

OpenFOAM의 중첩 격자 라이브러리에 대한 분석

동아대학교

김찬우, 이상봉

INDEX 목차 目次

01 서론

02 중첩 격자 해석 인자 평가

03 중첩 격자의 문제점

04 overset cell type

05 결론

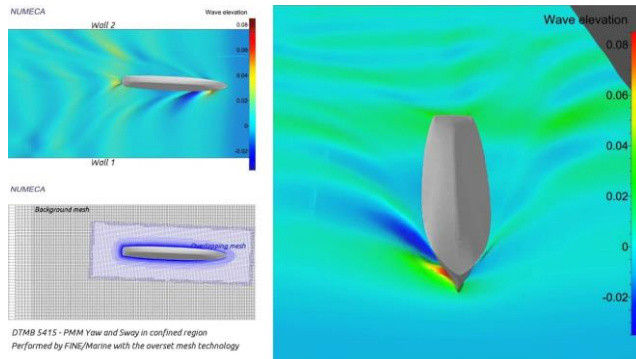
❖ 서론

INDEX

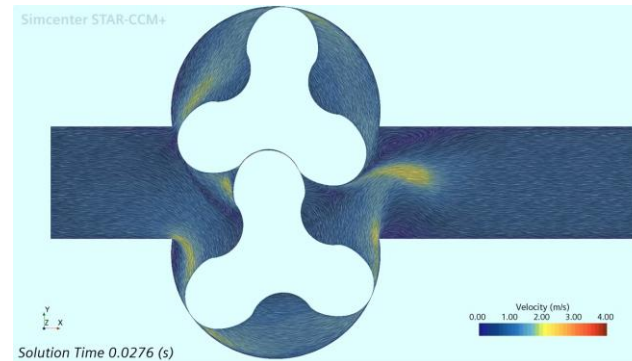
- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

- 물체의 운동을 CFD에서 구현하기 위해 다양한 방법 존재 (slidingMesh, overset, etc.)
- 동체를 포함한 복잡한 대상의 해석 격자를 보다 유용하게 생성하기 위해서는 중첩 격자 기법이 적합
- 중첩격자 (overset)
 - 2개 이상의 격자계를 중첩하여 해석하는 기법
 - 복잡한 형상 및 물체의 운동을 나타내는데 효과적인 기법

6DOF motions



Lobe Blower



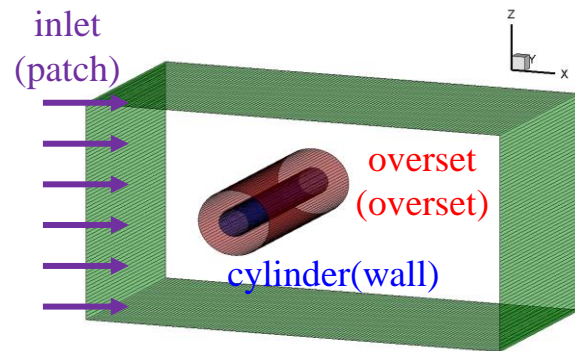
❖ 서론

INDEX

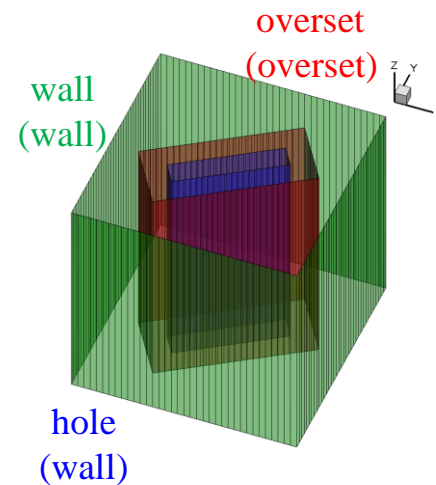
- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

- ESI에서도 OpenFOAM-v1906에서부터 중첩 격자 기능 추가
- OpenFOAM-v2206 overset tutorial (overPimpleDyMFoam)

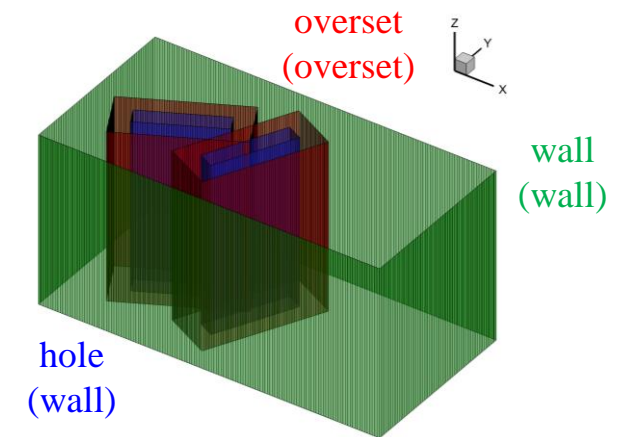
➤ cylinder



➤ simpleRotor



➤ twoSimpleRotor



- OpenFOAM-v2206에서 지원하는 중첩 격자 라이브러리에 대한 분석 진행

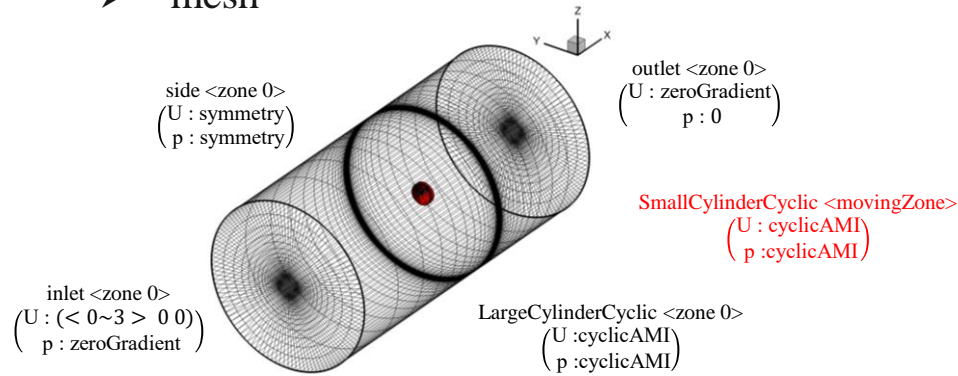
❖ 중첩 격자 해석 인자 평가

INDEX

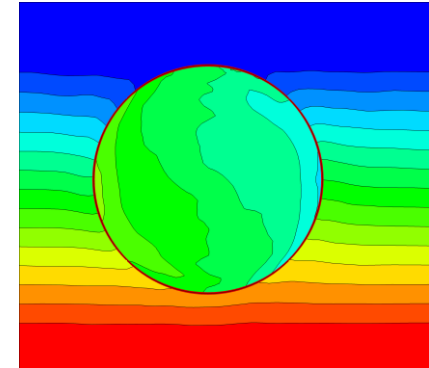
- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

- 아래 2case를 이용하여 중첩 격자 해석이 계산에 미치는 영향 분석
- 유동의 회전이 발생하는 **cyclicAMI**

➤ mesh

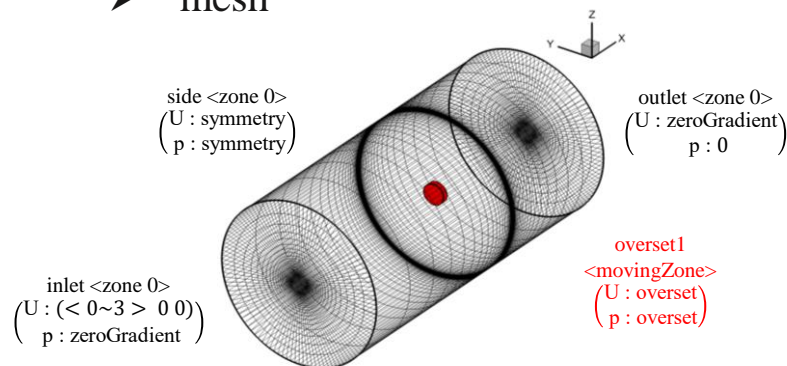


➤ Ux contour (yz plane)

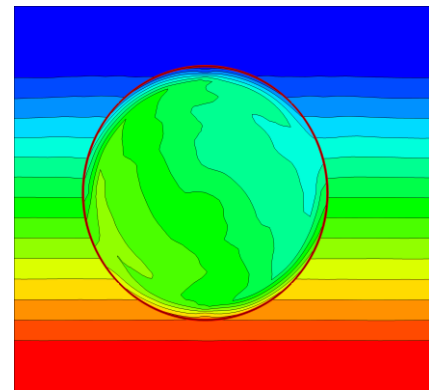


- 위 계산과 동일한 조건의 **overset**

➤ mesh



➤ Ux contour (yz plane)



❖ 중첩 격자 해석 인자 평가

INDEX

- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

- 해석 인자 선정

➤ 유동이 회전하는데 영향을 미치는 해석 인자 선정 (dt, cell size, rps)

➤ dt, cell size, rps에 따른 영향을 보기 위해 아래의 case들을 진행

계산 case (총 120 case)

dt							
0.00025s	0.0005s	0.001s	0.002s	0.004s	0.008s	0.016s	0.032s

cell size		
0.004m (dense)	0.008m (medium)	0.016m (coarse)

rps				
7rps	8rps	9rps	10rps	11rps

- dt, cell size, rps를 변수로 가지는 무차원 수를 이용하여 비교

➤ *dimensionless num* : $\frac{\Delta d}{2\pi r w \times \Delta t}$

Δd : cell size (m)

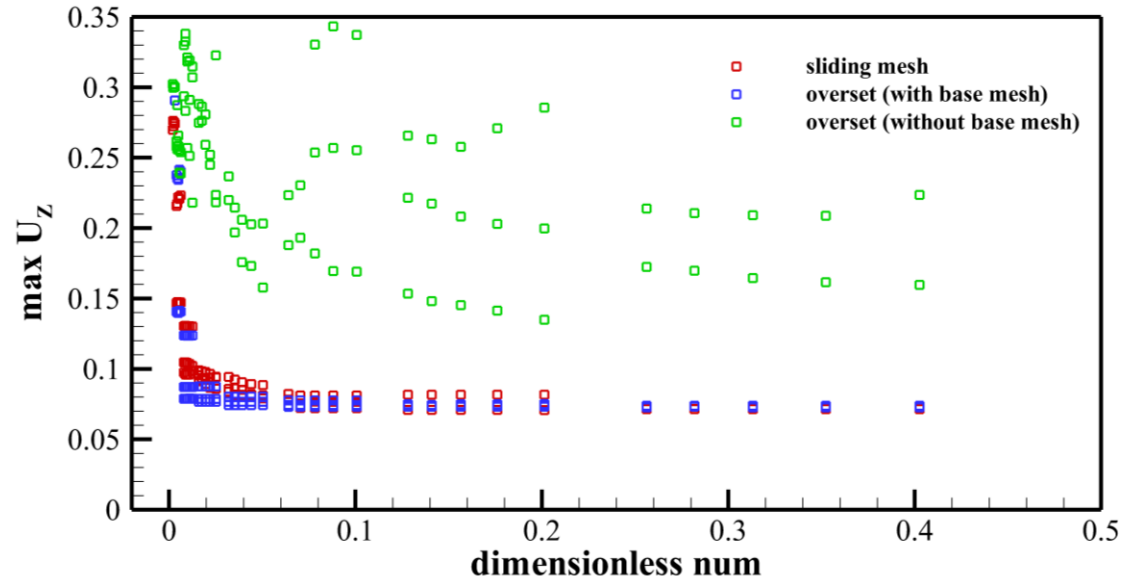
r : radius (m)

w : rps (rad/s)

Δt : dt (s)

❖ 중첩 격자 해석 인자 평가

- sliding mesh와 overset 비교



- dt, cell size, rps에 따른 유동의 회전 정도 비교
- 새로운 무차원수에 따른 $max U_z$

- 전체적으로 base mesh가 있는 overset이 sliding mesh에 비해 좋은 결과를 보이는 것을 확인

INDEX

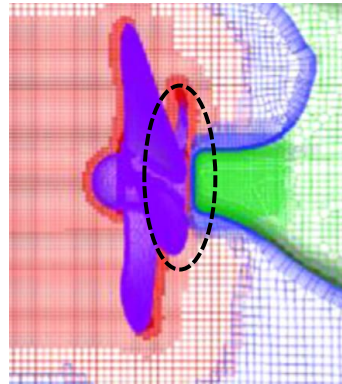
- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

❖ 중첩 격자의 문제점

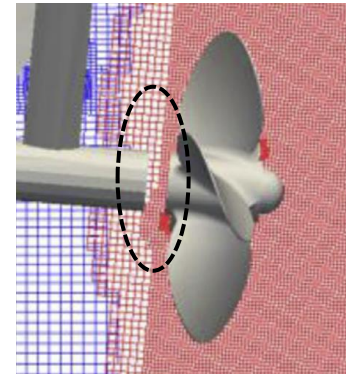
INDEX

- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

- 중첩 격자 계산의 경우 서로 다른 격자계의 경계면이 서로 맞닿아 있거나 가까이 있는 경우 계산에 문제 발생
 - 서로 다른 격자계에 있는 벽끼리 맞닿는 순간 계산에 문제 발생
- 상기 문제는 OpenFOAM에 구현되어 있는 overset library의 구조적 특성



Dynamic overset grids in OpenFOAM with application to KCS self-propulsion and maneuvering



Numerical simulations of zigzag maneuver of free running ship in waves by RANS-Overset grid method

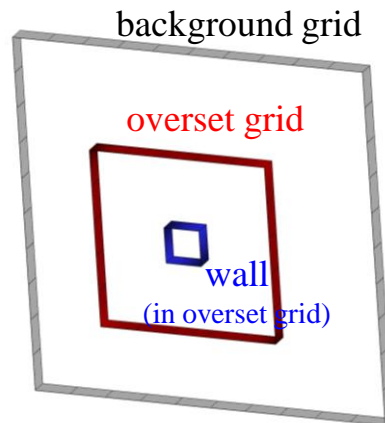
- 이러한 문제가 발생하는 원인 파악 후 개선 여부를 판단하기 위해 overset 코드 분석 필요

❖ overset cell type

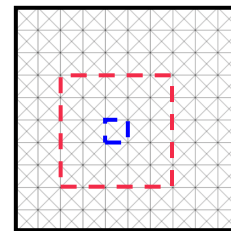
INDEX

- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

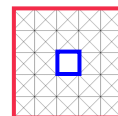
- 중첩 격자 계산은 cell들의 타입을 3가지로 분류한 후 계산 진행
 - interpolated : 중첩된 격자계에서 값을 받아오는 cell
 - hole : 중첩된 격자계에 있는 wall 경계 내부의 cell이며, 계산을 진행하지 않는 cell
 - calculated : interpolated type과 hole type을 제외한 나머지 cell
- example (2D square cylinder)
 - mesh (cell count : 124)



background grid (10x10)

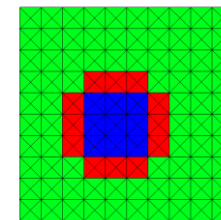


overset grid (5x5)

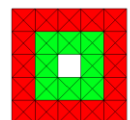


➢ cell type contour

background grid



overset grid



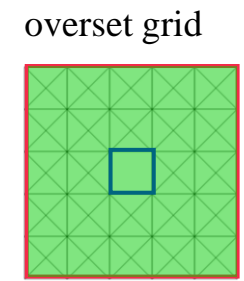
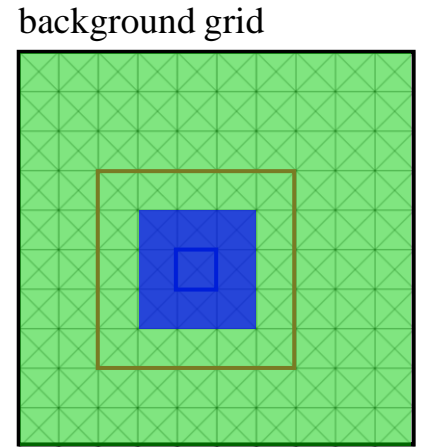
// contour color : interpolated, hole, calculated

❖ overset cell type

INDEX

- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

- overset code algorithm (1) <markPatchesAsHoles>



// contour color : interpolated, hole, calculated

- 중첩된 격자계에서 경계면에 맞닿아있는 cell들을 HOLE 타입으로 설정

inverseDistanceCellCellStencil.C:302

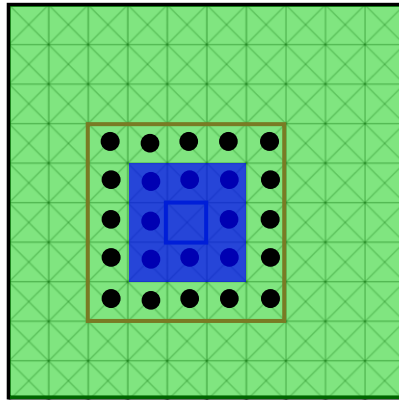
```

302 if (srcPatchBb.overlaps(tgtPatchBb))
303 {
304     const PackedList<2>& srcPatchTypes = patchParts[srcI];
305     const labelVector& zoneDivs = patchDivisions[srcI];
306
307     forAll(tgtCellMap, tgtCelli)
308     {
309         label celli = tgtCellMap[tgtCelli];
310         treeBoundingBox cBb(cellBb(mesh_, celli));
311         cBb.min() -= smallVec_;
312         cBb.max() += smallVec_;
313
314         if
315         (
316             overlaps
317             (
318                 srcPatchBb,
319                 zoneDivs,
320                 srcPatchTypes,
321                 cBb,
322                 patchCellType::PATCH
323             )
324         )
325         {
326             allCellTypes[celli] = HOLE;
327         }
328     }
329 }
    
```

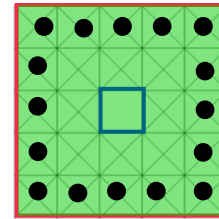
❖ overset cell type

- overset code algorithm (2) <markDonors>

background grid



overset grid



// contour color : interpolated, hole, calculated

- 중첩된 격자들의 stencil(Donor cell)을 설정하는 단계
- cellCenter의 points를 가지고 있는 다른 격자계에서의 cell을 stencil cell로 설정
- searchBox(fvSchemes options) 안에 있는 cell들에 대해 진행
- stencil cell이 없는 경우 그 cell은 stencil cell을 가지지 않음
- stencil cell이 HOLE type인 경우 그 cell은 stencil cell을 가지지 않음

inverseDistanceCellCellStencil.C:473

```
473 // 1. do processor-local src/tgt overlap
474 {
475     labelList tgtToSrcAddr;
476     waveMethod::calculate(tgtMesh, srcMesh, tgtToSrcAddr);
477     forAll(tgtCellMap, tgtCelli)
478     {
```

waveMethod.H:93

```
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             }
113             changedFacesInfo.append(meshToMeshData(tgtCelli));
114             break;
115         }
116     }
```

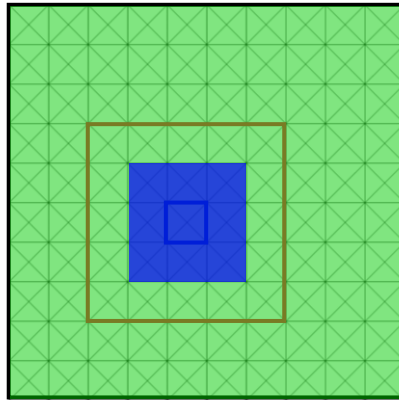
INDEX

- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

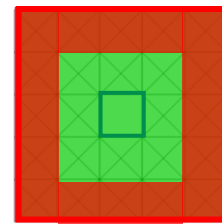
❖ overset cell type

- overset code algorithm (3) < make interpolated type cells >

background grid



overset grid



oversetPatch

// contour color : interpolated, hole, calculated

- overset patch와 맞닿아 있는 cell들의 cell type을 변경
- overset patch와 맞닿아 있는 cell이 stencil을 가지고 있는 경우
→ **interpolated type**
- overset patch와 맞닿아 있지만 cell이 stencil을 가지고 있지 않는 경우
→ **hole type**

inverseDistanceCellCellStencil.C:1648

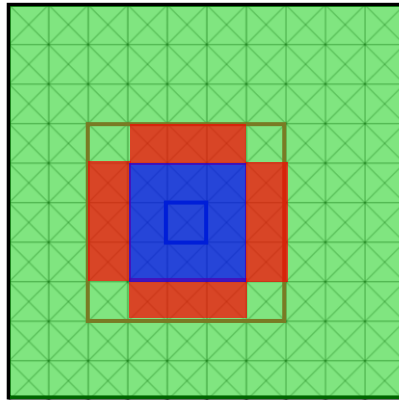
```

1648 // Use the patch types and weights to decide what to do
1649 forAll(allPatchTypes, cellI)
1650 {
1651     if (allCellTypes[cellI] != HOLE)
1652     {
1653         switch (allPatchTypes[cellI])
1654         {
1655             case OVERSET:
1656             {
1657                 // Require interpolation. See if possible.
1658
1659                 if (allStencil[cellI].size())
1660                 {
1661                     allCellTypes[cellI] = INTERPOLATED;
1662                 }
1663                 else
1664                 {
1665                     //allCellTypes[cellI] = CALCULATED;
1666
1667                     allCellTypes[cellI] = HOLE;
1668                 }
1669             }
1670         }
1671     }
1672 }
1673 }
1674 }
1675 }
1676 }
1677 }
1678 }
1679 }
1680 }
    
```

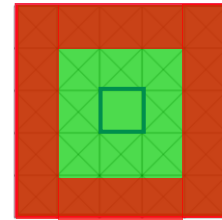
❖ overset cell type

- overset code algorithm (4) < walkFront >

background grid



overset grid



// contour color : interpolated, hole, calculated

- hole type cell 주변에 interpolated type cell layer 생성
- hole type과 맞닿아 있는 cell이 stencil을 가지고 있는 경우
→ **interpolated type**
- hole type과 맞닿아 있는 cell이 stencil을 가지고 있지 않은 경우
→ **hole type**

inverseDistanceCellCellStencil.C:875

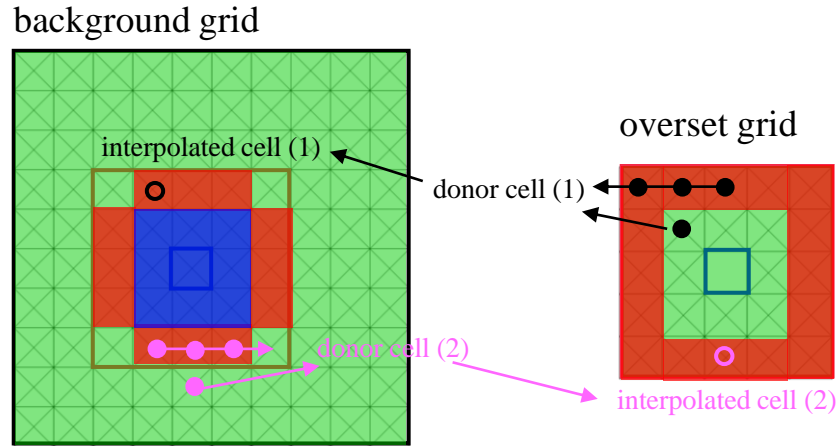
```

875 void Foam::cellCellStencils::inverseDistance::walkFront
876 (
877     const scalar layerRelax,
878     const labelListList& allStencil,
879     labelList& allCellTypes,
880     scalarField& allWeight
881 ) const
882 {
883     // Current front
884     bitSet isFront(mesh_nFaces());
885
886     const fvBoundaryMesh& fvm = mesh_boundary();
887
888     .
889     .
890     .
891     .
892     .
893     .
894     .
895     .
896     .
897     .
898     .
899     .
900     .
901     .
902     .
903     .
904     .
905     .
906     .
907     .
908     .
909     .
910     .
911     .
912     .
913     .
914     .
915     .
916     .
917     .
918     .
919     .
920     .
921     .
922     .
923     .
924     .
925     .
926     .
927     .
928     .
929     .
930     .
931     .
932     .
933     .
934     .
935     .
936     .
937     .
938     .
939     .
940     .
941     .
942     .
943     .
944     .
945     .
946     .
947     .
948     .
949     .
950     .
951     .
952     .
953     .
954     .
955     .
956     .
957     .
958     .
959     .
960     .
961     .
962     .
963     .
964     .
965     .
966     .
967     .
968     .
969     .
970     .
971     .
972     .
973     .
974     .
975     .
976     .
977     .
978     .
979     .
980     .
981     .
982     .
983     .
984     .
985     .
986     .
987     .
988     .
989     .
990     .
991     .
992     .
993     .
994     .
995     .
996     .
997     .
998     .
999     .
1000    .
1001    .
1002    .
1003    .
1004    .
1005    .
1006    .
1007    .
1008    .
1009    .
1010    .
1011    .
1012    .
1013    .
1014    .
1015    .
1016    .
1017    .
1018    .
1019    .
1020    .
1021    .
1022    .
1023    .
1024    .
1025    .
1026    .
1027    .
1028    .
1029    .
1030    .
1031    .
1032    .
1033    .
1034    .
1035    .
1036    .
1037    .
1038    .
1039    .
1040    .
1041    .
1042    .
1043    .
1044    .
1045    .
1046    .
1047    .
1048    .
1049    .
1050    }
1051 }
1052
1053     DAUsyncTools::syncFaceList(mesh_, newIsFront,
orEqOp<unsigned int>());
1054     syncTools::syncFaceList(mesh_, newFraction,
maxEqOp<scalar>());
1055
1056     isFront.transfer(newIsFront);
1057     fraction.transfer(newFraction);
1058 }
1059 }

```

❖ overset cell type

- overset code algorithm (5) < createStencil >



// contour color : interpolated, hole, calculated

- stencil (donor cell) 재 설정
- interpolated cell 이외의 cell들의 stencil (donor cell) 정보는 제거
- interpolated cell의 stencil (donor cell)과 맞닿아있는 모든 cell들을 stencil 로 설정
- stencil (donor cell)로 선정된 cell 중 hole type의 cell은 stencil (donor cell)에서 제거

inverseDistanceCellCellStencil.C:1097

```

1097 void Foam::cellCellStencils::inverseDistance::createStencil
1098 (
1099     const globalIndex& globalCells
1100 )
1101 {
1117     const vector greatPoint(GREAT, GREAT, GREAT);
1118
1119     boolList isValidDonor(mesh_.nCells(), true);
1120     forAll(cellTypes_, celli)
1121     {
1122         .
1123         .
1148         if (slots.size() != 1)
1149         {
1150             FatalErrorInFunction<< "Problem:" << slots
1151                 << abort(FatalError);
1152         }
1153     }
1154     .
1155     .
1273     cellInterpolationMap_.reset
1274     (
1275         new mapDistribute
1276         (
1277             globalCells,
1278             cellStencil_,
1279             compactMap
1280         )
1281     );
1282 }
    
```

❖ overset cell type

INDEX

- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

- 서로 다른 격자계의 경계면이 서로 맞닿아 있는 경우 walkFront 단계에서 hole type cell들이 다른 cell들로 전파가 되어 문제가 발생
- 서로 다른 격자계의 경계면들이 가까이 있는 경우에도 동일한 문제가 발생할 수 있는데 이 경우에는 fvSchemes의 oversetInterpolation의 값으로 문제 회피 가능 (inverseDistance)

문제 발생시 cell type

```
searchBox (-0.160 -0.16 -0.060)(0.160 0.160 0.060)
searchBoxDivisions (10 10 10)
```

background grid



overset grid

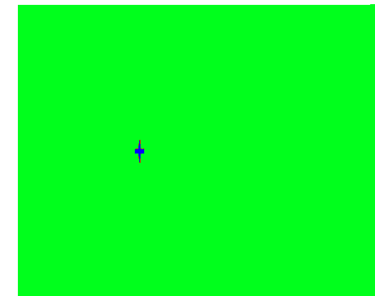


- background 영역의 대부분의 cell들이 hole type으로 설정
- 계산 진행 x

oversetInterpolation 값 조절 후 cell type

```
searchBox (-0.144 -0.144 -0.046)(0.144 0.144 0.046)
searchBoxDivisions (200 200 200)
```

background grid



overset grid



- resolution 값을 늘려 cell type이 정상적으로 적용
- 계산 소요 시간 증가

// contour color : interpolated, hole, calculated

❖ 결론

INDEX

- 서론
- 해석 인자 평가
- 문제점
- overset cell type
- 결론

- OpenFOAM-v2206에서 지원하는 중첩 격자 라이브러리에 대한 분석 진행
- 해석 인자에 따른 영향을 분석했으며, sliding mesh에 비해 base mesh가 있는 overset case가 더 좋은 결과를 보이는 것을 확인
- OpenFOAM의 overset 라이브러리는 서로 다른 격자계의 경계면이 서로 맞닿아 있는 경우에 cell type을 선정하는 알고리즘에 문제가 존재
- 서로 다른 격자계의 경계면이 가까이 있는 경우에도 동일한 문제가 발생할 수 있으며, 이 경우 fvSchemes의 oversetInterpolation 설정 값을 통해 발생하는 문제를 회피 가능