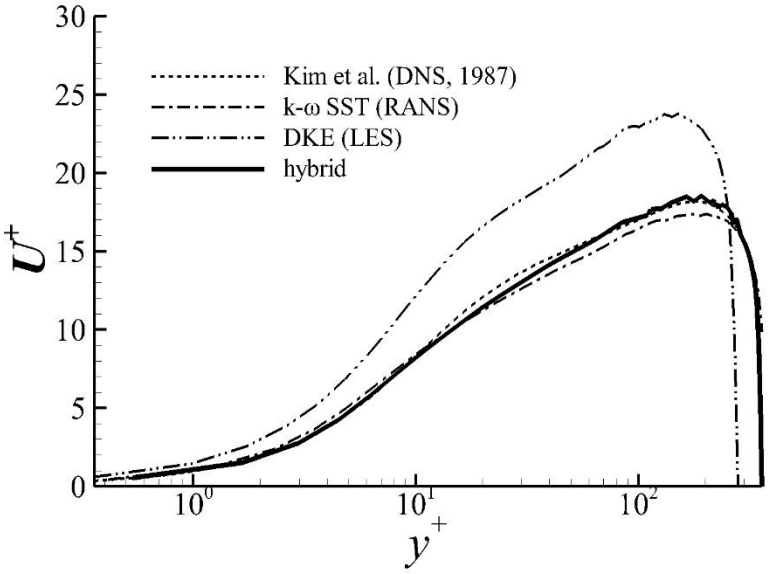
OpenFOAM의 개발 사례

- ✓ **Hybrid 난류 모델 라이브러리 개발**
  - **다종 난류 모델 사용이 가능한 OpenFOAM 라이브러리**
  - **LES 수준의 격자에서는 LES 해석, RANS 수준의 격자에서는 RANS 해석**



(a) top wall using RANS model


(b) bottom wall using LES model



OpenFOAM의 개발 사례

✓ **혼종 시간 차분 라이브러리 개발**

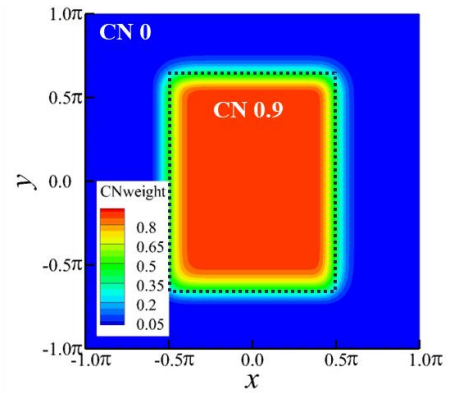
- **수치 안정화가 필요한 영역에서는 1차 Euler 기법**
- **수치 정확도가 필요한 영역에서는 2차 Crank Nicolson 기법**
- **irregular wave 해석 영역에서는 2차 C-N, 해양구조물 영역에서는 1차 Euler 적용 목적**

$$0 \le x, y \le 2\pi$$

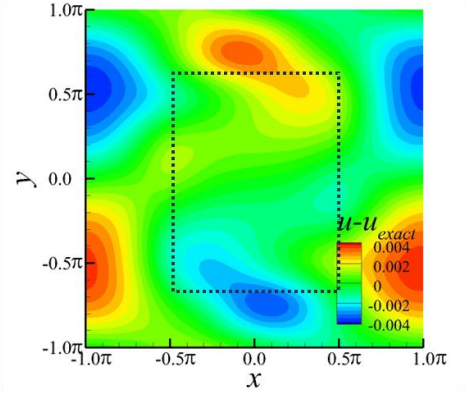
$$u = \cos(x)\sin(y)\,e^{-2\nu t}$$

$$v = -\sin(x)\cos(y)\,e^{-2\nu t}$$

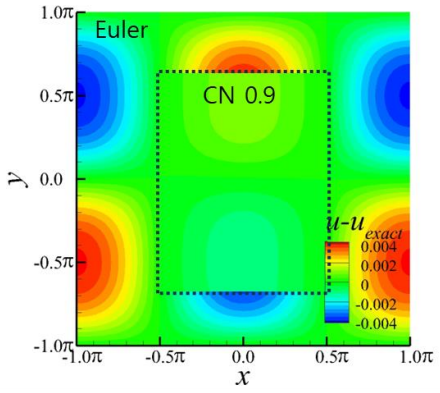
$$p = -\frac{\rho}{4}\{\cos(2x) + \cos(2y)\}e^{-4\nu t}$$



Δt=0.1 for hybridCN (100x100)
CN weight distribution (0~0.9)
smoother 10



Δt=0.1 for hybridCN (100x100)
error distribution

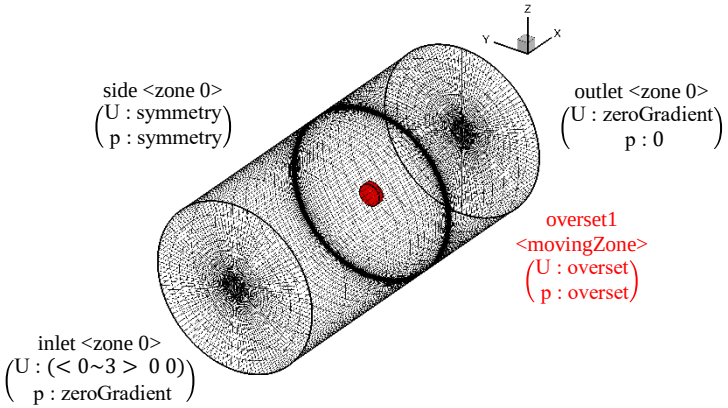


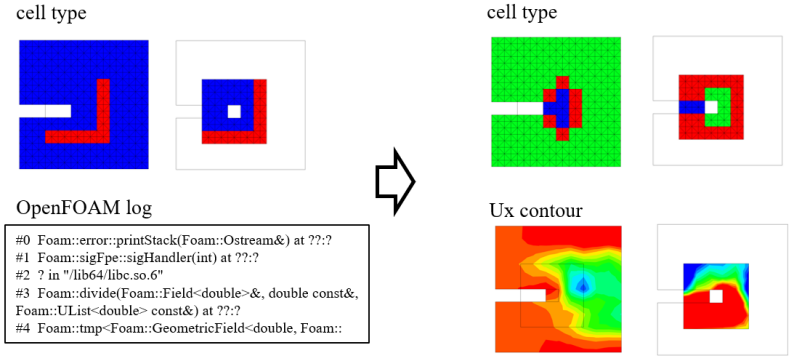
combination of error images

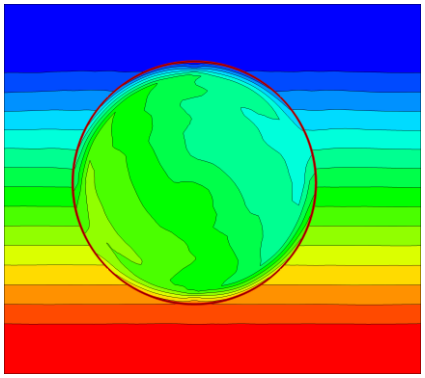
DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM의 개발 사례

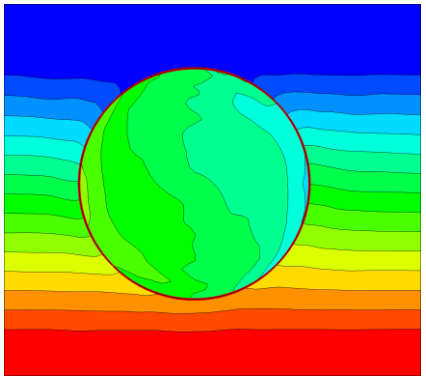
- ✓ **중첩 격자를 이용한 자항 해석**
  - **OpenFOAM의 중첩 격자 코드 버전 변환**
  - **중첩 격자의 문제 파악 및 개선 방향 수립**
  - **OpenFOAM 중첩 격자 라이브러리 개선 및 Suggar++ 라이브러리 비교**



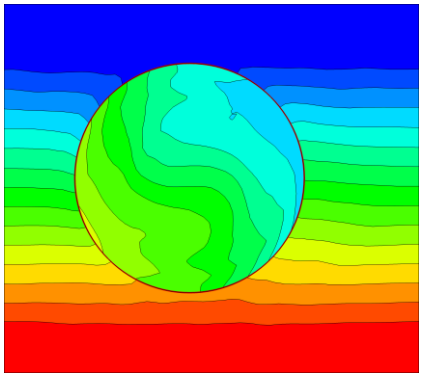




overset



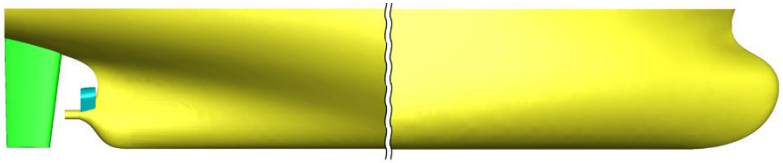
cyclicAMI



NCC

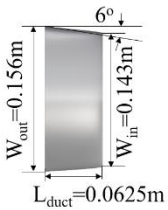
OpenFOAM 기반 캐비테이션 해석

- ✓ **캐비테이션 및 침식 예측 고도화**
  - STAR-CCM+를 이용한 캐비테이션 해석 기법 정립
  - 선미 부가물의 후류에서 발생하는 캐비테이션 예측 오차 발생
  - 캐비테이션 모델 및 수치 인자 최적화에도 불구하고 예측 한계
  - 캐비테이션에서 난류 모델 영향 지배적이라는 기존 연구 사례 (LES 추천)
  - 하이브리드 모델 기반 캐비테이션 해석 진행

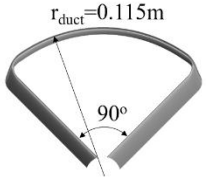


(a) sketch of KVLCC2 with a pre-swirl duct

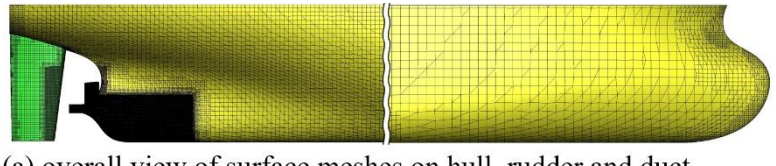
(b) 3D view



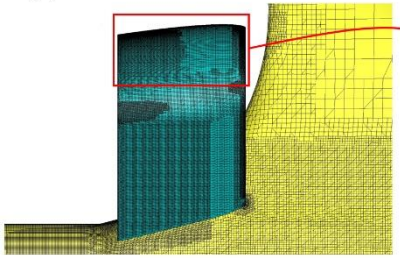
(c) top view



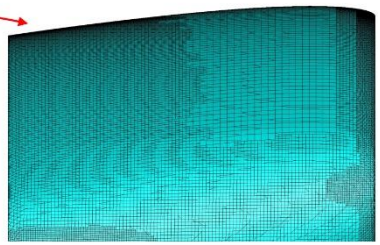
(d) front view



(a) overall view of surface meshes on hull, rudder and duct



(b) surface mesh near the duct



(c) surface mesh on the duct

DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM 기반 캐비테이션 해석

✓ 캐비테이션 및 침식 예측 고도화

✓ 통합 난류 모델 적용

- RANS : k-ε, k-ω SST 모델의 난류 점성항 통합 해석
- LES : WALE, dynamic k-eqn. 아격자 모델의 난류 점성항 통합 해석
- PANS : 정적/동적 fk 기반의 난류 점성항 통합 해석
- 난류 점성항 기반의 하이브리드 모델(RANS/LES/PANS)에 대한 OpenFOAM 소스 코드 개발

✓ 하이브리드 모델의 가중치 개발

- 경계면 기반 공간 거리 변수 개발
- 거리 변수에 따른 난류 점성항의 가중치 단위화
- 난류 점성항 하이브리드 모델에 대한 거리 기반 가중치 적용을 위한 소스 코드 개발

✓ 경계면 벽함수 및 난류 모델 입력 인터페이스 수정/개발

- 하이브리드 모델을 위한 벽함수 개발
- Turbulence properties의 난류 모델 입력 인터페이스 코드 수정/개발

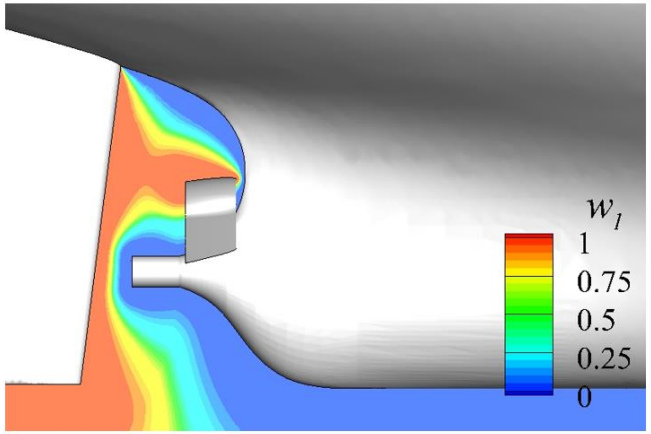
✓ 캐비테이션 모델 연계

- 하이브리드 모델에 대한 캐비테이션 모델 연계

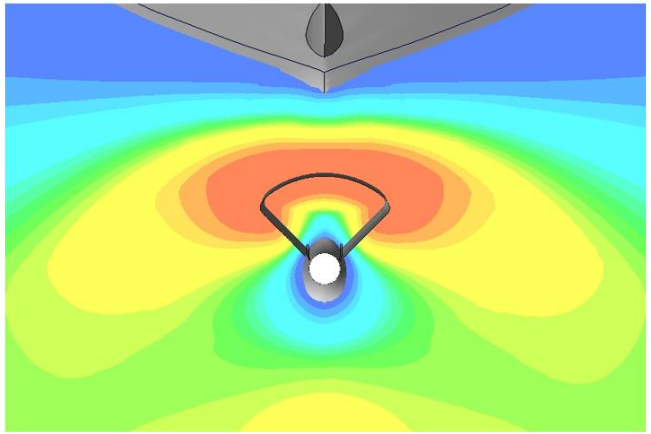
DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM 기반 캐비테이션 해석

✓ **캐비테이션 및 침식 예측 고도화**



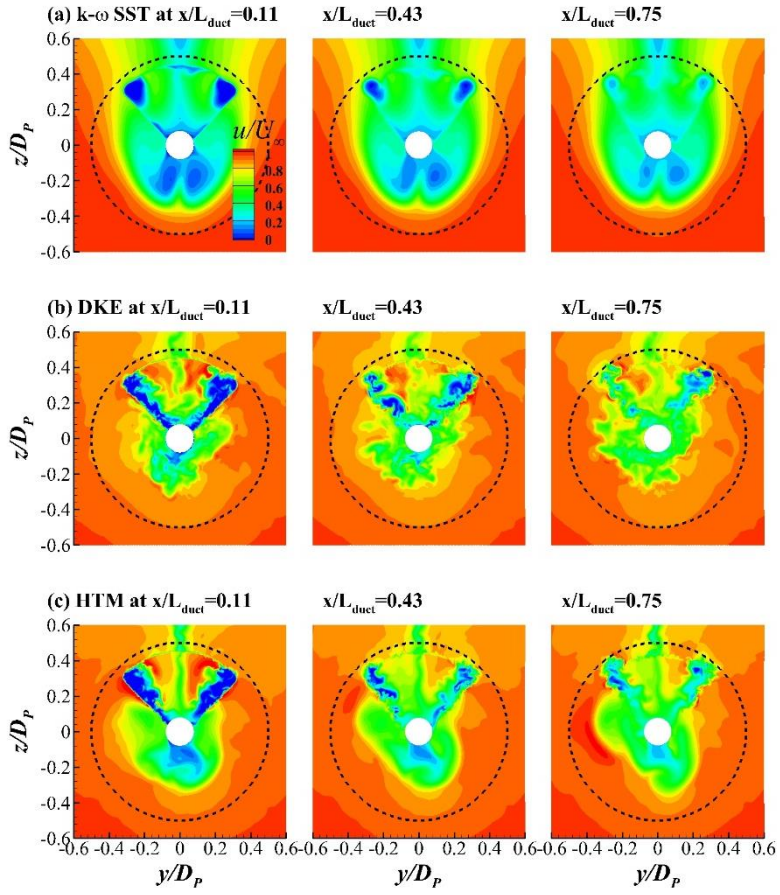
(a) side view of weighting coefficient for DKE model

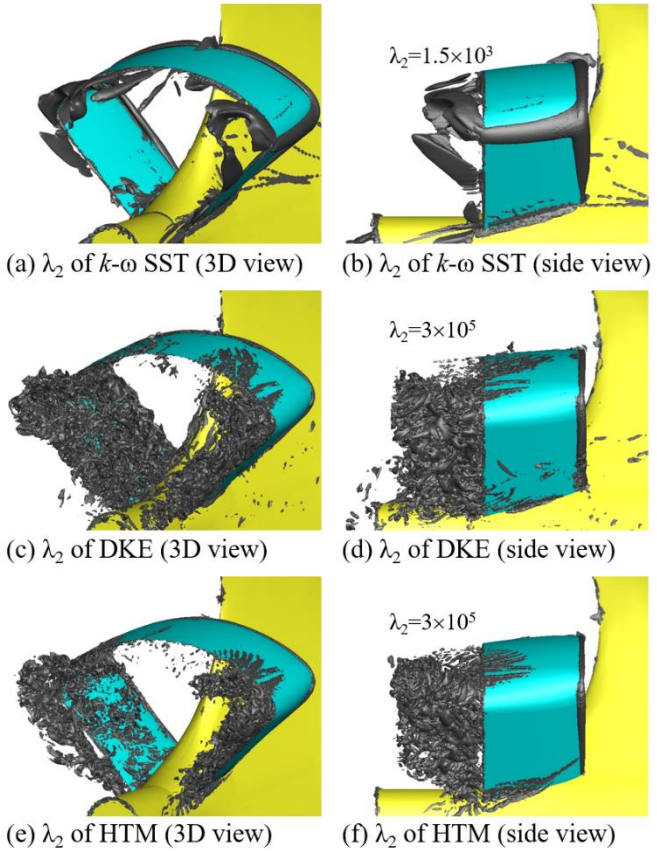

(b) rear view of weighting coefficient for DKE model

- hybridTurb 난류 모델의 RANS/LES 공간 가중치
  : 경계 조건 기반으로 생성

OpenFOAM 기반 캐비테이션 해석

✓ **캐비테이션 및 침식 예측 고도화**





(a) λ_2 of k-ω SST (3D view) (b) λ_2 of k-ω SST (side view)

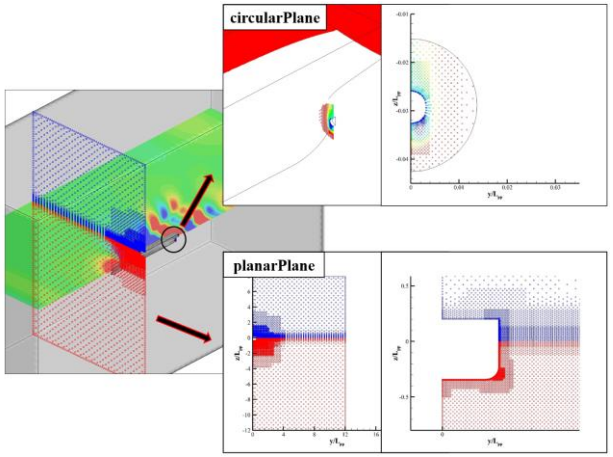
(c) λ_2 of DKE (3D view) (d) λ_2 of DKE (side view)

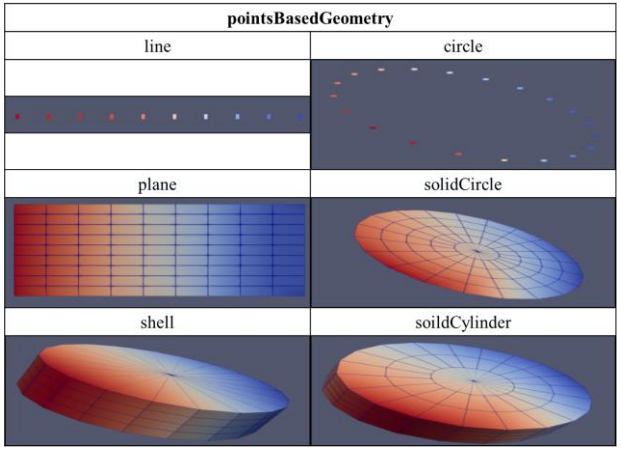
(e) λ_2 of HTM (3D view) (f) λ_2 of HTM (side view)

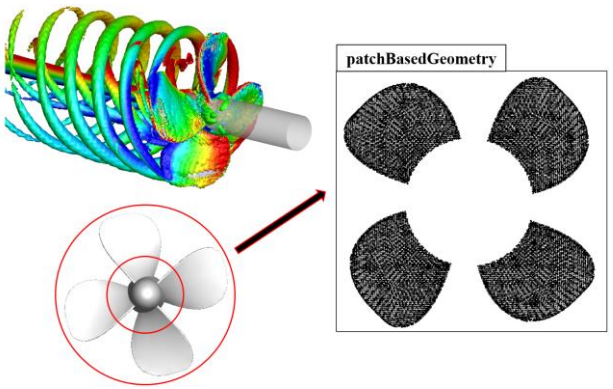
DONG-A UNIVERSITY
조선해양플랜트공학과

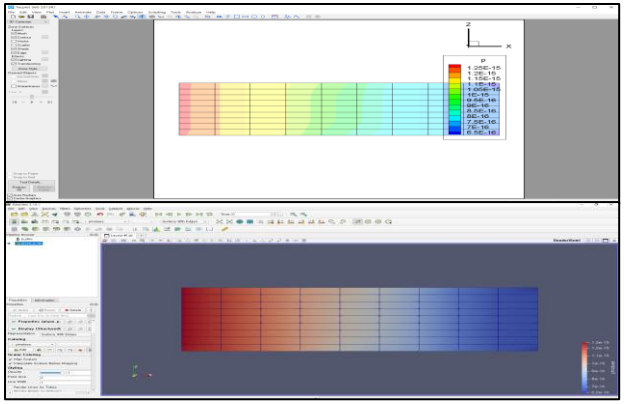
OpenFOAM 기반 캐비테이션 해석

✓ **캐비테이션 및 침식 예측 고도화 (후처리)**









DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM에 대한 생각

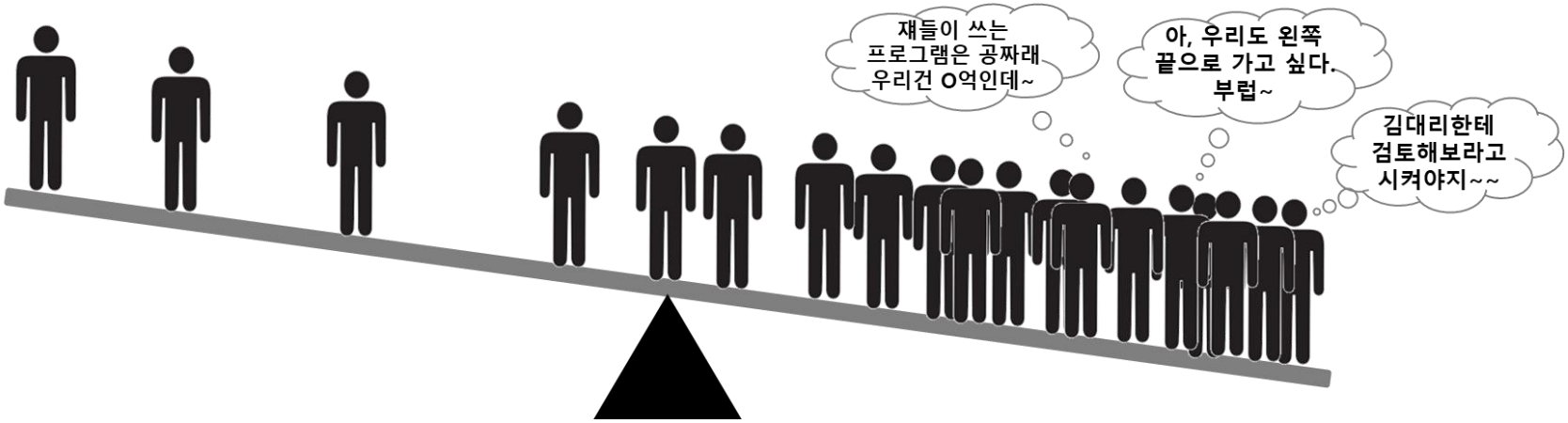
- ✓ **OpenFOAM을 시작하고 가장 많이 들은 질문**
  - **OpenFOAM 결과가 상용 S/W(Fluent, STAR-CCM+ 등)와 같나?**
  - **OpenFOAM이 상용 S/W 혹은 in-house 코드를 대체할 수 있나?**
  - **우리도 OpenFOAM 할 수 있나, 얼마나 하면 OpenFOAM을 잘 할 수 있나?**

→ 신뢰도와 비용 관점 (기술과 경제성의 관점)

→ CFD의 관점에 대한 이해

OpenFOAM의 기술성과 경제성

✓ **OpenFOAM에 대한 일반적인 생각 (장점)**



Open Source S/W
- 공짜다
- 소프트웨어사로부터 독립적이다
- 기밀 보안성이 높다

Commercial S/W
- 비싸다
- 소프트웨어사에 종속적이다
- 경쟁사에게 노출될 수 있다

OpenFOAM의 기술성과 경제성

✓ **OpenFOAM에 대한 일반적인 생각 (단점)**

Open Source S/W
- 개발자는 죽을 맛이다
- 기술 지원 받을 곳이 없다
- S급 개발자가 나가면 끝이다
- 시간/노력을 고려하면 오히려 비쌀 수도 있다

…

Commercial S/W
- 사용하기 쉽다
- 돈만 있으면 기술 지원은 얼마든지..
- 사용자에 대해 덜 의존적이다
- 정해진 가격에 구매할 수 있다

…

OpenFOAM의 특성 및 개발

- ✓ OpenFOAM에 대한 잠정적 결론

  1) 경제성 : OpenFOAM은 싸면서 비싸다

  - 라이선스 비용 무료
  - 개발 및 인력 비용 고려

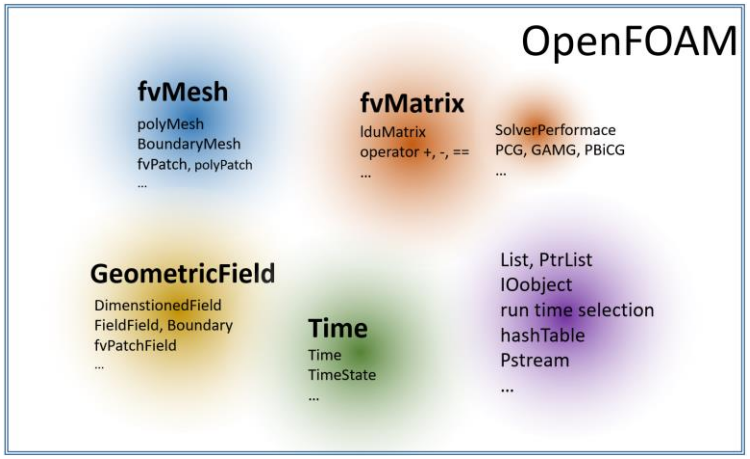
  2) 기술성 : OpenFOAM은 상용 S/W의 대체재이면서 보완재이다

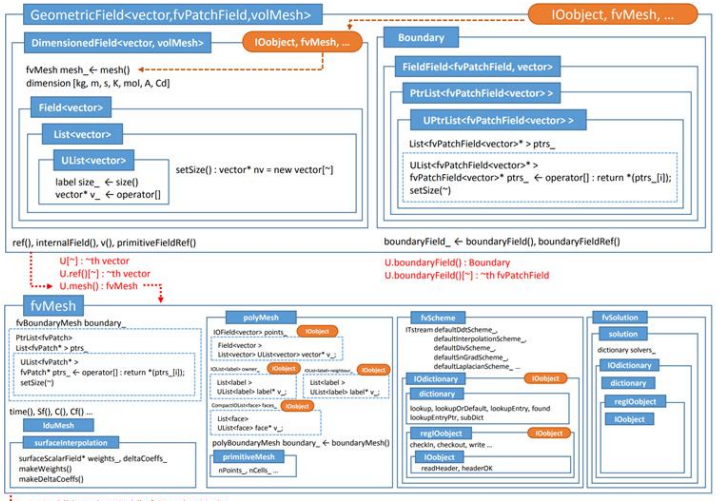
  - 상용 S/W의 높은 편의성, 낮은 개방성/확장성
  - OpenFOAM의 높은 개방성/확장성, 낮은 편의성

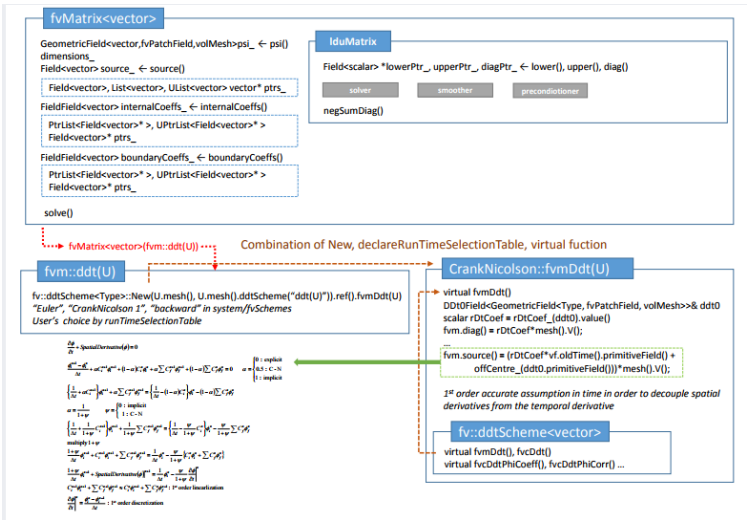
- ✓ OpenFOAM으로 하는 코드 개발은 어떻게 이루어지나?

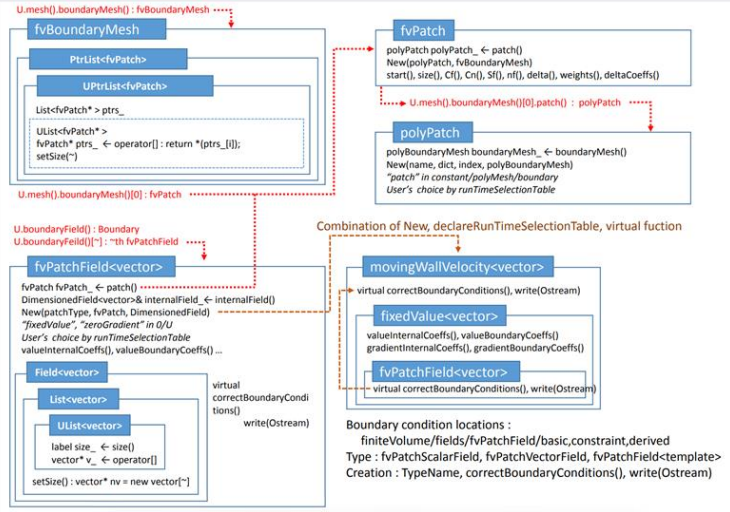
OpenFOAM 코드 개발 과정

✓ OpenFOAM 코드 구조에 대한 이해







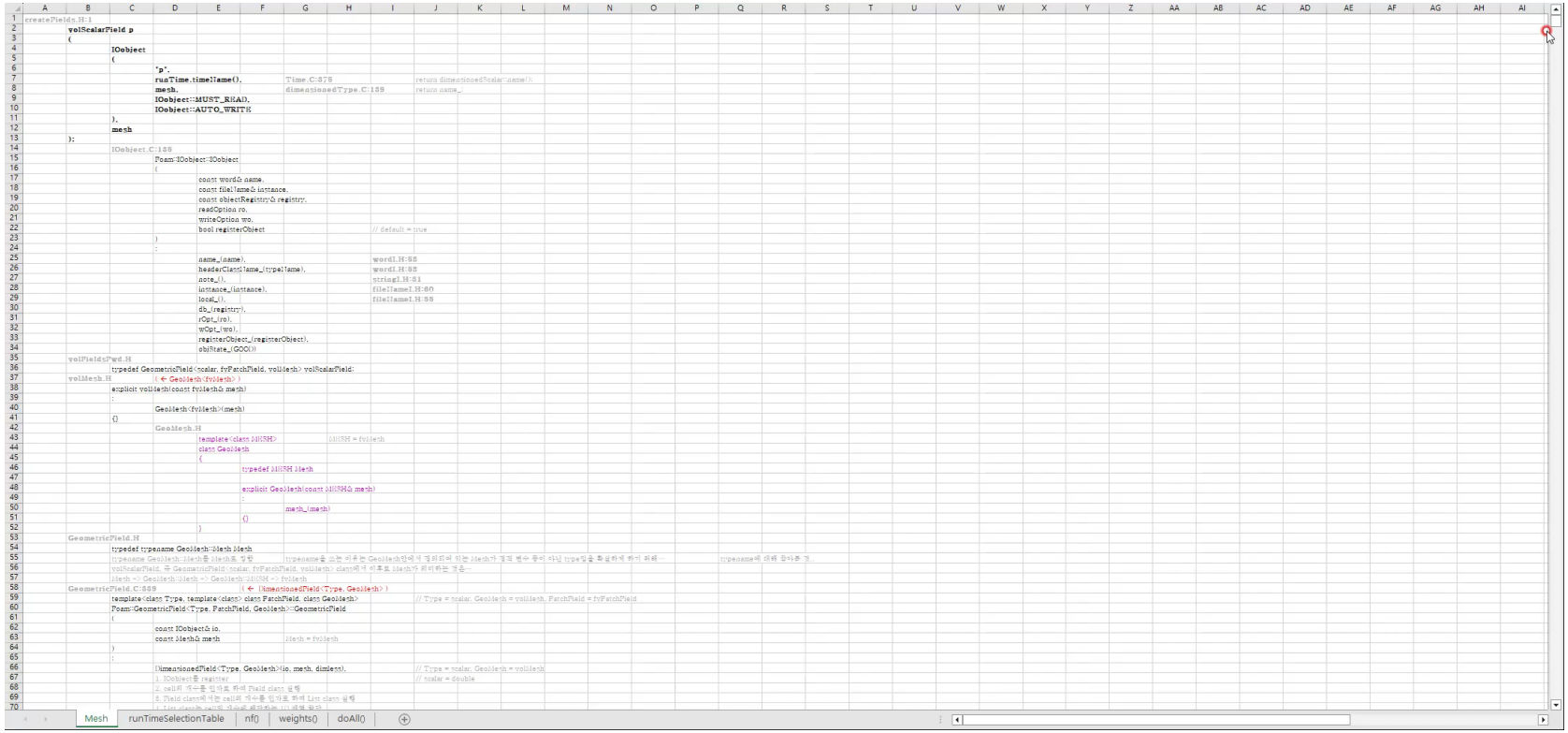


DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM 코드 개발 과정

✓ **코드 구조에 대한 이해는 어떻게?**

- 소스 코드를 한줄 한줄 읽으면서 (2010년 엑셀 파일 정리 예시)

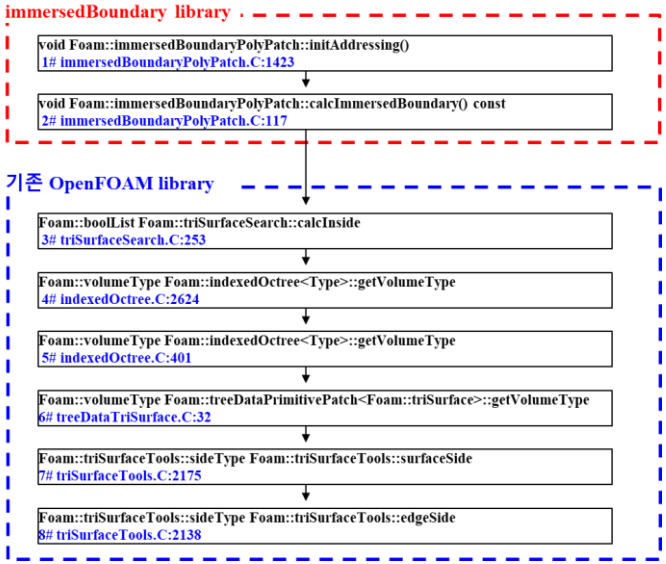


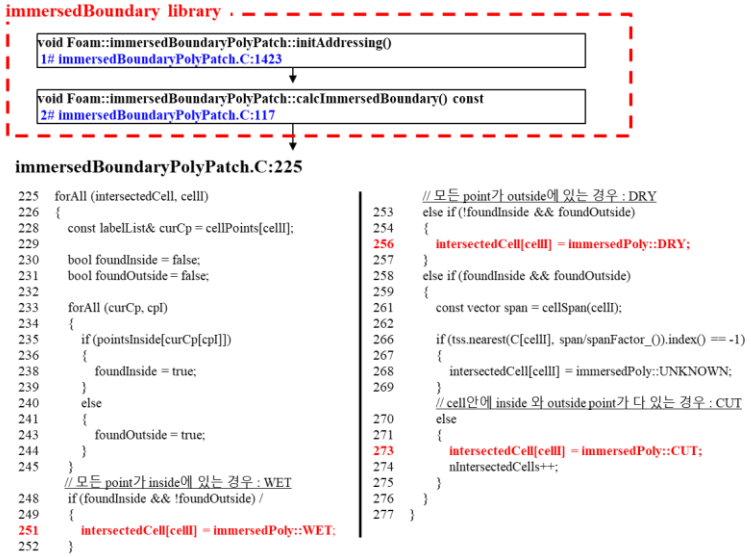
- 2012년 이후부터는 gdb 기반의 소스 코드 분석

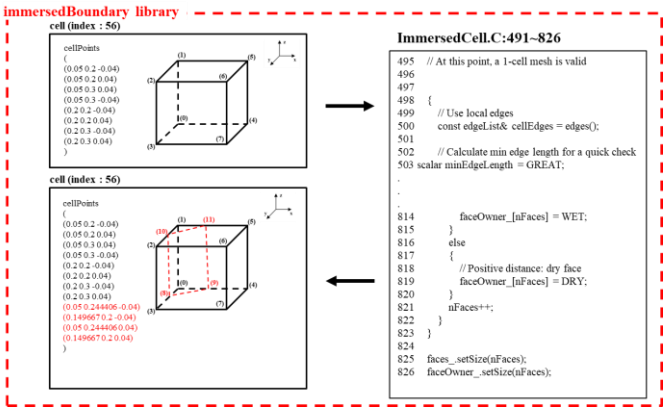
DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM 코드 개발 과정

✓ 코드 구조 연계 분석 (gdb)





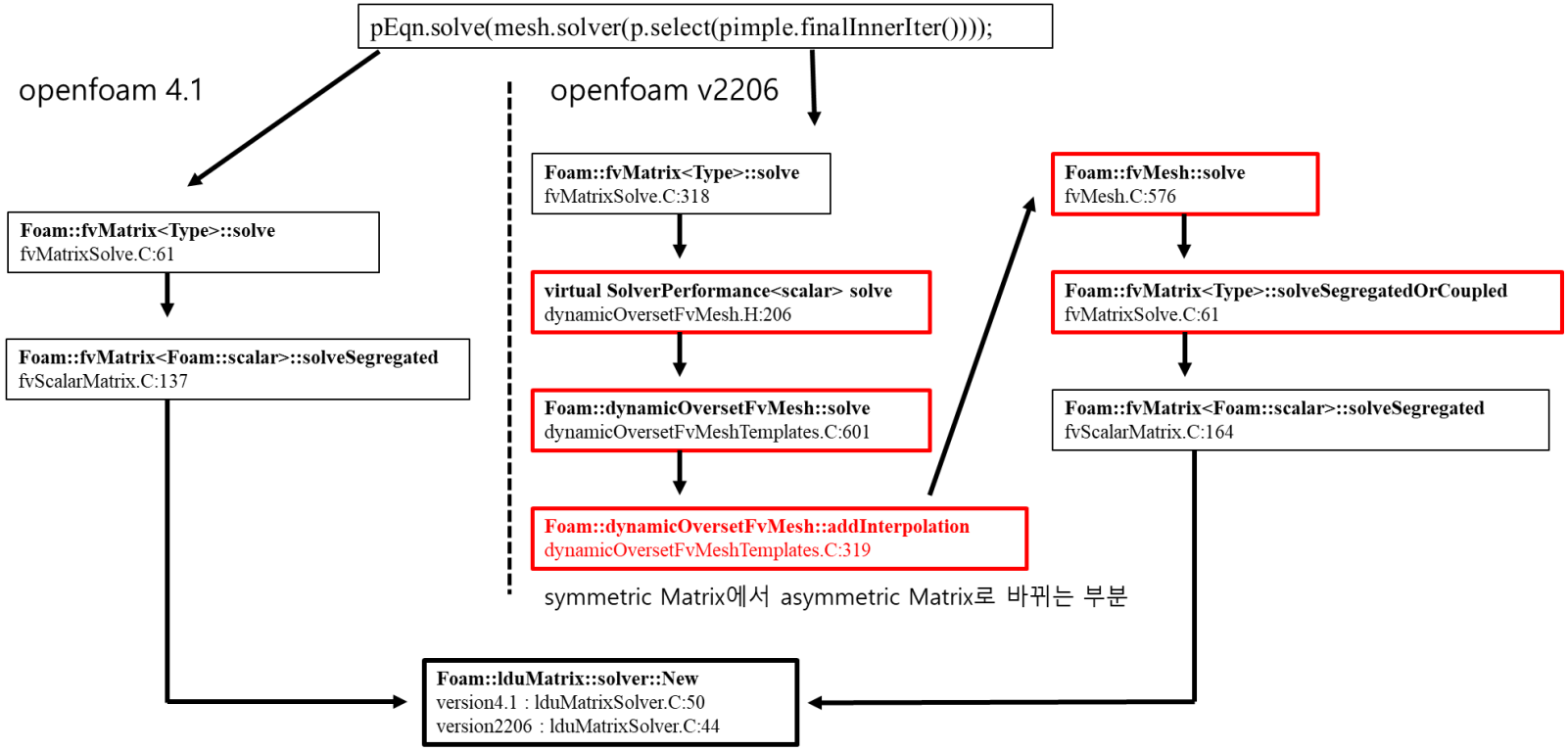


- OpenFOAM 버전 변환
- 기능에 대한 분석 및 개선 여부 가능성 검토
- 신규 개발 코드의 연계 포인트 결정
- 신규 개발 시 상속 클래스의 레벨 결정

DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM 코드 개발 과정

✓ OpenFOAM 버전의 기능 비교 과정 예시



- OpenFOAM v2206의 overset library에 대한 OF-4.1 혹은 OF-7로의 이식 가능성 검토
- v2206에서 복사할 클래스와 OF-4.1에서 수정할 클래스를 결정

OpenFOAM 코드 개발 과정

✓ OpenFOAM 기능 분석의 예시

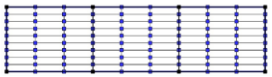
✓ **make Points & find Cell**

system/ControlDict

```
97          planeDict
98          {
99            firstPoint (-1 0 0);
100           secondPoint (-1 0 0.5);
101           thirdPoint (1 0 0.5);
102           numberOfEdgePoints 10;
103         }
```

Equation

```
538  pt[i][j] = secP + v1*i/(nP-1) + v2*j/(nP-1);
539  c[i][j] = mesh().findCell(pt[i][j]);
```



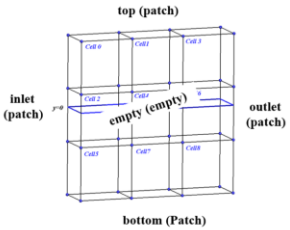
interpolate
interpolate scheme

pointsBasedGeometryTemplates.C : 292

```
278  autoPtr<interpolation<Type>> interpolators
279  (
280    interpolation<Type>::New("cellPoint", volFld)
281  );
282  List<List<Type> > gatherValues(Pstream::nProcs());
283  List<List<scalar> > gatherProcI(Pstream::nProcs());
284  List<Type> Values;
285  List<scalar> procI;
286  List<Type> sortValues(makeC_.size());
287  List<vector> sortPt(makeC_.size());
288  forAll(makeC_, i)
289  {
290    if(makeC_[i][0] == Pstream::myProcNo())
291    {
292      Values.append(interpolators().interpolate(makePt_[i], makeC_[i][1], -1));
293      procI.append(i);
294    }
295    if(makeC_[i][0] == -1)
296    {
297      WarningInFunction
298      << "When pointsBasedGeometryType is " << pointsBasedGeometryType_.typeName()
<< endl
299      << "    This point(" << makePt_[i] << ") is out of boundary." << endl;
300      break;
301    }
302  }
```

✓ controlDict 에 정의된 조건에 해당하는 point 생성
✓ point가 속해있는 cell을 검색 후 cell index 저장
✓ point, cell index, face index (default = -1) 및 interpolation Scheme 활용

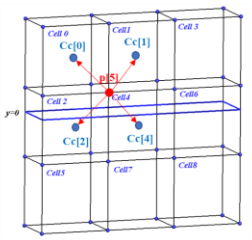
✓ **interpolation Scheme (cellPoint)**



✓ point를 공유하는 모든 face의 boundary Type 을 check

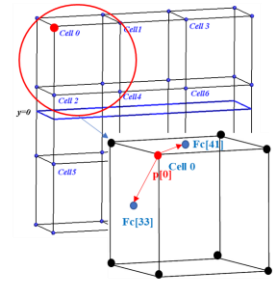
모두 empty 인 경우 : internal
아닌 경우 : boundary

makeInternalWeights()



weight_[0]= dist_[0] (p[5]-Cc[0]) / sumDist
weight_[1]= dist_[1] (p[5]-Cc[1]) / sumDist
weight_[2]= dist_[2] (p[5]-Cc[2]) / sumDist
weight_[3]= dist_[3] (p[5]-Cc[4]) / sumDist

makeboundaryWeights()



weight_[0]= dist_[0] (p[0]-Fc[33]) / sumDist
weight_[1]= dist_[1] (p[0]-Fc[41]) / sumDist

```
275 inline Foam::scalar Foam::tetrahedron<Point, PointRef>::pointToBarycentric
276 (
277   const point& pt,
278   barycentric& bary
279 ) const
280 {
284   vector e0(a_ - d_);
285   vector e1(b_ - d_);
286   vector e2(c_ - d_);
288   tensor t
289   (
290     e0.x(), e1.x(), e2.x(),
291     e0.y(), e1.y(), e2.y(),
292     e0.z(), e1.z(), e2.z()
293   );
294
295   scalar detT = det(t);
296
297   if (Foam::mag(detT) < small)
298   {
299     // Degenerate tetrahedron, returning 1/4 barycentric coordinates
300
301     bary = barycentric(0.25, 0.25, 0.25, 0.25);
302
303     return detT;
304   }
305
306   vector res = inv(t, detT) & (pt - d_);
307
308   bary[0] = res.x();
309   bary[1] = res.y();
310   bary[2] = res.z();
311   bary[3] = 1 - cmptSum(res);
312
313   return detT;
314 }
```

$$\begin{pmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{pmatrix} = \mathbf{T}^{-1}(\mathbf{r} - \mathbf{r}_4)$$

where $\mathbf{T}$ is now a 3×3 matrix:

$$\mathbf{T} = \begin{pmatrix} x_1 - x_4 & x_2 - x_4 & x_3 - x_4 \\ y_1 - y_4 & y_2 - y_4 & y_3 - y_4 \\ z_1 - z_4 & z_2 - z_4 & z_3 - z_4 \end{pmatrix}$$

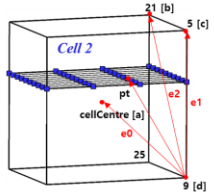
and $\lambda_4 = 1 - \lambda_1 - \lambda_2 - \lambda_3$ with the corresponding Cartesian coordinates:

$$\begin{aligned} x &= \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + (1 - \lambda_1 - \lambda_2 - \lambda_3) x_4 \\ y &= \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 + (1 - \lambda_1 - \lambda_2 - \lambda_3) y_4 \\ z &= \lambda_1 z_1 + \lambda_2 z_2 + \lambda_3 z_3 + (1 - \lambda_1 - \lambda_2 - \lambda_3) z_4 \end{aligned}$$

$$T\begin{pmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{pmatrix} = (d - pt)$$

$$\begin{pmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{pmatrix} = T^{-1}(d - pt)$$

$$\begin{pmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{pmatrix} = \frac{adj\ (T)}{\det(T)}(d - pt)$$

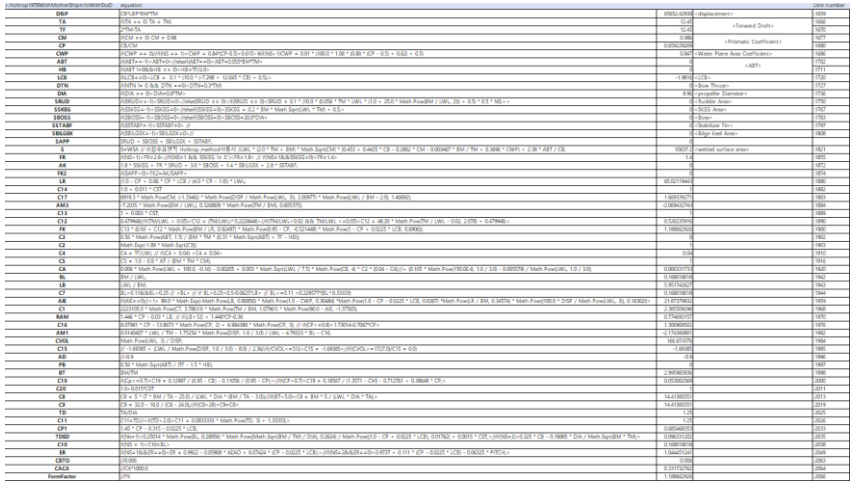


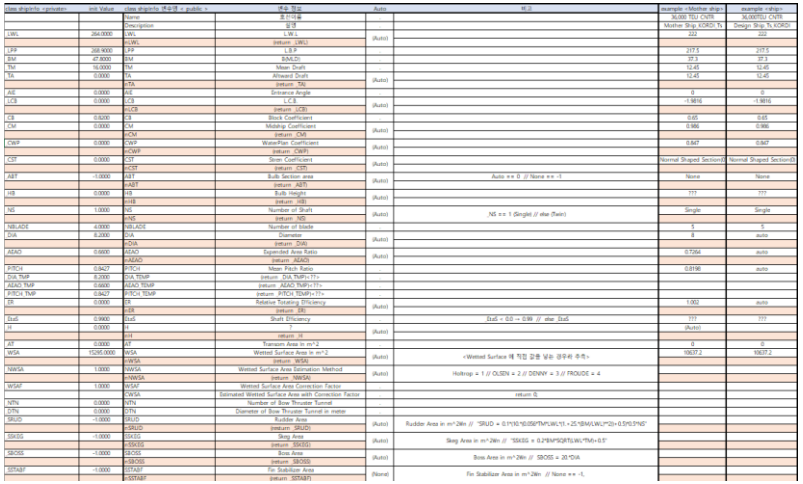
- 값에 대한 확인
- 수학적 타당성에 대한 검토
- 최선의 기능 및 옵션 선택

DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM 코드 개발 과정

✓ 개발/수정 라이브러리의 구조 설계





- 연계 포인트 설계 이후 상속 클래스 결정
- class private 변수 설정
- constructor와 member function 설정 (기능에 따른 const 결정)

DONG-A UNIVERSITY
조선해양플랜트공학과

OpenFOAM 코드 개발 과정

✓ 코드 개발 혹은 변경 레포트 작성

Before — dynamicPolyRefinementFvMesh.C:239

```
239   if
240   (
241       timeID > 0
242     && curTimeIndex_ < timeID
243     && (performRefinement || performUnrefinement)
244   )
245   {
246       // Update current time index to skip multiple topo changes per single
247       // time step
248       curTimeIndex_ = time().timeIndex();
.
.
.
.
.
337       if (sizeCellMap)
338       {
339           Info<< "Successfully performed polyhedral refinement. "
340               << "Changed from " << nOldCells << " to " << sizeCellMap
341               << " cells." << endl;
342       }
343       else
344       {
345           Info<< "Refinement/unrefinement not performed in this time step "
346               << "since no cells were selected." << endl;
```

After — dynamicPolyRefinementFvMesh.C

```
244   if
245   (
246       timeID > 0
247     && curTimeIndex_ < timeID
248     && (performRefinement || performUnrefinement)
249   )
250   {
251 /*
252       // Update current time index to skip multiple topo changes per single
253       // time step
254       curTimeIndex_ = time().timeIndex();
345       if (sizeCellMap)
346       {
347           Info<< "Successfully performed polyhedral refinement. "
348               << "Changed from " << nOldCells << " to " << sizeCellMap
349               << " cells." << endl;
350       }
351       else
352       {
353           Info<< "Refinement/unrefinement not performed in this time step "
354               << "since no cells were selected." << endl;
355       }
356 */ // by SBLEE
```

Before — writeIbMasks.C:160

```
160   forAll (mesh.boundary(), patchI)
161   {
162       if (!isA<immersedBoundaryFvPatch>(mesh.boundary()[patchI]))
163       {
164           sGamma.boundaryField()[patchI] =
165               magSf.boundaryField()[patchI]/
166               mesh.boundary()[patchI].patchSlice(magRawFaceAreas);
167
168           gamma.boundaryField()[patchI] =
169               sGamma.boundaryField()[patchI];
170       }
171       else
172       {
173           sGamma.boundaryField()[patchI] = 1;
174           gamma.boundaryField()[patchI] = 1;
175       }
176   }
```

After — writeIbMasks.

```
168   forAll (mesh.boundary(), patchI)
169   {
170       if (!isA<immersedBoundaryFvPatch>(mesh.boundary()[patchI]))
171       {
172           //sGamma.boundaryField()[patchI] = // by SBLEE
173           sGamma.boundaryFieldRef()[patchI] =
174               magSf.boundaryField()[patchI]/
175               mesh.boundary()[patchI].patchSlice(magRawFaceAreas);
176
177           // gamma.boundaryField()[patchI] = // by SBLEE
178           gamma.boundaryFieldRef()[patchI] =
179               sGamma.boundaryField()[patchI];
180       }
181       else
182       {
183           // sGamma.boundaryField()[patchI] = 1; // by SBLEE
184           sGamma.boundaryFieldRef()[patchI] = 1;
185           // gamma.boundaryField()[patchI] = 1; // by SBLEE
186           gamma.boundaryFieldRef()[patchI] = 1;
187       }
188   }
```

Before — immersedBoundaryOmegaWallFunctionFvPatchScalarField.C:145

```
145 immersedBoundaryOmegaWallFunctionFvPatchScalarField::
146 immersedBoundaryOmegaWallFunctionFvPatchScalarField
147 (
148     const immersedBoundaryOmegaWallFunctionFvPatchScalarField& ptf,
149     const DimensionedField<scalar, volMesh>& iF
150 )
151 :
152     omegaWallFunctionFvPatchScalarField(ptf, iF),
153     immersedBoundaryFieldBase<scalar>
154     (
155         ptf.ibPatch(),
156         ptf.setDeadValue(),
157         ptf.deadValue()
158     )
159 {
160     this->setPatchType(ptf);
161 }
```

After — immersedBoundaryOmegaWallFunctionFvPatchScalarField.C

```
165 immersedBoundaryOmegaWallFunctionFvPatchScalarField::
166 immersedBoundaryOmegaWallFunctionFvPatchScalarField
167 (
168     const immersedBoundaryOmegaWallFunctionFvPatchScalarField& ptf,
169     const DimensionedField<scalar, volMesh>& iF
170 )
171 :
172     omegaWallFunctionFvPatchScalarField(ptf, iF),
173     immersedBoundaryFieldBase<scalar>
174     (
175         ptf.ibPatch(),
176         ptf.setDeadValue(),
177         ptf.deadValue()
178     ),
179     refValue_(ptf.refValue_) // by SBLEE
180 {
181     // this->setPatchType(ptf); // by SBLEE
182     this->patchType()=ptf.patchType();
183 }
```

Before — immersedBoundaryPolyPatch.C:102

triSurfSearchPtr_의 변수에 surface의 정보가 저장되어있지 않음.

New triSurfaceSearch(triSurface type 변수) 가 필요함

이를 위해 triSurfaceMesh type인 ibMesh_ 변수에 (triSurface)를 씌워 해결하고자 하였지만 이 경우 surface의 정보가 triSurfSearchPtr_ 변수로 들어가지 않음

이 문제를 해결하기 위해 triSurface의 정보를 return 해주는 surface() 함수를 이용하여 해결

```
102     // It is an error to attempt to recalculate
103     // if the pointer is already
104     if (triSurfSearchPtr_)
105     {
106         FatalErrorInFunction
107             << "triSurface search algorithm already exist"
108             << abort(FatalError);
109     }
110
111     //triSurfSearchPtr_ = new triSurfaceSearch(ibMesh_);
112     triSurfSearchPtr_ = new triSurfaceSearch((triSurface)ibMesh_); // by SBLEE
113
114 }
```

After — immersedBoundaryPolyPatch.C:

```
102     // It is an error to attempt to recalculate
103     // if the pointer is already
104     if (triSurfSearchPtr_)
105     {
106         FatalErrorInFunction
107             << "triSurface search algorithm already exist"
108             << abort(FatalError);
109     }
110
111     //triSurfSearchPtr_ = new triSurfaceSearch(ibMesh_);
112     //triSurfSearchPtr_ = new triSurfaceSearch((triSurface)ibMesh_); // by SBLEE
113     triSurfSearchPtr_ = new triSurfaceSearch(ibMesh_.surface()); //by CWKIM
114
115 }
```


DONG-A UNIVERSITY
조선해양플랜트공학과

결론

- ✓ OpenFOAM 사용 기간 : 약 10년

- ✓ OpenFOAM에 대한 개발 느낌 : 레고 조립
  - 가능하면 있는 클래스와 기능을 조합해서…
  - 피치 못할 경우 새로운 블록을 제작해야 하나, 쫌…

- ✓ OpenFOAM의 버전 업데이트에 대한 느낌
  - 간혹 기본 플랫폼에 대한 변경 (상용 S/W의 구성 모방)
  - 핵심 엔진(안정성, 속도개선 등)에 대한 업데이트는 그다지…