OpenFOAM의 overset 라이브러리, suggar++

동아대학교

김찬우, 이상봉

INDEX 목차 目次

❖ 서론

➢ dynamic mesh

- 계산 중 격자가 변하도록 하는 기능
- 격자의 변화는 사전에 정해진 물체의 운동, 유체의 힘에 의한 운동 등 다양한 이유로 발생

➢ dynamic mesh 사용 예시

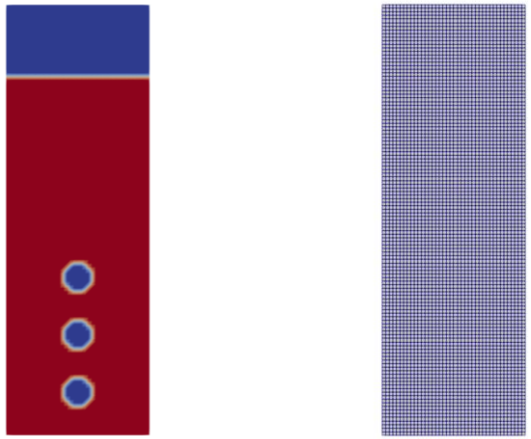
- 물체의 운동에 따른 전체 메시 변경
  - ✓ sloshing tank
- 물체의 운동 및 물체의 변화에 따른 특정 영역의 메시 변경
  - ✓ propeller의 회전 운동, 선박의 6자유도 운동, FSI
- 특정 기준에 따라 새로운 셀을 추가하거나 제거
  - ✓ 자유 수면 (VOF = 0.5)

❖ 서론

➢ dynamic mesh 종류 (1)

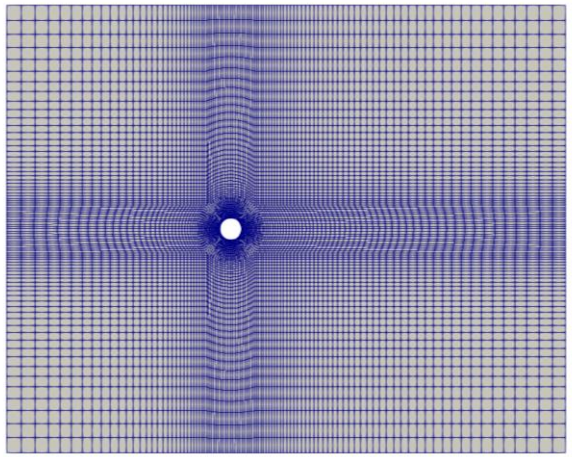
- adaptive mesh refinement
  - ✓ 특정 기준에 따라 새로운 셀을 추가하거나 제거하는 기법

Three rising bubbles (VOF)



- mesh morphing
  - ✓ 격자 점들을 이동시켜 물체의 움직임을 구현하는 기법

cylinder oscillation



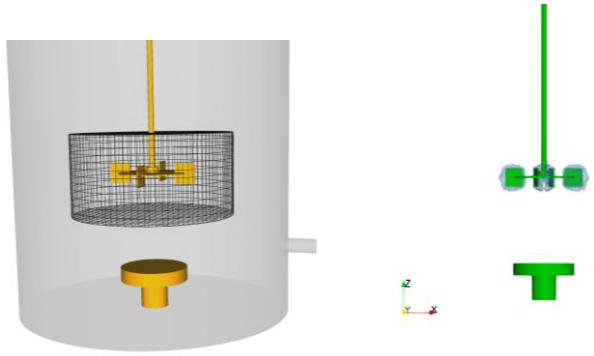
자료 출처 : OpenFOAM - Dynamic meshes tutorials (wolfdynamics.com)

❖ 서론

➢ dynamic mesh 종류 (2)

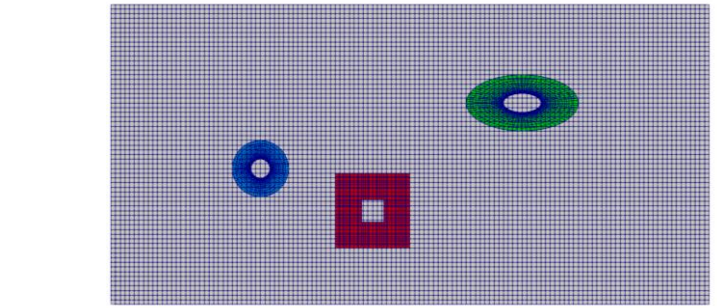
- sliding mesh
  - ✓ 격자계를 분리한 후 격자계의 움직임을 통해 움직임을 구현하는 기법
  - ✓ 서로 다른 격자계의 경계면이 맞닿아 있어야 가능한 기법

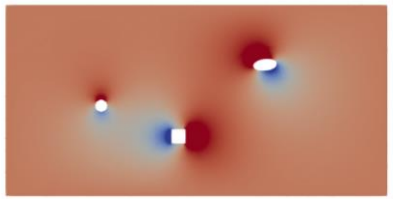
Continuous stirring tank reactor

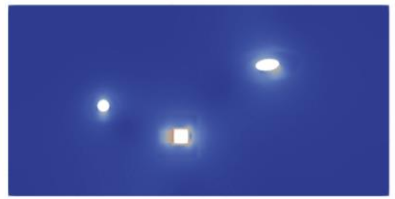


- overset mesh
  - ✓ 중첩된 격자계의 움직임을 통해 물체의 운동을 구현하는 기법

multiple bodies





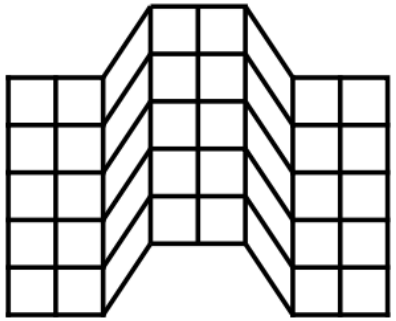


자료 출처 : OpenFOAM - Dynamic meshes tutorials (wolfdynamics.com)

❖ 서론

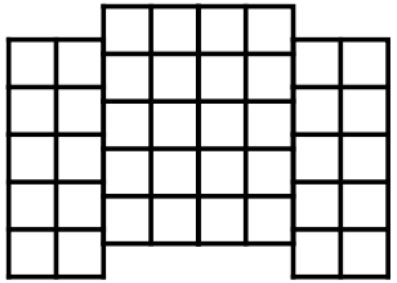
➢ 각 기법의 특징

- morphing



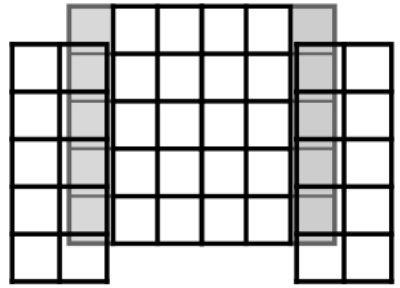
✓ 질량, 운동량 보존
✓ 격자 플럭스 계산 필요
✓ 대변위 운동 적용 불가
✓ 변위 방향 제한 없음

- sliding mesh



✓ 질량, 운동량 비보존
✓ 면적 기반 2차 정확도 내삽
✓ face-pairing 필수 (수치 불안정)
✓ face에 수직한 운동 모사 불가

- overset

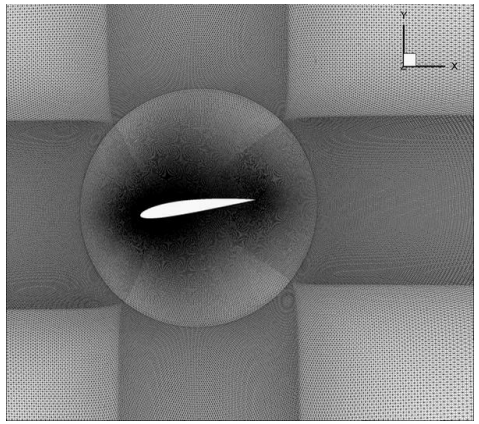


✓ 질량, 운동량 보존
✓ 거리 기반 관계식 도출
✓ cell-stencil 필수 (해석 시간)
✓ 변위 방향 제한 없음

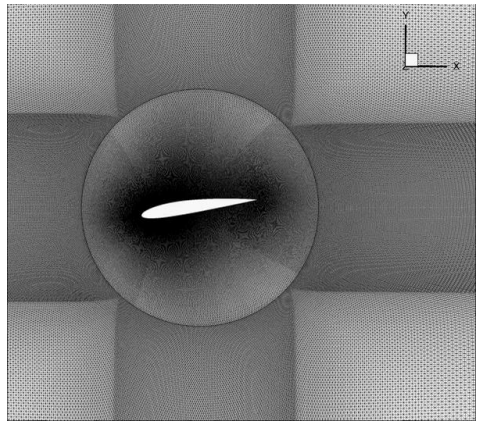
❖ 서론

➢ 각 기법의 특징

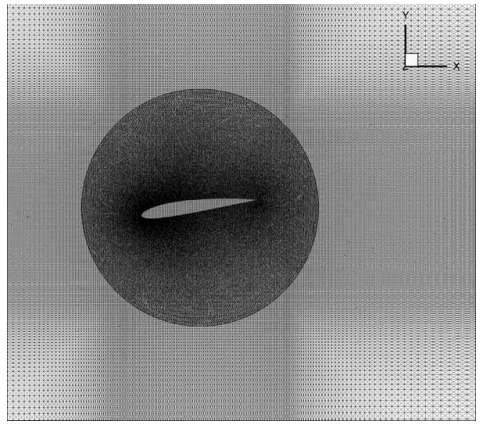
morphing



sliding mesh

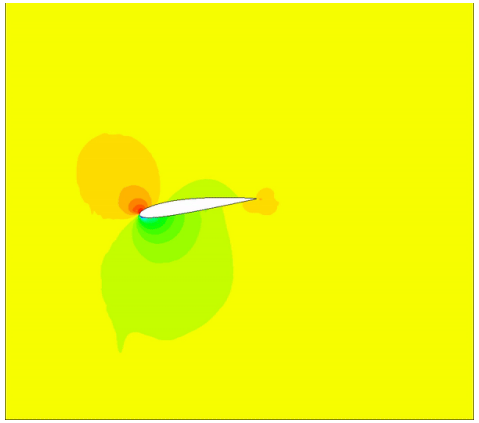


overset

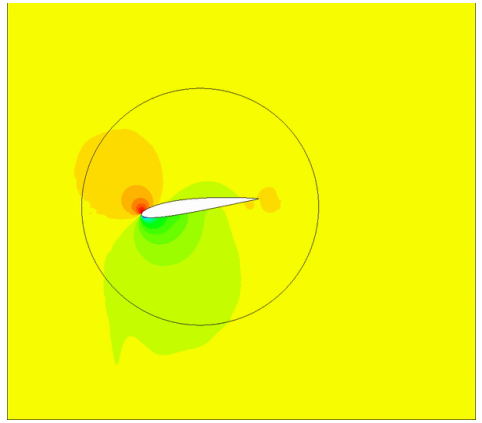


- pressure contour

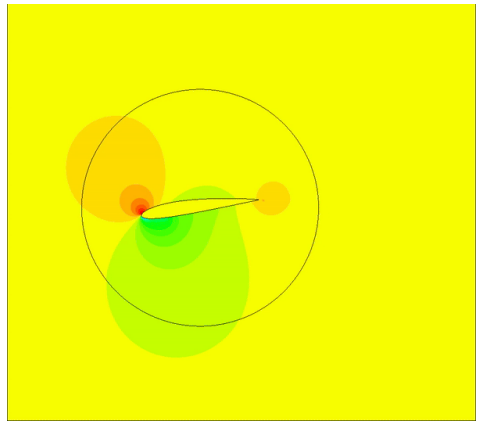
morphing



sliding mesh



overset



❖ 서론

➢ overset 기법 관련 연구

- Chandar, D. D., Boppana, B., & Kumar, V. (2018). A comparative study of different overset grid solvers between openfoam, starccm+ and ansys-fluent. In *2018 AIAA Aerospace Sciences Meeting* (p. 1564).
  - ✓ OpenFOAM, STAR-CCM+, ANSYS-Fluent에서 지원하는 overset 기법 결과 비교 및 분석
- Chan, W. M. (2009). Overset grid technology development at NASA Ames Research Center. *Computers & Fluids*, *38*(3), 496-503.
  - ✓ overset 기술의 발전 과정 및 사용 예시를 정리한 연구 (NASA Ames Research Center)

➢ OGS24 (overset grid symposium)

- 2년 마다 개최되며 overset 관련 연구들 공유하는 overset 심포지엄
- 2024 Sinclair center, Dayton, OH USA에서 개최
  - ✓ 2024, Scott Sherer & Daniel Garmann, Efficient Partitioning Strategy for Structured Overset Grids
  - ✓ 2024, Ralph Noack, Suggar++ Improvements and an Augmented Xray Hole Cutting Method

❖ 서론

➢ overset 관련 프로그램

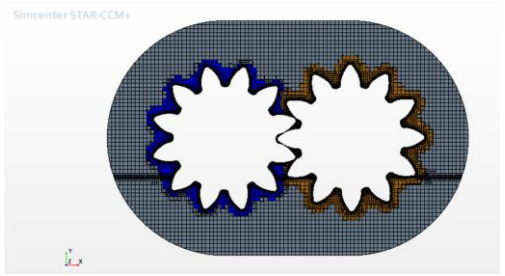
- OpenFOAM-v1706
- OpenFOAM-extend-4

} open source code

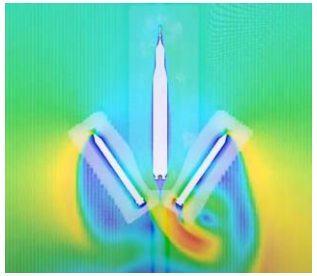
- STAR-CCM+
- Fluent
- Caelus

기타 프로그램을 이용한 overset 계산들

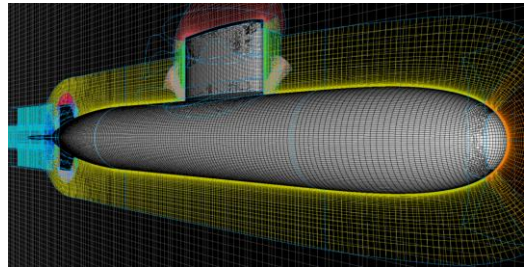
✓ STAR-CCM+



✓ Fluent



✓ Caelus



➢ overset 기법의 알고리즘 분석

❖ overset

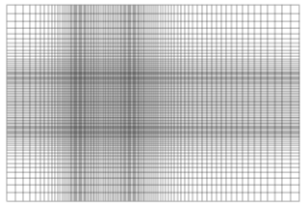
➢ overset 계산 알고리즘

격자 정보 전달 → 불필요한 cell 탐색 → 중첩된 cell 탐색 → cell type 선정 → matrix 수정

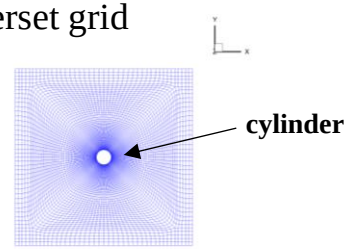
1) 불필요한 cell 탐색 : cell이 다른 격자계와 중첩되어 있는지를 판단
2) 중첩된 cell 탐색 : 어떤 cell과 중첩되어 있는지 탐색
3) cell type 선정 : cell의 위치 및 특성에 따라 cell type을 분류
4) matrix 수정 : 위 과정에서 얻은 결과를 바탕으로 matrix 수정

➢ overset 계산 예제 (2D cylinder)

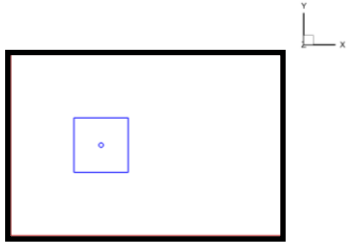
background grid



overset grid



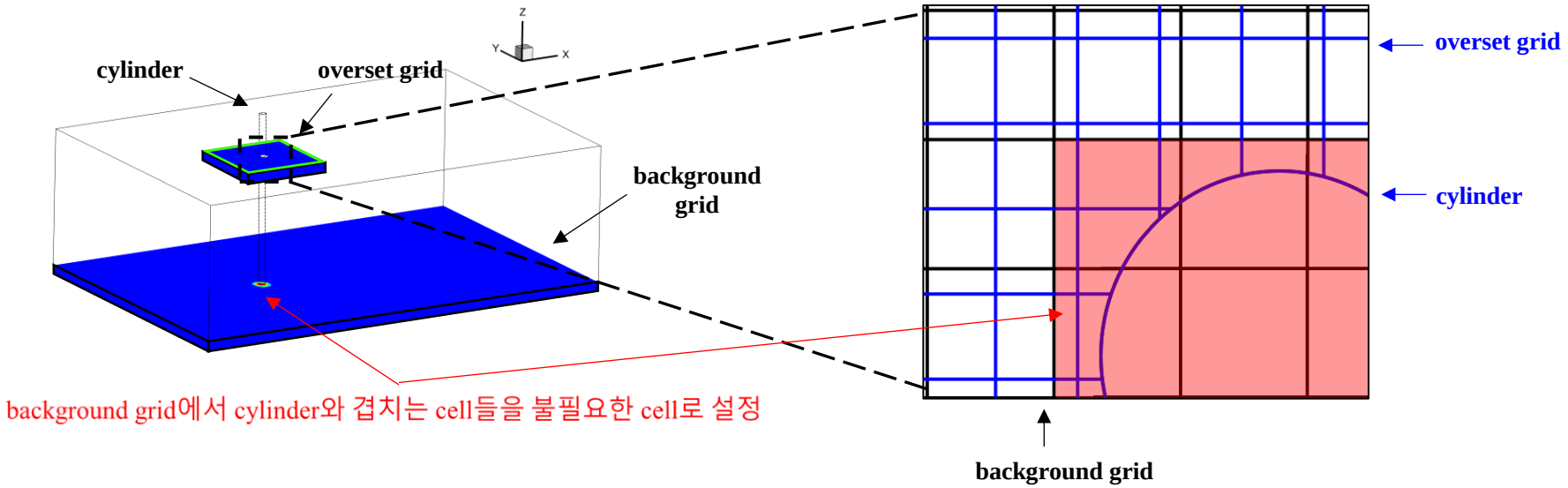
mergeMesh



❖ overset

➢ 불필요한 cell 탐색

- 계산에 불필요한 cell을 선정하는 단계이며 이 과정을 hole cutting이라고 함
- 일반적으로 중첩된 격자계에 중첩된 cell이 없는 경우 불필요한 cell로 설정
- hole cutting 방법에는 다양한 방법들이 존재 (ex. direct cutting, octree-based cutting, etc.)

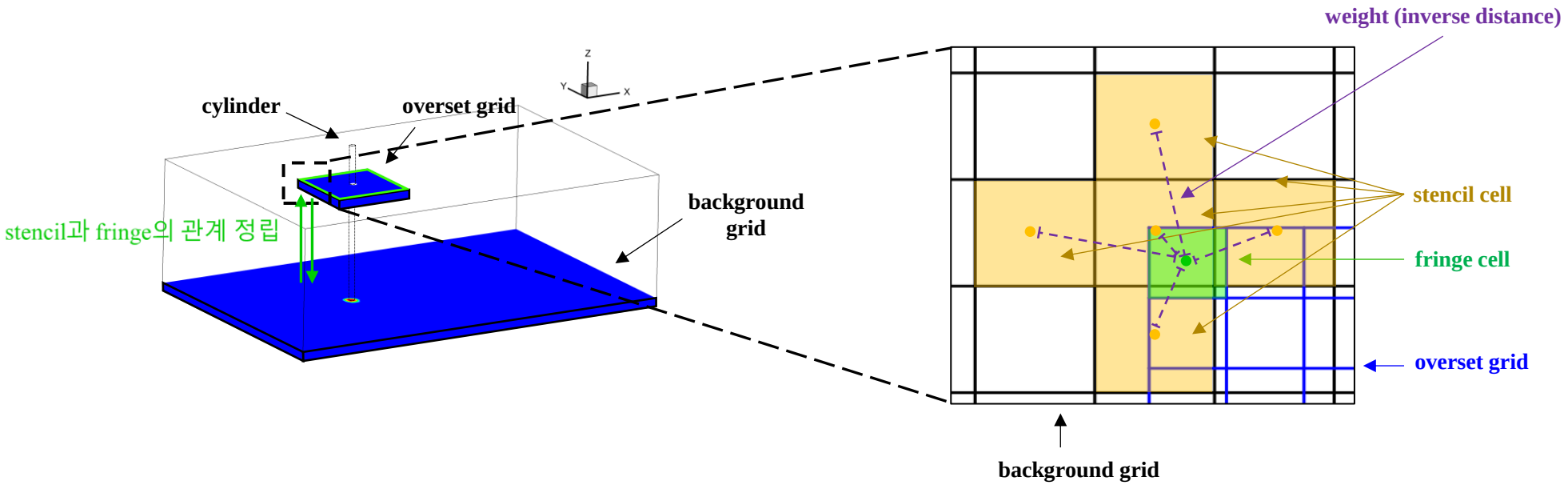


❖ overset

➢ 중첩된 cell 탐색

- 다른 격자계로부터 중첩된 cell을 탐색하는 단계
- 중첩된 cell들은 서로 정보를 전달

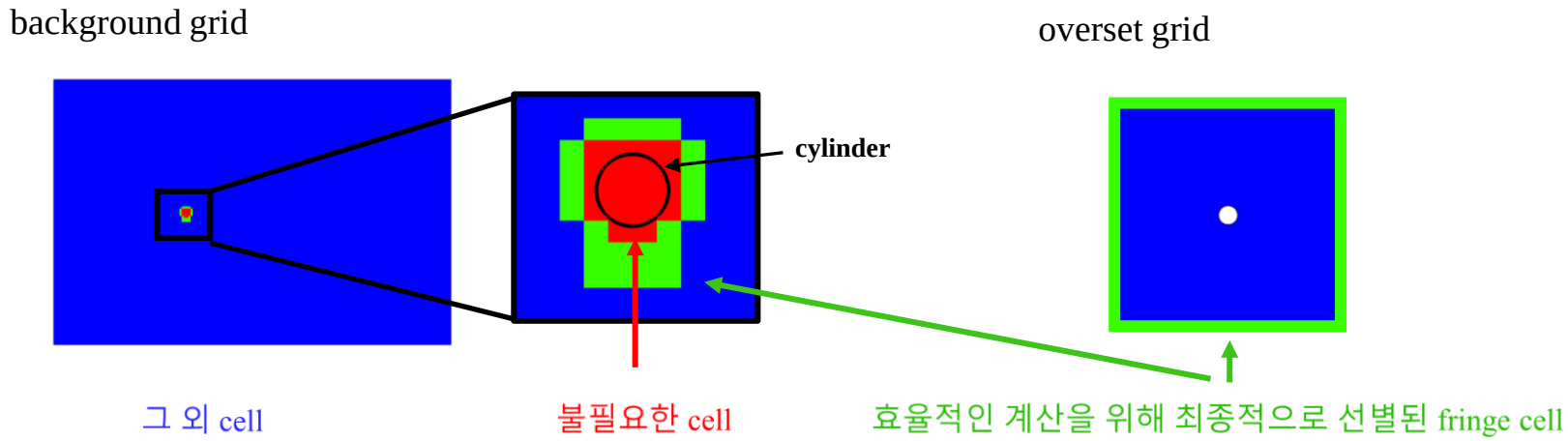
✓ stencil : 정보를 주는 cell ✓ fringe : 정보를 받는 cell ✓ weight : fringe가 가지고 있는 stencil의 가중치 정보
ex) inverse distance, volume



- 불필요한 cell을 제외한 모든 cell들에 대하여 stencil, fringe, weight 정보를 계산

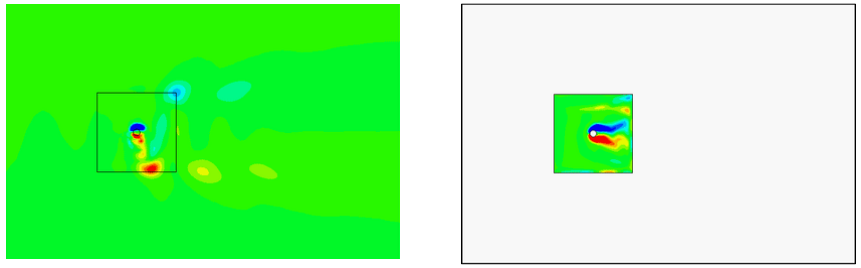
❖ overset

➢ cell type 선정

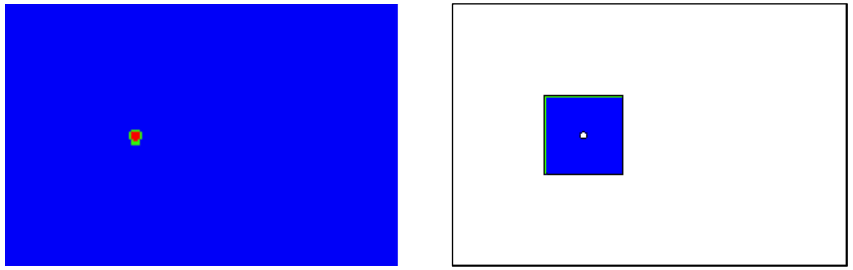


- cell type 선정 알고리즘은 격자계의 움직임이 있는 경우 모든 time step에서 진행

2D cylinder oscillation vorticity Z contour



2D cylinder oscillation cell type contour



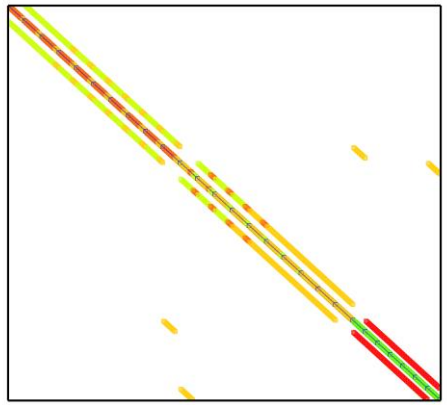
❖ overset

➢ overset matrix 수정

- overset 계산으로 나온 stencil, weight 정보를 이용하여 matrix 재구성
- matrix의 계수를 수정하는 단계에 있어서 대칭 행렬이 비대칭 행렬이 되는 경우도 존재

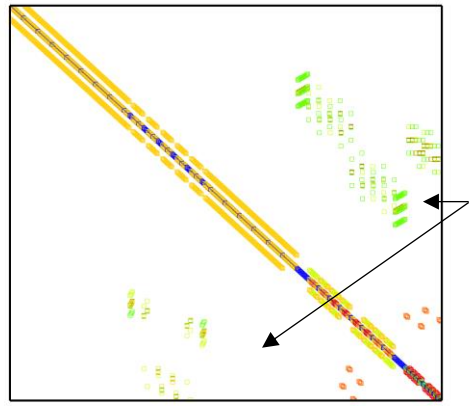
➢ overset matrix visualization

overset을 사용하지 않은 case의 matrix



주 대각 성분으로 상삼각 성분과 하삼각 성분이 대칭을 유지 (symmetry matrix)

overset을 사용한 case의 matrix



각 계수들의 수정을 통해 행렬의 대칭성 붕괴 (asymmetry matrix)

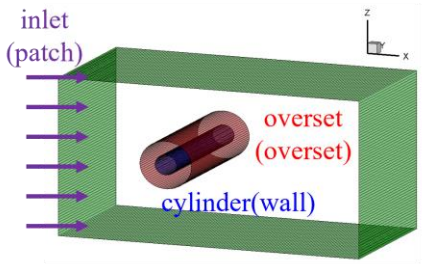
➢ open source code인 OpenFOAM을 이용하여 overset이 코드로 어떻게 구현되어 있는지 분석

❖ OpenFOAM overset

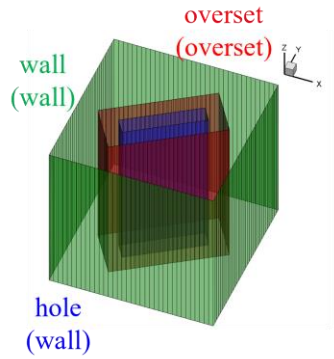
➢ OpenFOAM overset

- OpenFOAM-v1706에서 부터 중첩 격자 기능 지원
- overset tutorials (OpenFOAM-v2206)

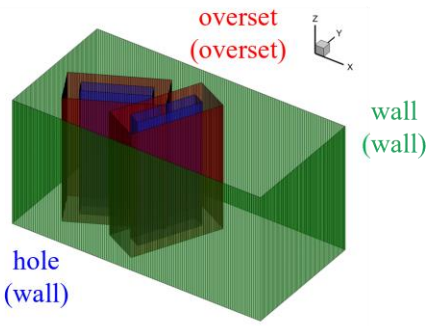
✓ cylinder



✓ simpleRotor



✓ twoSimpleRotor



- OpenFOAM overset interpolation schemes
  - ✓ cellVolumeWeight, inverseDistance, leastSquares, trackingInverseDistance
- OpenFOAM overset motion
  - ✓ translate, rotate, osciilation, 6DOF, etc.

❖ OpenFOAM overset

➢ OpenFOAM overset cell type

- hole type : overset 계산에 불필요한 cell
- interpolated type : stencil cell로부터 정보를 받아오는 fringe cell
- calculated type : 그 외 cell

➢ OpenFOAM overset 알고리즘

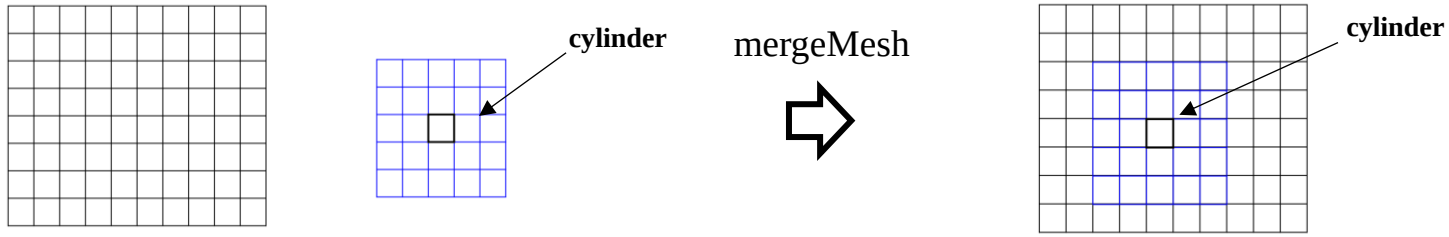
격자 정보 전달 ➡ 불필요한 cell 탐색 ➡ 중첩된 cell 탐색 ➡ cell type 선정 ➡ matrix 수정

<calculated type으로 초기화> ➡ <hole type 선정> ➡ < 중첩된 cell 탐색> ➡ <interpolated type 선정> ➡ <stencil 추가 선정 >

<stencil과 weight 정보를 통해 matrix 수정>

- example (2D square cylinder)
  - ✓ mesh (cell count : 124)

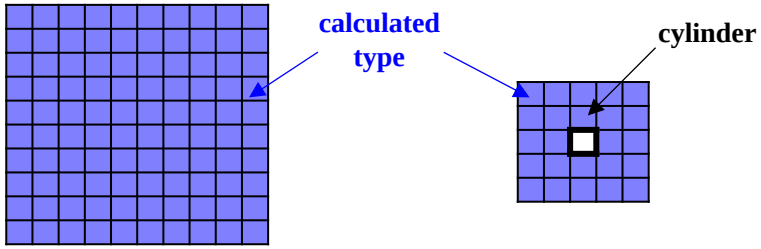
background grid (10x10) overset grid (5x5)



❖ OpenFOAM overset

➢ overset code algorithm (1)
< calculated type으로 초기화 >

- background grid
- overset grid



contour color : hole, interpolated, calculated

✓ fvSchemes 파일에 있는 overset option을 적용

✓ 모든 cell들을 calculated type으로 설정

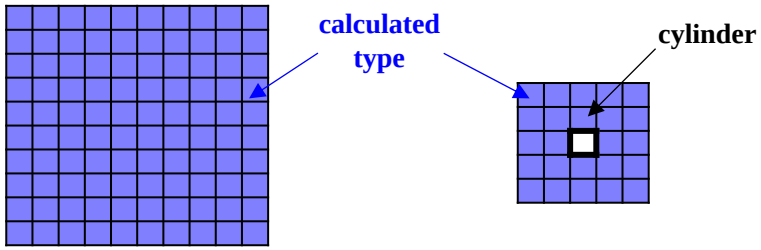
inverseDistanceCellCellStencil.C:1730

```
1730 bool Foam::cellCellStencils::inverseDistance::update()
1731 {
1732     scalar layerRelax(dict_.getOrDefault("layerRelax", 1.0));
1733
1734     scalar tol = dict_.getOrDefault("tolerance", 1e-10);
1735     smallVec_ = mesh_.bounds().span()*tol;
1736
1737     const labelIOList& zoneID = this->zoneID();
1738
1739     label nZones = gMax(zoneID)+1;
1740     labelList nCellsPerZone(nZones, Zero);
1741     forAll(zoneID, cellI)
1742     {
1743         nCellsPerZone[zoneID[cellI]]++;
1744     }
1745     Pstream::listCombineAllGather(nCellsPerZone,
plusEqOp<label>());
1746
1747     const boundBox& allBb(mesh_.bounds());
1748
1749     PtrList<fvMeshSubset> meshParts(nZones);
1750List<treeBoundBoxList> meshBb(nZones)
.
.
.
1905     // Current best guess for cells. Includes best stencil. Weights
1906     // add up to volume.
1907     labelList allCellTypes(mesh_.nCells(), CALCULATED);
1908     labelListList allStencil(mesh_.nCells());
1909     // zoneID of donor
1910     labelList allDonorID(mesh_.nCells(), -1);
1911
1912     const globalIndex globalCells(mesh_.nCells());
1913
1914     PstreamBuffers pBufs(Pstream::commsTypes::nonBlocking);
```

❖ OpenFOAM overset

➢ overset code algorithm (2)
<hole type 선정>

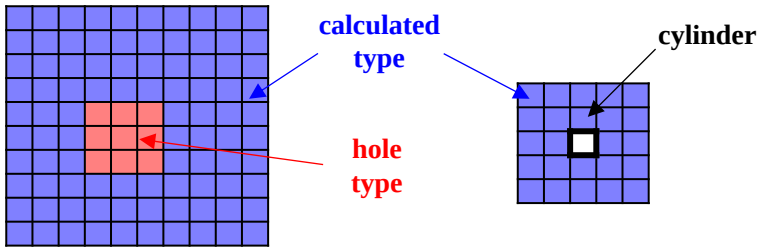
- background grid
- overset grid



contour color : hole, interpolated, calculated

✓ 불필요한 cell 선정

- background grid
- overset grid



contour color : hole, interpolated, calculated

✓ cylinder가 중첩된 영역의 background grid의 cell을 hole type으로 선정

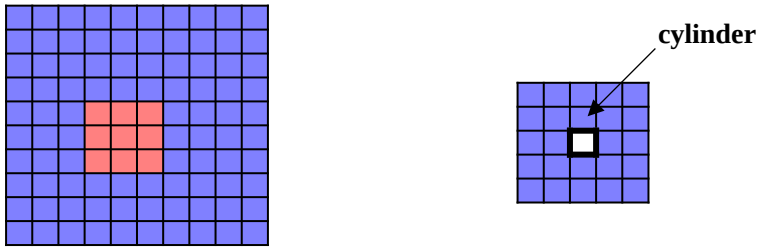
inverseDistanceCellCellStencil.C:398

```
398         UIPstream is(procI, pBufs);
399         {
400             treeBoundBox receivedBb(is);
401             if (srcPatchBb != receivedBb)
402             {
403                 FatalErrorInFunction
404                     << "proc:" << procI
405                     << " srcPatchBb:" << srcPatchBb
406                     << " receivedBb:" << receivedBb
407                     << exit(FatalError);
408             }
409         }
410         const labelVector zoneDivs(is);
411         const PackedList<2> srcPatchTypes(is);
412
413         forAll(tgtCellMap, tgtCelli)
414         {
415             label celli = tgtCellMap[tgtCelli];
416             treeBoundBox cBb(cellBb(mesh_, celli));
417             cBb.min() -= smallVec_;
418             cBb.max() += smallVec_;
419             if
420             (
421                 overlaps
422                 (
423                     srcPatchBb,
424                     zoneDivs,
425                     srcPatchTypes,
426                     cBb,
427                     patchCellType::PATCH
428                 )
429             )
430             {
431                 allCellTypes[celli] = HOLE;
432             }
```

❖ OpenFOAM overset

➢ overset code algorithm (3)
<중첩된 cell 탐색>

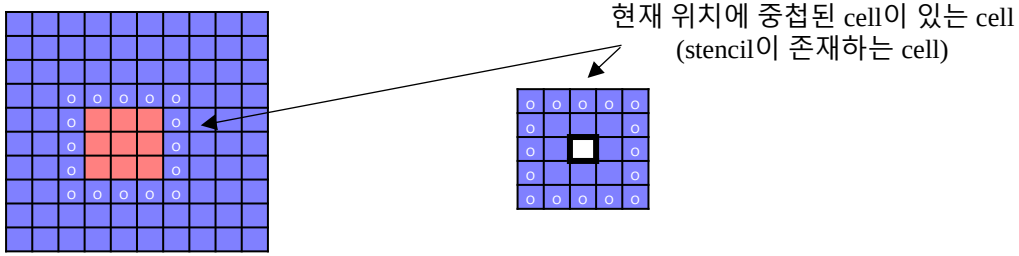
- background grid
- overset grid



contour color : hole, interpolated, calculated

✓ hole type을 제외한 모든 cell에 대해 중첩된 cell 탐색

- background grid
- overset grid



contour color : hole, interpolated, calculated

✓ stencil과 fringe cell 사이의 관계 정립

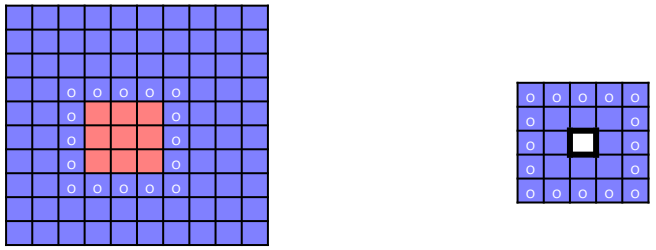
waveMethod.C:89

```
81      meshToMeshData::trackData td(tgt);
82
83      label startCelli = 0;
84
85      while (true)
86      {
87          changedFaces.clear();
88          changedFacesInfo.clear();
89
90          // Search for starting seed
91          for (; startCelli < src.nCells(); startCelli++)
92          {
93              if (!cellData[startCelli].valid(td))
94              {
95                  nSeeds++;
96                  const point& cc = src.cellCentres()[startCelli];
97
98                  if (!tgtBb.contains(cc))
99                  {
100                     // Point outside local bb of tgt mesh. No need to
101                     // search. Register as no correspondence
102                     cellData[startCelli] = meshToMeshData(-1);
103                 }
104                 else
105                 {
106                     label tgtCelli = tgt.findCell(cc,
polyMesh::CELL_TETS);
107                     if (tgtCelli != -1)
108                     {
109                         // Insert any face of cell
110                         label facei = src.cells()[startCelli][0];
111                         changedFaces.append(facei);
112 changedFacesInfo.append(meshToMeshData(tgtCelli));
```

❖ OpenFOAM overset

➢ overset code algorithm (4) <interpolated type 선정>

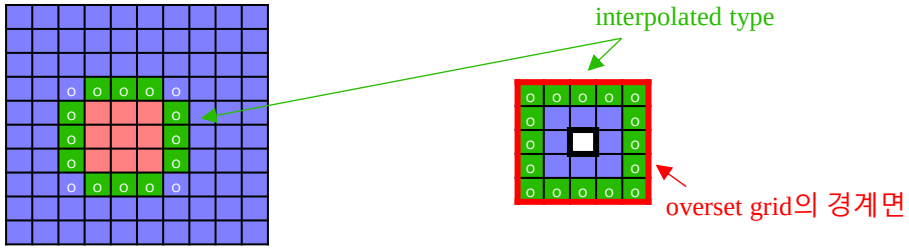
- background grid
- overset grid



contour color : hole, interpolated, calculated

✓ hole과 맞닿은 cell과 overset grid의 경계면과 맞닿은 cell들을 interpolated type으로 설정

- background grid
- overset grid



contour color : hole, interpolated, calculated

✓ interpolated type cell은 무조건 stencil을 가지고 있는 cell만 가능

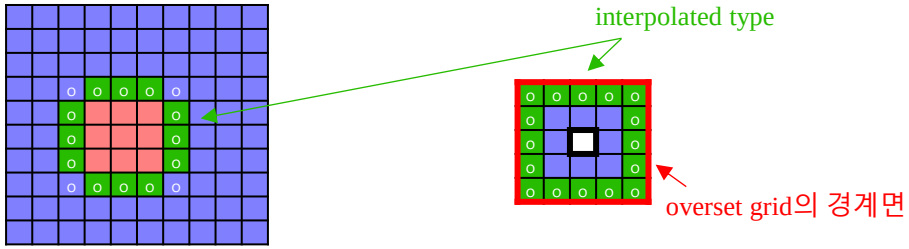
inverseDistanceCellCellStencil.C:1233

```
1994
1995    // Use the patch types and weights to decide what to do
1996    forAll(allPatchTypes, cellI)
1997    {
1998        if (allCellTypes[cellI] != HOLE)
1999        {
2000            switch (allPatchTypes[cellI])
2001            {
2002                case OVERSET:
2003                {
2004                    // Require interpolation. See if possible.
2005
2006                    if (allStencil[cellI].size())
2007                    {
2008                        allCellTypes[cellI] = INTERPOLATED;
2009                    }
2010                    else
2011                    {
2012                        // If there are no donors we can either set the
2013                        // acceptor cell to 'hole' i.e. nothing gets calculated
2014                        // if the acceptor cells go outside the donor meshes
or
2015                        // we can set it to 'calculated' to have something
2016                        // like an acmi type behaviour where only covered
2017                        // acceptor cell use the interpolation and non-covered
2018                        // become calculated. Uncomment below line. Note:
this
2019                        // should be switchable?
2020                        //allCellTypes[cellI] = CALCULATED;
2021
2022                        allCellTypes[cellI] = HOLE;
2023                    }
2024                }
2025            }
2026    }
```

❖ OpenFOAM overset

➢ overset code algorithm (5)
< stencil 추가 선정 >

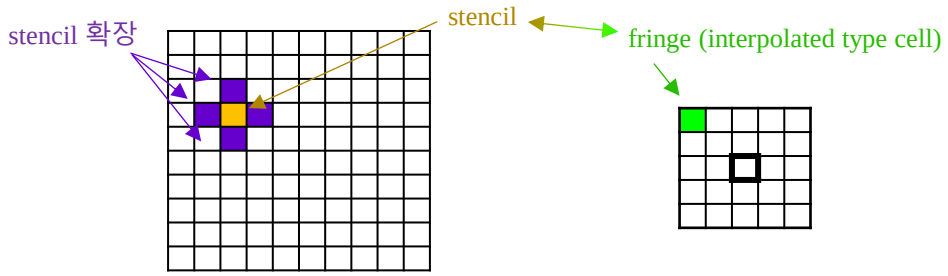
- background grid
- overset grid



contour color : hole, interpolated, calculated

✓ interpolated type cell의 stencil을 확장하는 단계

- background grid
- overset grid



contour color : hole, interpolated, calculated

✓ 기존의 stencil cell과 맞닿은 cell들을 stencil cell로 설정하여 확장

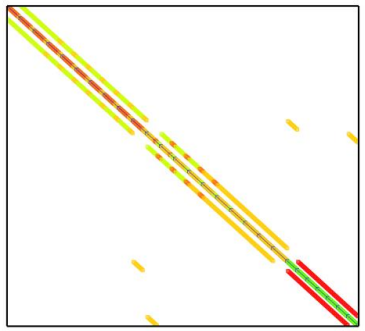
inverseDistanceCellCellStencil.C:1097

```
1589    // - donorCellCells : stencil (with first element the original
donor)
1590    // - donorWeights : weights for donorCellCells
1591    cellInterpolationMap().distribute(donorCellCells);
1592    cellInterpolationMap().distribute(donorWeights);
1593    cellInterpolationMap().distribute(samples);
1594
1595    // Check which acceptor has won and transfer
1596    forAll(interpolationCells_, i)
1597    {
1598        if (!doneAcceptor[i])
1599        {
1600            label cellI = interpolationCells_[i];
1601            const labelList& slots = cellStencil_[cellI];
1602
1603            if (slots.size() != 1)
1604            {
1605                FatalErrorInFunction << "Problem:" << slots
1606                    << abort(FatalError);
1607            }
1608
1609            label slotI = slots[0];
1610
1611            // Important: check if the stencil is actually for this cell
1612            if (samples[slotI] == mesh_.cellCentres()[cellI])
1613            {
1614                cellStencil_[cellI].transfer(donorCellCells[slotI]);
1615                cellInterpolationWeights_[cellI].transfer
1616                (
1617                    donorWeights[slotI]
1618                );
1619                // Mark cell as being done so it does not get sent over
1620                // again.
1621                doneAcceptor.set(i);
1622            }
```

❖ OpenFOAM overset

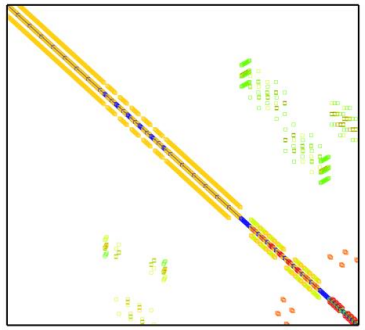
➢ overset code algorithm (6)
< stencil과 weight 정보를 통해 matrix 수정 >

- 수정 전 matrix



✓ stencil과 weight 정보를 이용하여 matrix 계수 수정
✓ 이 단계에서 symmetric matrix가 asymmetric matrix로 변경

- 수정 후 matrix



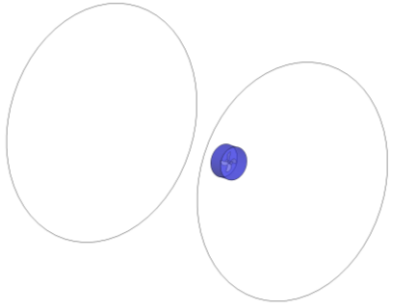
inverseDistanceCellCellStencil.C:1097

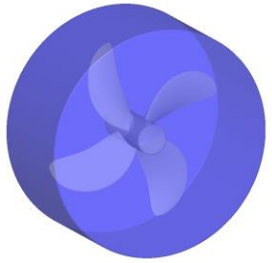
```
646    // Calculate stabilised diagonal as normalisation for interpolation
647    const scalarField norm(normalisation(m));
648
649    // Switch to extended addressing (requires mesh::update() having
been
650    // called)
651    active(true);
652
653    // Adapt matrix
654    scalarField oldUpper(m.upper());
655    scalarField oldLower(m.lower());
656    FieldField<Field, Type> oldInt(m.internalCoeffs());
657    FieldField<Field, Type> oldBou(m.boundaryCoeffs());
658    const label oldSize = bpsi.size();
659
660    addInterpolation(m, norm);
661
662    // Swap psi values so added patches have patchNeighbourField
663    correctBoundaryConditions<GeoField,
calculatedProcessorFvPatchField<Type>>
664    (
665        bpsi,
666        true
667    );
668
669    // Use lower level solver
670    SolverPerformance<Type> s(m.solve(dict));   // by CWKIM
671
672    // Restore boundary field
673    bpsi.setSize(oldSize);
674
675    // Restore matrix
676    m.upper().transfer(oldUpper);
677  m.lower().transfer(oldLower);
```

❖ OpenFOAM overset

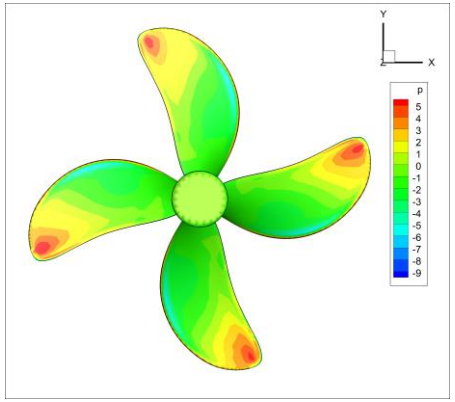
➢ OpenFOAM overset 계산 결과

- KP458 propeller open water test
  - ✓ background (82,770)
  - ✓ overset (160,831)

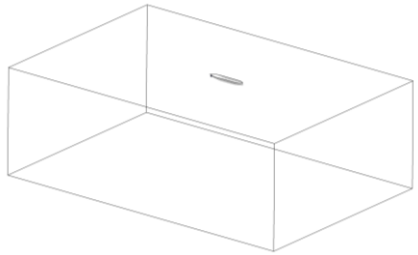




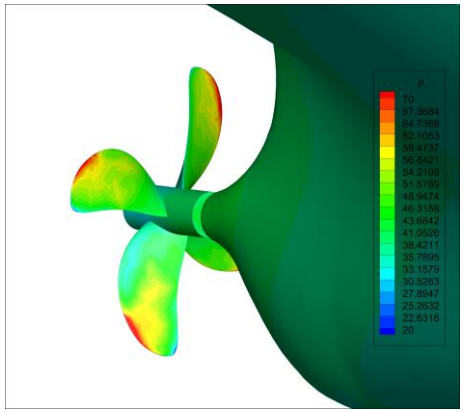
  - ✓ p contour



- KVLCC2 + KP458
  - ✓ background (962,490)
  - ✓ overset (634,967)



  - ✓ p contour

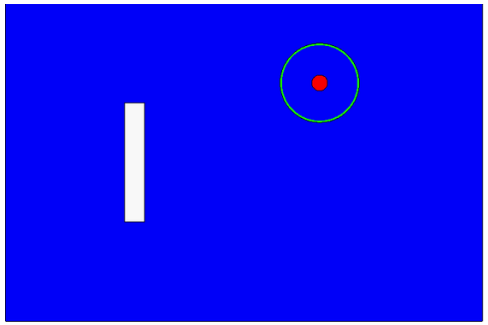


❖ OpenFOAM overset

➢ OpenFOAM overset이 가지고 있는 문제 <zero gap>

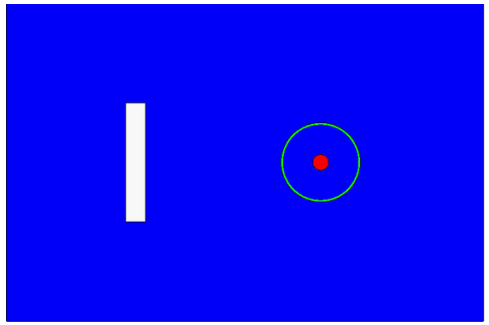
- 서로 다른 격자계의 wall과 wall 경계조건이 만날 때 발생
- 서로 다른 격자계의 wall type 경계가 서로 맞닿는 경우 interpolated type으로 선정된 cell의 stencil을 찾지 못해 에러가 발생

✓ wall 경계 조건이 가까이 지나가는 경우



contour color : hole, interpolated, calculated

✓ wall 경계 조건이 서로 만나는 경우



서로 만나는 순간 대부분의 cell들이 hole type으로 선정

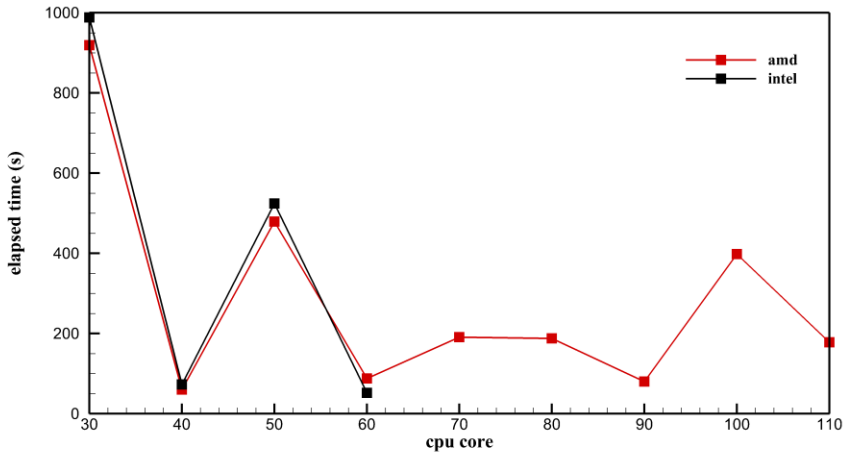
- overset 코드가 가지는 구조적인 문제
- STAR-CCM+의 경우 zero gap interface,
  suggar++의 경우 immersed boundary를 이용하여 다음과 같은 문제를 보완

❖ OpenFOAM overset

➢ OpenFOAM overset이 가지고 있는 문제 <병렬 계산 시간 증가>

- 특정 코어 수에서 계산 시간이 증가하는 문제가 존재

코어 수에 따른 계산 시간 비교 (cell count : 1.5M, 3step)



inverseDistanceCellCellStencil.C:566

```
566   forAll(tgtCellMap, tgtCelli)
567   {
568       label celli = tgtCellMap[tgtCelli];
569       if (srcOverlapProcs.size())
570       {
571           treeBoundBox subBb(cellBb(mesh_, celli));
572           subBb.min() -= smallVec_;
573           subBb.max() += smallVec_;
574
575           forAll(srcOverlapProcs, i)
576           {
577               label procI = srcOverlapProcs[i];
578               if (subBb.overlaps(srcBbs[procI]))
579               {
580                   tgtSendCells[procI].append(tgtCelli);
581               }
582           }
```

- 이는 중첩된 격자계에서 stencil을 찾는 알고리즘이 병렬에 최적화 되어있지 않아 발생
- stencil을 찾기 위해 decompose된 모든 격자들을 탐색해야 하며 이 과정에 mpi 문제 발생

➢ stencil을 찾는 최적 알고리즘을 개발 또는 타 코드 활용 가능성 검토 <suggar++>

❖ suggar++

➢ suggar++

- 서로 다른 격자계의 stencil (refered as dci in suggar++)를 찾아주는 overset 소프트웨어
- centos6, centos7, ubuntu 1604, ubuntu 1804, macosx 환경에서 지원
- 라이브러리 형태로 사용할 수 있으며, 이를 통해 다른 flow solver와 연결 가능
- structured, unstructured 격자 둘 다 지원
- mpi를 통한 병렬 계산 지원

➢ suggar++ 라이브러리 구성

- Suggar++********-centos7-64-dynamic.tar.gz : suggar++ 라이브러리
- DiRTlib : suggar++ 값을 타 프로그램과 연결하기 위해 필요한 라이브러리
- P3Dlib : DiRTlib 컴파일을 위해 필요한 프로그램

- GKlib : suggar++의 병렬 계산을 하기 위해 필요한 라이브러리 (1)
- METIS : suggar++의 병렬 계산을 하기 위해 필요한 라이브러리 (2)
- ParMETIS : suggar++의 병렬 계산을 하기 위해 필요한 라이브러리 (3)

} open source code

❖ suggar++

➢ suggar++ 계산 파일 구성

- input data

1. Input.xml
   : 계산 정보및 경계조건을 저장하는 파일
   : xml 형식으로 구성

2. Grids
   : 격자 정보를 저장하는 폴더
   : plot3d, ugrid, cobalt등 격자 포맷 사용 가능

- output data

1. composite (grid)
   : Grid 정보를 다 합친 하나의 격자를 생성

2. output++.dci
   : 격자들의 연결 정보를 담은 파일
   : cell stencil, cell weight, transformation matrix

- suggar++ input.xml example

```
<global>
  <ignore_direction dir="z"/>
<output>
  <composite_grid filename="composite.p3dudl" style="p3d"/>
  <domain_connectivity filename="output++.dci" style="dci"/>
</output>
  <body name="Root">
    <body name="airfoil">
      <volume_grid name="0012"
        filename="Grids/0012-3d.p3du" style="p3d"/>
    </body>
  <body name="outer">
    <volume_grid name="outer"
      filename="Grids/cart-3d.p3du" style="p3d"/>
</body>
```

- suggar++에서 지원하는 격자 체계

structured grids:

- ✓ Cartesian
- ✓ Curvilinear including block-to-block grids

Unstructured grids:

- ✓ All tetrahedral
- ✓ Mixed element
- ✓ Octree-base Cartesian (cell-centered only)
- ✓ **General polyhedral elements**

❖ suggar++

➢ suggar++ 병렬 계산

- suggar++의 병렬 계산을 위해서 계산 코어 수 만큼 격자를 나누는 과정이 필요

※ GKlib, METIS, ParMETIS 라이브러리 필요. 환경 변수 설정 후 suggar++ 컴파일 진행 (ex. PARMETISINX_DIR)

✓ suggar++ -partition_mesh (N)

격자계를 N개로 나누는 과정
완료후 Grids-ParMetis-NP-(N) 이라는 폴더 생성

→ partition_mesh options : parMetis, SDV

- suggar++ 병렬 계산 실행 후 dci 정보는 partition 진행 전 기존 격자 정보를 토대로 출력

❖ suggar++

➢ suggar++와 flow solver 연동 사례

- suggar++의 경우 dci 정보만 출력해주기 때문에 계산을 위해선 추가 flow solver가 필요
- suggar++를 연동하여 overset 계산을 하는 다양한 프로그램 존재

✓ **foamedOver**

Boger, D.A., Noack, R.W. and Paterson, E.G. (2010) FoamedOver: A Library to Add a Dynamic Overset Grid Capability to OpenFOAM. *5th Open-FOAM Workshop*, Chalmers University of Technology, Gothenburg, 21-24 June 2010, 24

✓ **Applied CCM : Caelus**

Zhang, C. H. E. N. L. I. A. N. G., et al. "Implementation of Overset Grid in OpenFOAM and its validation to PMM model test of a container ship." *Proceedings of the 13th OpenFOAM Workshop (OFW13), Shanghai, China*. 2018.

✓ **WAVIS**

김유철, 김윤식, 김진, & 김광수. (2019). 선박의 유동해석 문제에 대한 중첩격자기법 (Suggar++) 의 활용. *대한조선학회 논문집*, *56*(1), 47-57.

✓ **OpenFOAM**

Gopalan, H., Jaiman, R., & Chandar, D. D. (2015). Flow past tandem circular cylinders at high Reynolds numbers using overset grids in OpenFOAM. In *53rd AIAA Aerospace Sciences Meeting* (p. 0315).

✓ **CFDShip-IOWA**

Martin, J. E., Michael, T., & Carrica, P. M. (2015). Submarine maneuvers using direct overset simulation of appendages and propeller and coupled CFD/potential flow propeller solver. *Journal of Ship Research*, *59*(01), 31-48.

❖ suggar++

➢ OpenFOAM oveset과 suggar++ 비교

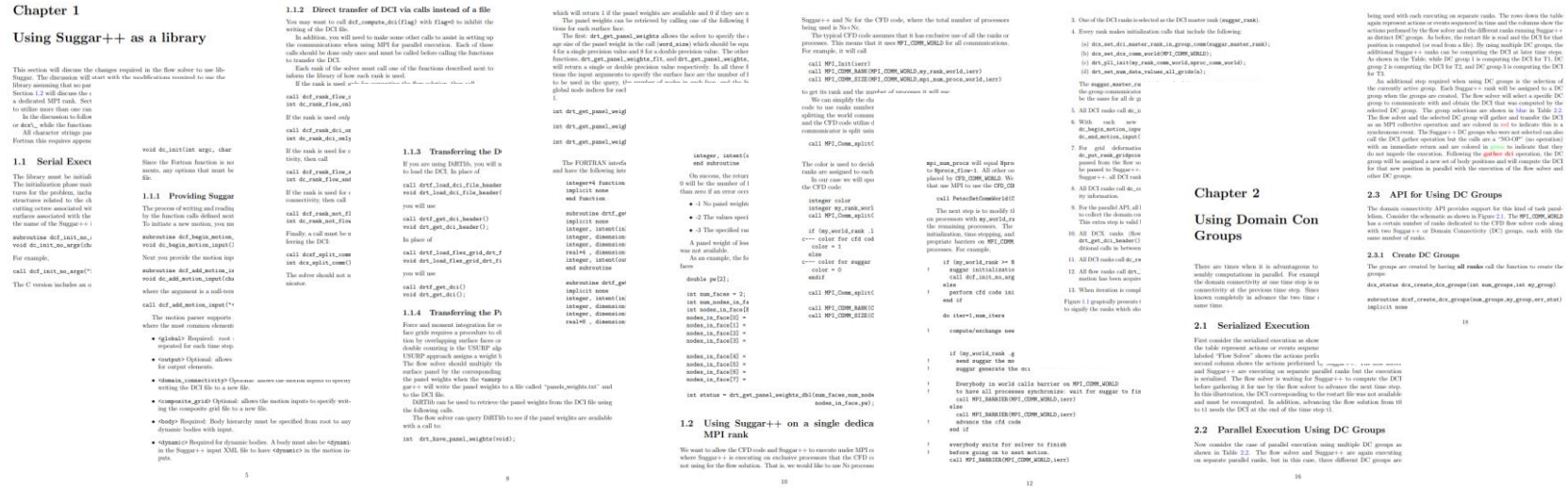
	OpenFOAM overset (v2206)	**suggar++**
mesh format	OpenFOAM (polyMesh)	plot3d, VGRID, UGRID, Cobalt, AFLR3, CGNS
mesh motions	translate, rotate, osciilation, 6DOF..	translate, rotate, rotate_about_v, scale
hole cutting method	direct cutting	query cutters, direct cutting, xray cutting,octree-based cutting, hybrid cutting, etc.
interpolation schemes	cellVolumeWeight, inverseDistance, leastSquares, trackingInverseDistance	linear, lagrangian, donor_cell_only, injection, least_sq, inv_dist, laplacian, clipped_laplacian
flow solver	overLaplacianDyMFoam, overSimpleFoam, overPimpleDyMFoam, overInterDyMFoam, overRhoPimpleDyMFoam	X
zero gap problem	X	immersed boundary approach
overset cell types	hole, interpolated, calculated	in, in_ib, out, out_ib, fringe_inner, fringe_outer, periodic, orphan …
general polyhedral mesh parallel computation	O	X

INDEX

❖ suggar++

➢ suggar++와 OpenFOAM 연동 코드 개발

- OpenFOAM 격자를 suggar++에서 지원하는 격자 포맷으로 변환하는 유틸리티 개발
- OpenFOAM에 설정된 motion 정보를 suggar++에 전달
- suggar++에서 계산된 dci 정보를 OpenFOAM으로 전달
- 병렬 계산을 위한 코드 개발
  - ✓ suggar++_programmerse_guide



Chapter 1

Using Suggar++ as a library

Chapter 2

Using Domain Con
Groups

❖ suggar++

➢ suggar++와 OpenFOAM 연동 (격자 변환 유틸리티 개발)

- suggar++가 지원하는 격자 포맷 테스트 진행
  - ✓ general polyhedral mesh를 구현할 수 있는 Cobalt 포맷으로 결정
- OpenFOAM 격자를 Cobalt 포맷의 격자로 변환할 수 있는 foamToCobalt 유틸리티 개발
- 경계 조건의 경우 아래의 표에 따라 변환하여 사용

OpenFOAM bc	suggar++ bc
fixedValue, zeroGradient	farfield
overset	overlap
wall	solid, non-solid
symmetry, empty	symmetry
cyclic	periodic

- foamToCobalt.C:72

```
72    os<<"3 1 "<<bMesh.size()<<std::endl;
73    os<<nPoints<<" "<<nFaces<<" "<<nCells;
74
75    label nVerticesPerFaceMax = 0;
76    for(int faceI=0; faceI<nFaces; faceI++)
77    {
78        const face& f = faces[faceI];
79
80        if(nVerticesPerFaceMax < f.size())
81            nVerticesPerFaceMax = f.size();
82    }
84    label nFacesPerCellMax = 0;
85    for(int cellI=0; cellI<nCells; cellI++)
86    {
87        const cell& c = cells[cellI];
89        if(nFacesPerCellMax < c.size())
90            nFacesPerCellMax = c.size();
91    }
93    os<<" "<<nVerticesPerFaceMax<<"
"<<nFacesPerCellMax<<std::endl;
```

❖ suggar++

➢ suggar++와 OpenFOAM 연동 (motion 정보 전달)

- dynamicMeshDict와 time step에 따라 달라지는 격자의 움직임을 매 step마다 suggar++로 전달 필요
- OpenFOAM에서 사용하는 값을 suggar++의 형식에 맞게 변환

OpenFOAM

septernion타입의 변수를 이용하여 격자 이동을 결정

→ 병진 운동의 정보를 가지는 vector v_ (T_x, T_y, T_z);

→ 회전 운동의 정보를 가지는 quaternion r_$(R_w, (R_x, R_y, R_z))$;

⇩

suggar++

suggar++의 경우 4 by 4의 transformation matrix를 통해 격자 이동을 결정

OpenFOAM의 septernion 변수를 이용해서 transformation matrix 구성

$$\begin{Bmatrix} (R_w + R_x - R_y - R_z) & (2R_xR_y - 2R_wR_z) & (2R_xR_z + 2R_wR_y) & (T_x) \\ (2R_xR_y + 2R_wR_z) & (R_w - R_x + R_y - R_z) & (2R_yR_z - 2R_wR_x) & (T_y) \\ (2R_xR_z - 2R_wR_y) & (2R_yR_z + 2R_wR_x) & (R_w - R_x - R_y + R_z) & (T_z) \\ (0) & (0) & (0) & (1) \end{Bmatrix}$$

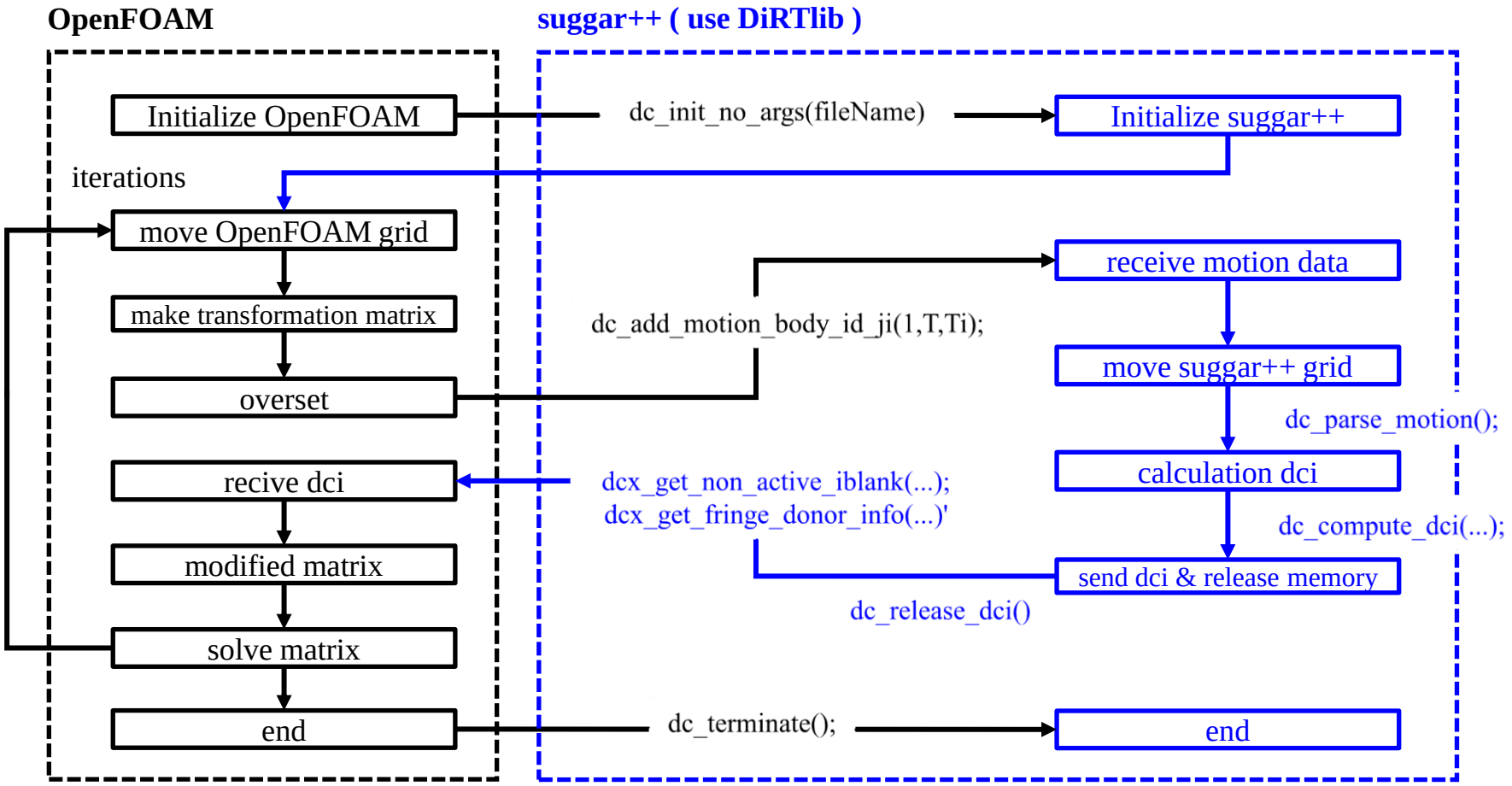
- suggarCellCellStencil.C:351

```
IOdictionary dynDict(ioDynDict);
    const dictionary& coefDict =
dynDict.subDict("multiSolidBodyMotionSolverCoeffs");

    // fixed this code <optimization>
    const dictionary& cellZoneDict =
coefDict.subDict("movingZone");

    autoPtr<solidBodyMotionFunction> sMPtr
    (
        solidBodyMotionFunction::New(cellZoneDict,
mesh_.time())
    );

    const solidBodyMotionFunction& sM = sMPtr();

    scalarRectangularMatrix transformationMatrix(4,4);
    const septernion& transformation = sM.transformation();
    const vector& translatingInfo = transformation.t();
    const tensor& rotatingInfo = transformation.r().R();
```

❖ suggar++

➢ suggar++와 OpenFOAM 연동 (dci 정보 전달)

- DiRT 라이브러리를 통해 fortran 언어 기반의 suggar++를 C++ 언어로 구성된 OpenFOAM에 연동



❖ suggar++

➢ suggar++와 OpenFOAM 연동 (dci 정보 전달)

- DiRT 라이브러리를 통해 fortran 언어 기반의 suggar++를 C++ 언어로 구성된 OpenFOAM에 연동

suggarCellCellStencil.H:44

```
ifndef cellCellStencils_suggar_H
#define cellCellStencils_suggar_H

#include "cellCellStencil.H"
#include "volFields.H"
#include "labelVector.H"
#include "treeBoundBoxList.H"
#include "pointList.H"
#include "globalIndex.H"
#include "bitSet.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

namespace Foam
{

class fvMeshSubset;

namespace cellCellStencils
{

/*---------------------------------------------------------------------------*\
                Class suggar Declaration
\*---------------------------------------------------------------------------*/

class suggar
:
    public cellCellStencil
```

suggarCellCellStencil.C:58

```
extern "C" {
    // DiRTlib Function
    void drt_get_dci_header();
    void drt_get_dci();
    double drt_wallclock();
    int drt_get_number_dci_grids();
    void drt_set_dci_grid_index(int index);
    int drt_get_current_dci_grid_index();
    struct drt_grid *drt_get_current_dci_grid();
    void drt_filter_blanked_points(int ngrid_points, int *iblank);
    void drt_set_fringe_type_for_receptors(int *iblank,int grid_index);
    void drt_print_call_trace(const char* const str, char* args);
    int drt_is_az_buf_enabled();
    int0 drt_get_solver_time_step();
    //void drt_process_panel_weights(int ngrids);
    int drt_get_az_buf_n_per_rev();
    int1 drt_get_az_buf_key();
    void drt_restore_saved_drt_lists(int1 key);
    int  drt_is_dci_header_loaded();
}

// * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * //
namespace Foam
{
namespace cellCellStencils
{
    defineTypeNameAndDebug(suggar, 0);
    addToRunTimeSelectionTable(cellCellStencil, suggar, mesh);
```

❖ suggar++

➢ suggar++와 OpenFOAM 연동 (dci 정보 전달)

- DiRT 라이브러리를 통해 fortran 언어 기반의 suggar++를 C++ 언어로 구성된 OpenFOAM에 연동

suggarCellCellStencil.H:296

```
//- Indices of interpolated cells
    virtual const labelUList& interpolationCells() const
    {
        return interpolationCells_;
    }

    //- Return a communication schedule
    virtual const mapDistribute& cellInterpolationMap() const
    {
        //if (!cellInterpolationMap_)
        if (!cellInterpolationMap_.valid())
        {
            const_cast<suggar&>(*this).update();
        }
        return cellInterpolationMap_();
    }

    //- Per interpolated cell the neighbour cells (in terms of slots as
    //  constructed by above cellInterpolationMap) to interpolate
    virtual const labelListList& cellStencil() const
    {
        return cellStencil_;
    }

    //- Weights for cellStencil
    virtual const scalarListList& cellInterpolationWeights() const
    {
        return cellInterpolationWeights_;
    }
```

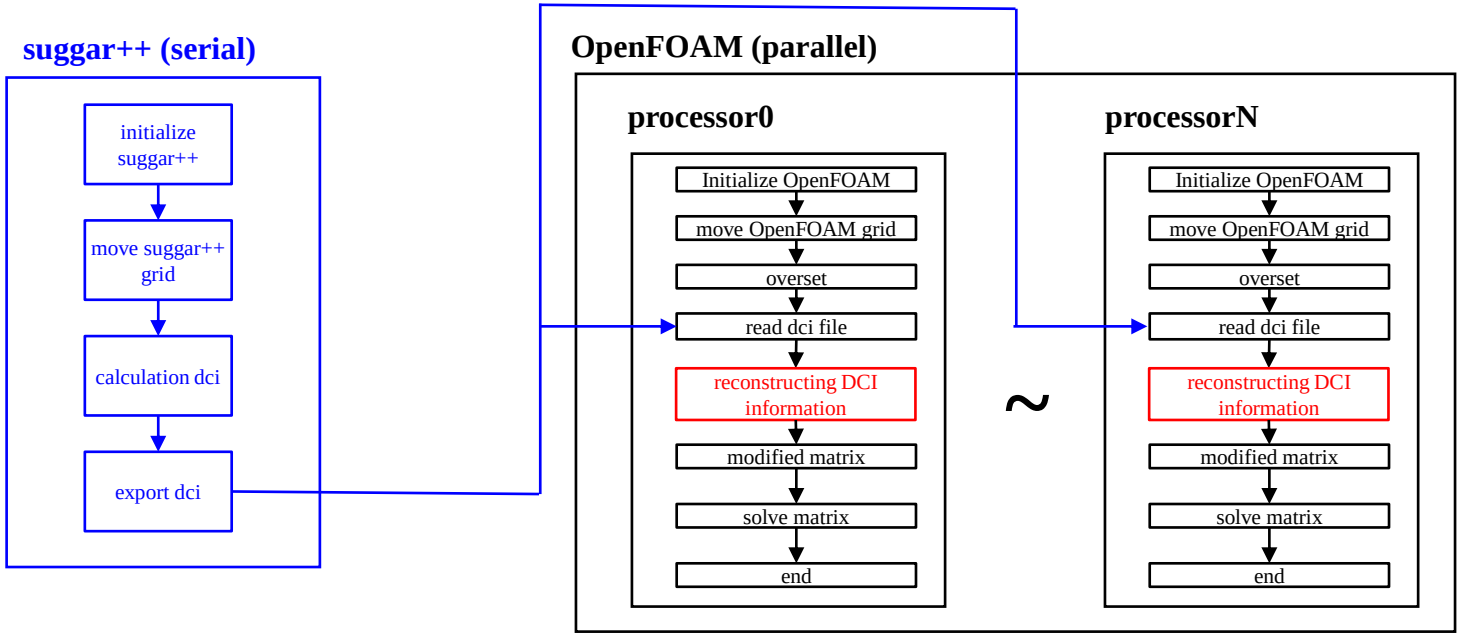
suggarCellCellStencil.C:546

```
 int indices[nFringes[igrid]];
        int nDonorMembers[nFringes[igrid]];
//if(this_is_dci_master_rank)
        dcx_get_fringe_indices(igrid+1, indices, nDonorMembers);
        int sumNDonorMembers =
dcx_get_sum_n_donor_members(igrid+1);
        if(sumNDonorMembers)
        {
            int donorMemberGrids[nFringes[igrid]];
            int donorMemberIndices[sumNDonorMembers];
            double donorMemberWeights[sumNDonorMembers];
            dcx_get_fringe_donor_info(igrid+1, donorMemberGrids,
donorMemberIndices, donorMemberWeights);
                        int dd=0;
            for(int i=0; i<nFringes[igrid]; i++)
            {
                label cellMemberGrids = donorMemberGrids[i]-1;
                labelList cellStencil(nDonorMembers[i]);
                scalarList cellWeight(nDonorMembers[i]);
                            for(int j=0; j<nDonorMembers[i]; j++)
                {
                    cellStencil[j] = startZoneCellIndex[cellMemberGrids] +
donorMemberIndices[dd] - 1;
                    cellWeight[j] = donorMemberWeights[dd] + VSMALL;
dd++;
                }
                suggarCellStencil[startZoneCellIndex[igrid] + indices[i] -1]
= cellStencil;
                suggarCellInterpolationWeights[startZoneCellIndex[igrid] +
```

❖ suggar++

➢ suggar++와 OpenFOAM 연동 (병렬 계산)

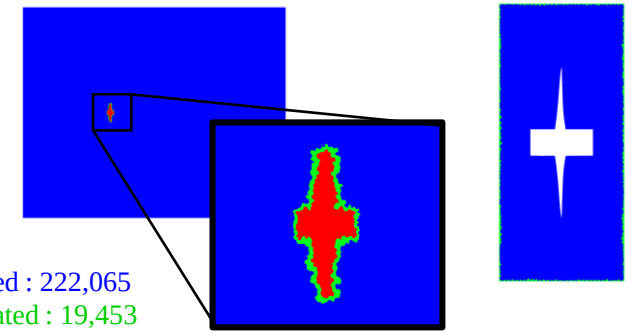
- suggar++는 general polyhedral mesh인 경우 partition_mesh를 하는 코드가 아직 개발되지 않아 병렬 계산 진행 X
- suggar++를 단일 코어로 계산한 후, 출력된 dci를 바탕으로 OpenFOAM에서 병렬로 실행
- suggar++의 격자 정보와 decompose된 OpenFOAM 격자 정보를 일치시키는 과정이 필요



❖ suggar++

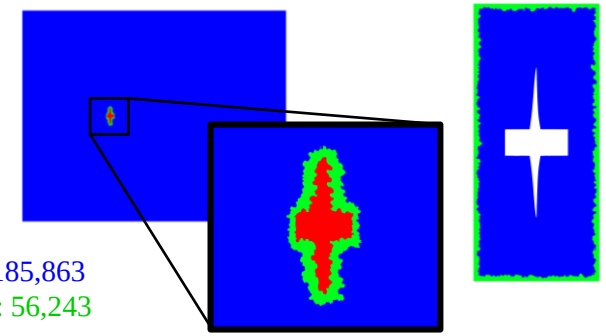
➢ suggar++와 OpenFOAM 연동 (병렬 계산 결과)

- POW KP458 (polyhedral)
  - OpenFOAM cell type



calculated : 222,065
interpolated : 19,453
hole : 2,083

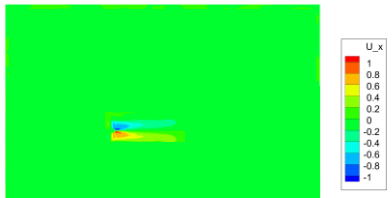
  - suggar++ cell type



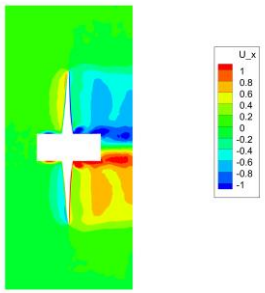
calculated : 185,863
interpolated : 56,243
hole : 1,495

  - OpenFOAM Ux contour

background

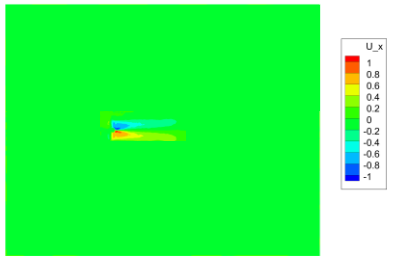


overset

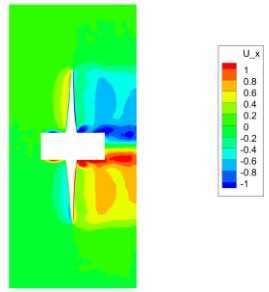


  - suggar++ Ux contour

background



overset



❖ suggar++

➢ 코드 개발 과정 및 추후 계획

- 코드 개발 과정 (1년)

git hub에 업로드 예정

1) overset 기법 분석 및 선행 연구 조사 (1달)

2) OpenFOAM overset 코드 분석 (3달)

3) suggar++ 실행 및 분석 (1달)

4) OpenFOAM과 suggar++를 연동하기 위해 각종 유틸리티 개발 (3달)

5) DiRT 라이브러리를 이용하여 OpenFOAM과 suggar++ 코드 연동 (serial) (2달)

6) OpenFOAM과 suggar++ 코드 연동 (parallel) (2달)

- DiRT 라이브러리를 이용하여 OpenFOAM과 suggar++ 연동한 serial 코드는 튜토리얼과 메뉴얼을 제작 후 git hub에 업로드 예정

❖ 결론

- OpenFOAM overset과 suggar++ 비교 및 분석 진행
- OpenFOAM overset
  - OpenFOAM overset에서 제공하는 options 확인 및 cell type 선정 알고리즘을 분석
  - zero gap, 병렬 계산 지연 문제 등 다양한 문제점들을 확인
- suggar++
  - suggar++의 구성 및 특징들을 분석하였으며, suggar++와 OpenFOAM overset을 직접적으로 비교
  - general polyhedral mesh 병렬 계산을 하지 못한다는 문제점을 확인
- suggar++와 OpenFOAM을 연동할 수 있도록 코드 개발을 진행