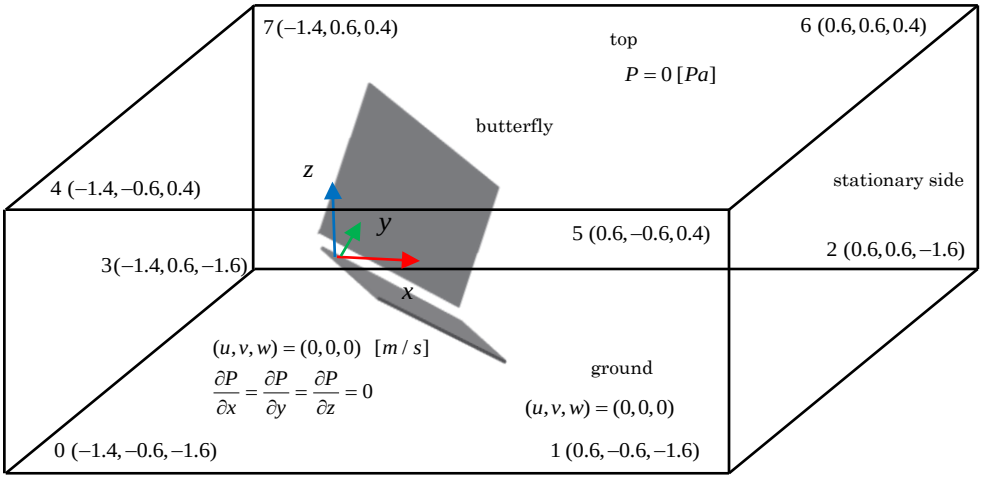
OpenFoam CFD（重合格子移動境界：自由落下）

2021, May. 24 修正

openFoam **v2012** の tutorial のプログラムを書き換えて，重合格子による移動境界流体解析を行う．後述の理由により，重心周りの運動方程式に変更するライブラリを作成する．
流れとしては，
(1) CAD で翅を左右別々に作成して stl
(2) Tutorial の重合格子非圧縮流体のチュートリアルをコピー
(3) 計算条件を書き換え
(4) ライブラリのコンパイル
(5) 実行
以下，計算条件
1 剛体モデル： 翼長 5cm，翼弦 5cm，左右の間隔 0.5cm，反り角 15deg
運動：剛体の蝶の滑空（自由落下）
境界条件は表のとおり：下が地上，それ以外開放



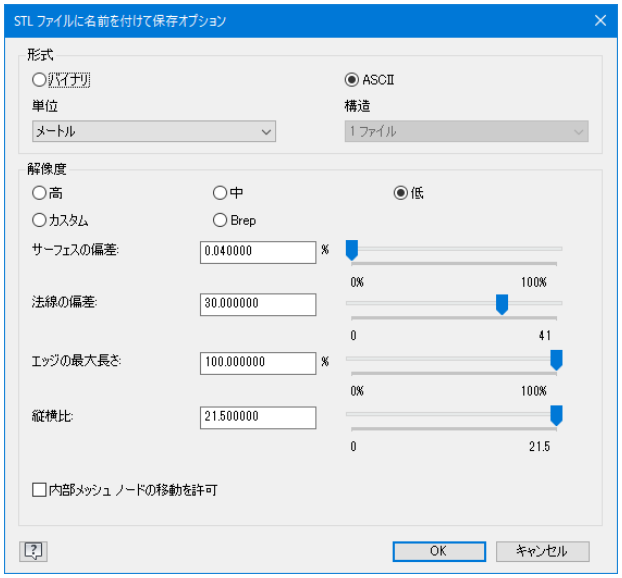
	Type	U 速度	p 圧力	pointDisplacement 運動で与える変位	Zone ID
top	patch	type zeroGradient;	type fixedValue; value 0;	type uniformFixedValue; uniformValue (0 0 0);	type zeroGradient;
ground	patch	type uniformFixedValue; uniformValue (0 0 0);	type zeroGradient;	type uniformFixedValue; uniformValue (0 0 0);	type zeroGradient;
stationarySides	patch?	type zeroGradient;	type zeroGradient;	type uniformFixedValue; uniformValue (0 0 0);	type zeroGradient;
Oversetbutterflt	overset	type overset;	type overset;	patchType overset; type zeroGradient;	type overset;
butterfly	wall ここは patch?	type movingWallVelocity; value uniform (0 0 0);	type zeroGradient;	type calculated;	type zeroGradient;
butterflySides	overset	type overset;	type overset;	patchType overset; type zeroGradient;	type overset;

注：slip は，スカラー値の場合には zeroGradient，ベクトル量の場合には，法線方向が zero,接線方向が zeroGradient とのこと．無限遠にするには天井（top）は低いかも．

1. CAD モデル作成

Inventor で剛体モデルを作成し，stl として保存する．ここでは，”butterfly.stl”とした．座標系はそのまま反映されるので注意．オプションで「ASCII」，単位は「メートル」，解像度は「低」（brep 以外）にしておく．CAD の座標系は，openFoam

の座標系と一致する．



2. 計算条件の変更

2.1 Tutorial から似た計算をコピー

Linux の openFoam のチュートリアルにある twoSimpleRotors のディレクトリ

¥¥wsl$¥Ubuntu-18.04¥opt¥OpenFOAM¥OpenFOAM-v2012¥tutorials¥incompressible¥overPimpleDyMFoam¥twoSimpleRotors

のファイルを Windows の適当なディレクトリ（ここでは，butterfly12）にコピーしながらプログラミングする．これ以降は，windows OS 上での作業になる．ここでは，最終的に，butterfly12 の下に
0.org/
constant/
lib/
ovesetmesh/
system/
Allrun
Allclean
Librun
Libclean
として，いつもの構成で計算する．なお，重合格子は，openfoam ver7 系統では使えない．**v2012 系統のみ利用可能**である．朱書は自分で作る．

2.2 重合格子部分の計算空間の生成

蝶周りの計算空間を作成する．

/oversetmesh/constant/triSurface/

というディレクトリを作成し，CAD ファイル（butterfly.stl）をコピーする．次に，
... ¥butterfly12¥system¥blockMeshDict
を
... ¥butterfly12¥oversetmeshleft¥system¥blockMeshDict
としてコピーし，以下のように書き換える．

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
```

```
scale     1;

vertices
(
      (-0.1 -0.15 -0.1)                          /* 蝶を囲むサイズ．クッションが 10cm ぐらい．  */
      ( 0.2 -0.15 -0.1)
      ( 0.2    0.15 -0.1)
      (-0.1    0.1 -0.1)
      (-0.1 -0.15    0.15)
      ( 0.2 -0.15    0.15)
      ( 0.2    0.15    0.15)
      (-0.1    0.15    0.15)
);

blocks
(
      hex (0 1 2 3 4 5 6 7) butterflyZone (32 32 30) /*  ゾーン名は使わないけど一応つける．メッシュは 1cm 四方    */
      simpleGrading (((0.3 0.25 0.5) (0.4 0.5 1) (0.3 0.25 2)) ((0.3 0.25 0.5) (0.4 0.5 1) (0.3 0.25 2)) ((0.3 0.25 0.5) (0.4 0.5 1) (0.3 0.25
2)))          //  （全体の割合 A，格子数割合 B，格子サイズ変化割合 C）
);

edges
(
);

boundary
(
      butterflySides
      {
            type overset;                 /*  重合境界  */
            faces
            (
                  (0 3 2 1)               // 外側から見て反時計回り
                  (2 6 5 1)
                  (1 5 4 0)
                  (3 7 6 2)
                  (0 4 7 3)
                  (4 5 6 7)
            );
      }
      butterfly                           /*蝶境界*/
      {
            type wall;
            faces ();
      }
);
```

次に，蝶**の CAD モデルをブーリアン演算**するために同じ system に，
. snappyHexMeshDict
を以下の通り作成する．

```
FoamFile
{
      version       2.0;
      format         ascii;
      class          dictionary;
      object         snappyHexMeshDict;
}
```

```
castellatedMesh true;
snap            true;
addLayers       false;          /*  毎度のことで境界層はあきらめる*/

geometry
{
    butterfly                   /*  名前   */
    {
        type    triSurfaceMesh;
        file    "butterfly.stl";    /* CAD ファイル*/
    }
    refinementbox     /*  蝶の周りのメッシュを細かくする範囲  */
    {
        type    searchableBox;
        min (-0.04 -0.1 -0.04);
        max ( 0.10    0.1    0.08);
    }
};

castellatedMeshControls
{

    maxLocalCells 100000;
    maxGlobalCells 2000000;
    minRefinementCells 10;
    nCellsBetweenLevels 2;

    features ();

    refinementSurfaces
    {
        butterfly
        {
            // Surface-wise min and max refinement level
            level (4 4);                    /* 4 回分割．翅周りが 1cm->0.5cm->0.25cm->0.125cm->0.062cm ぐらいになる*/
        }
    }

    // Resolve sharp angles on fridges
    resolveFeatureAngle 30;

    refinementRegions
    {
        refinementbox
        {
           mode inside;
           levels ((1 1));          /** minimam level and maximum level :  レベル 1 は半分にする    **/
        }
    }

    locationInMesh (0.1 0 0);     /*  メッシュ領域の内側指定  */
    allowFreeStandingZoneFaces true;
}

snapControls
{
     nSmoothPatch 3;
     tolerance 1.0;
     nSolveIter 30;
     nRelaxIter 5;
```

```
}

addLayersControls      /* いらないけど一応書く  */
{
    relativeSizes true;

    layers
    {
        butterfly
        {
            nSurfaceLayers 3;
        }
    }
    expansionRatio 1.0;
    finalLayerThickness 0.5;
    minThickness 0.25;
    nGrow 0;
    featureAngle 360;
    nRelaxIter 32;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;

    nLayerIter 50;
}

meshQualityControls
{
    maxNonOrtho 65;

    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
    minVol 1e-13;
    minTetQuality 1e-30;
    minArea -1;
    minTwist 0.05;
    minDeterminant 0.001;
    minFaceWeight 0.05;
    minVolRatio 0.01;
    minTriangleTwist -1;
    nSmoothScale 4;
    errorReduction 0.75;
}

mergeTolerance 1e-6;
```

以下，中身に意味はないが，無いと snappyHexMesh が実行できないので，用意しておくファイル：

．controlDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
```

```
    object      controlDict;
}

application     subsetMesh;     /* 結局使わなかったが，書いてある実行コマンド */
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         6;
deltaT          0.01;
writeControl    adjustable;
writeInterval   0.1;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;
adjustTimeStep  yes;
maxCo           1;
maxAlphaCo      1;
maxDeltaT       1;
```

. fvSchemes

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}

gradSchemes
{
}

divSchemes
{
}

laplacianSchemes
{
}
```

. fvSolution

このファイルなんか，申し訳程度の記載すらない...

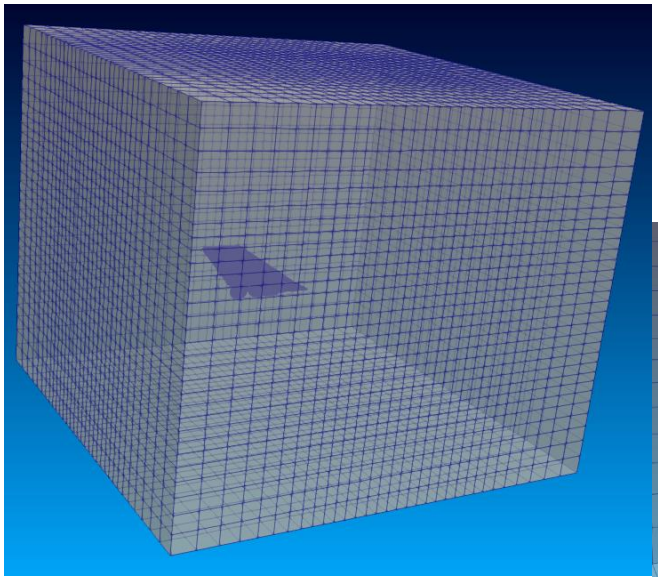
```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
```

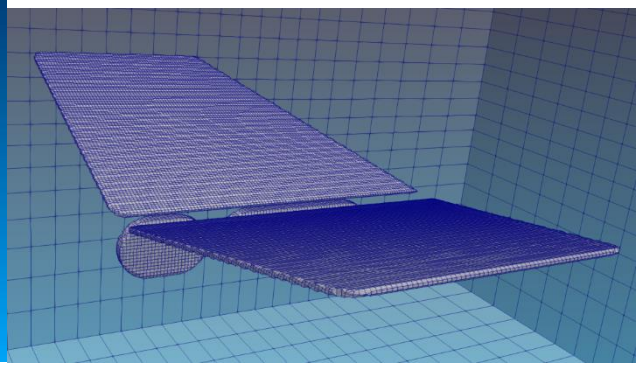
これで，system 内に 5 つのファイルができたことになる．ここで，linux から
...butterfly12¥oversetmeshleft¥

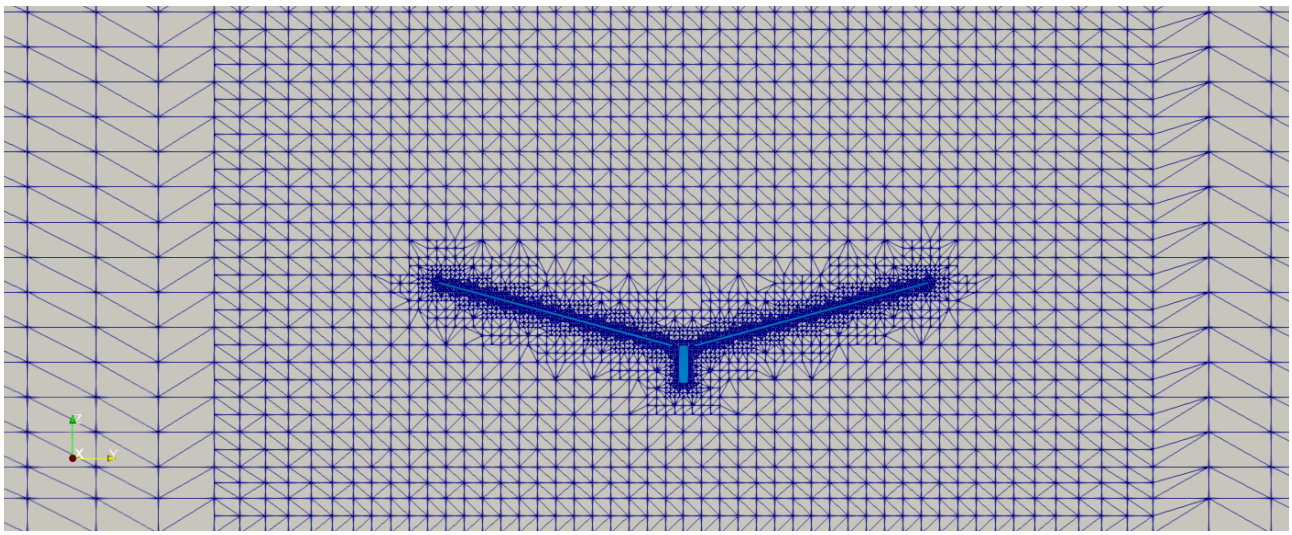
の場所で

> blockMesh
> snappyHexMesh –overwrite

を実行して，paraFoam で確認すると，以下のようなメッシュが作成されている．この計算空間の外側の境界面が，leftwingsides である．なお，windows から直接 paraview を開いて確認するときには，同じディレクトリ内に，***.foam というダミーファイルを作っておく必要がある．paraFoam はこれらの前処理をやっている．







中央が細分化した部分．膜翼厚を 0.5mm にしたので細分化レベルを 4 にしたため，こうなったがやりすぎ．格子が小さすぎるとクーラン数が大きくなり，計算時間が長くなる．

2.3 全体の計算空間の生成

butterfly12¥system
のディレクトリの中身の書き換え．

. blockMeshDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
```

```
}

scale     1;

vertices
(
    (-1.4   -0.6   -1.6)                    /* 下を長めにしてある．滑空軌跡を考慮する */
    ( 0.6   -0.6   -1.6)
    ( 0.6    0.6   -1.6)
    (-1.4    0.6   -1.6)
    (-1.4   -0.6    0.4)
    ( 0.6   -0.6    0.4)
    ( 0.6    0.6    0.4)
    (-1.4    0.6    0.4)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (80 32 80)
    simpleGrading (1 ( (0.3 0.25 0.5) (0.4 0.5 1) (0.3 0.25 2) ) 1)      // 中央部を refineMesh で細分化するため粗め.
);

edges
(
);

boundary
(
    stationarySides                 /* 側面境界：開放条件 */
    {
        type patch;
        faces
        (
            (3 7 6 2)
            (1 5 4 0)
            (0 4 7 3)
            (2 6 5 1)
        );
    }

    ground                          /* 地上 */
    {
        type wall;
        faces
        (
            (0 3 2 1)
        );
    }
    top                             /* 天井は無限遠として大気圧 0Pa にするが，近すぎるかも...   */
    {
        type patch;
        faces
        (
            (4 5 6 7)
        );
    }
);
```

. controlDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}

libs            (overset fvMotionSolvers);          /*これが重合格子ソルバ  */

DebugSwitches
{
    overset                 0;
    dynamicOversetFvMesh    0;
    cellVolumeWeight        0;
}

application     overPimpleDyMFoam;
startFrom       latestTime;     /*  計算開始は前回終了時から  */
startTime       0;
stopAt          endTime;
endTime         1;              /* 1 秒間の落下．計算空間をはみ出なければ長くてもよい*/
deltaT          0.001;          /*  時間刻み．クーラン数依存．  */
writeControl    adjustable;     /*  時間調整有設定  */
writeInterval   0.01;           /*  保存間隔  */
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;              /* 6 桁にしたので，途中で計算を止め，再開すると精度が落ちる．いやなら 12 に*/
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;         /* true or false  の場合と，yes or no  の場合があってよくわからず. . .  */
adjustTimeStep  true;
maxCo           1;              /*  最大クーラン数（速度×時間刻み／メッシュサイズ），本来 1 以下がいい  */
functions
{
   wingforce        /*   翼面荷重計算．postProcessing ディレクトリの下に保存される  */
   {
    type            forces;
    libs            ("libforcesButterflyCOM.so");   // 作成するライブラリ
    patches         (butterfly);

    writeControl        runTime;
    writeInterval       0.01;

    rho         rhoInf;         /*  空気密度．翅反力計算時のみ利用  */
    rhoInf      1.293;

    cofRname        butterflybody;          // モーメントを計算する点を読み込むファイル．dynamicMeshDict にも設定す
る．
    CofR            (0 0 0);    /*  モーメント計算の中心   */
   }
}
```

. fvSchemes

```
FoamFile
{
    version     2.0;
    format      ascii;
```

```
    class       dictionary;
    object      fvSchemes;
}

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss upwind;
//      div(phi,epsilon) Gauss limitedLinear 1;          /*k-delta であるが，この後層流設定するので消す*/
//      div(phi,k)       Gauss limitedLinear 1;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
    laplacian(diffusivity,cellDisplacement)    Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

oversetInterpolation
{
    method          cellVolumeWeight;
    // Faster but less accurate
    //method                trackingInverseDistance;
    //searchBox             (0 0 0)(0.02 0.01 0.01);
    //searchBoxDivisions    3{(64 64 1)};
}

fluxRequired
{
    default         no;
    pcorr           ;
    p               ;
}

oversetInterpolationSuppressed
{
    grad(p);
    surfaceIntegrate(phiHbyA);
}
```

. fvSolution

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}

solvers
{
    cellDisplacement
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
        maxIter         100;
    }

    p
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-8;
        relTol          0.01;
    }

    pFinal
    {
        $p;
        relTol          0;
    }

    pcorr
    {
        $p;
        solver          PCG;
        preconditioner  DIC;
    }

    pcorrFinal
    {
        $pcorr;
        relTol          0;
    }

//      "(U|k|epsilon)"   /* この後層流設定にするので k- ε は消しておく  */
    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-6;
        relTol          0;
    }

//      "(U|k|epsilon)Final"
    UFinal
```

```
    {
        $U;
        tolerance       1e-6;
        relTol          0;
    }
}

PIMPLE
{
    momentumPredictor   false;
    correctPhi          no;
    nOuterCorrectors    1;
    nCorrectors         3;
    nNonOrthogonalCorrectors 0;
    ddtCorr             true;

    pRefPoint           (0.0001 0.0001 0.001);
    pRefValue           0.0;
}

relaxationFactors
{
    fields
    {
    }
    equations
    {
        ".*"            1;
    }
}
```

. setFieldDict
どの重合格子領域かを区別するフラグ

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      setFieldsDict;
}

defaultFieldValues
(
    volScalarFieldValue zoneID 123          /* どれでもない適当な初期化数字．以下の処理でなくなる */
);

regions
(
    // Set cell values
    // (does zerogradient on boundaries)
    cellToCell
    {
        set butterflycell;          /* 左翅の計算領域．topoSet で設定する．0 番にする */

        fieldValues
        (
            volScalarFieldValue zoneID 0
        );
    }
```

```
    cellToCell
    {
        set calspacecell;           /* 全体の計算領域．topoSet で設定する．1 番にする */

        fieldValues
        (
            volScalarFieldValue zoneID 1
        );
    }
);
```

. topoSetDict
二つ計算領域をセルの領域として設定する．計算空間を合成した後に，設定する仕様のため，面倒くさくなっている．

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      topoSetDict;
}

actions
(
    {
        name      calspacecell;  /* 計算セル空間の名前 */
        type      cellSet;       /* 対象はセル */
        action    new;           /* 新規作成 */
        source    regionsToCell; /* 座標空間からセル領域に */
        insidePoints             /* 対象領域内部指定 */
        (
            (0 0 -0.5)                  /* 重合格子領域の重なっていない場所を指定 */
        );
    }

    {
        name      butterflycell;        /* 蝶セル領域の名前 */
        type      cellSet;
        action    new;
        source    cellToCell;           /* セル領域からセル領域 */
        set       calspacecell;         /* copy：　この時点では butterflycell と calspacecell は同じセル領域 */
    }
    {
        name      butterflycell;
        type      cellSet;
        action    invert;                       /* 反転：計算領域以外の領域に変換．よって，重合格子領域 */
    }
);
```

この指定は，paraview では，zoneID として確認できる．三種類の zone(0,1,2)が色分けされるはずである．また，このファイルは，
butterfly12¥constant¥polyMesh¥sets
のディレクトリに，
calspacecell
butterflycell
ファイルとして保存されている．

2.4 物性および運動の設定（constant ディレクトリ）
. dynamicMeshDict

CAD の iproperty の内容と一致するように幾何情報を入力する．

なお，重心位置や，比重などの変更は，CAD 上では行わず，この辞書ファイルで行う．

(1) 質量の変更

CAD データに対し，質量を 1/2（比重を 1/2）にしたい場合，

．質量（mass）を 1/2 にする．

．慣性テンソル（inertia）の全ての値を 1/2 にする．慣性モーメント mr^2 ＝（質量）×（距離）^2 に基づいている．

(2) 重心を移動させる場合

CAD データに対し，重心を（a, b, c）だけ移動したい場合，

．慣性テンソル（inertia=(Ixx Ixy Ixz　Iyy Iyz　Izz)）を平行軸の定理により以下のように移動させる．

$$Ixx -> Ixx + m(b^2 + c^2)$$

$$Iyy -> Iyy + m(a^2 + c^2)$$

$$Izz -> Izz + m(a^2 + b^2)$$

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      dynamicMeshDict;
}

motionSolverLibs    (rigidBodyMeshMotionButterflyCOM);      // 作成するライブラリ

dynamicFvMesh       dynamicOversetFvMesh;

motionSolver        rigidBodyMotion;

report              false;
rho rhoInf;                                 // このソルバでは必要．気液二相流では不要．
rhoInf 1.293;
g           (0 0 -9.81);                    // このソルバでは必要．気液二相流では不要．
```

```
solver
{
    type NewmarkCOM;          // 重心周りの運動方程式の NMB を設定
    gamma    0.75;            // Velocity integration coefficient:  "gamma > 0.5" is unconditionary stable, but low accuracy.
    beta     0.390625;    // Position integration coefficient:        beta = (gamma+0.5)^2/4
}
Iteration_number_for_MB        1;          // 構造計算の繰り返し回数．1 剛体なので 1 で OK.

OutputFiles            (butterflybody);      // 剛体の重心を保存するファイル名

bodies
{
    butterflybody                          // 剛体名
    {
        type              rigidBody;       // 剛体
        parent            root;            // 参照座標系は基準座標系
        mass               0.0005788;        // 質量．0.002894, rho = 1000 kg/m^3←これは CAD の値．ここでは，比重を
                                             1/5 にしている．
        inertia          (0.438e-6 0 0   2.01e-7 0   0.510e-6);    // 重心周りの慣性テンソル (1.93e-6 0 0   4.84e-7 0
                                                                    2.26e-6)．上記，平行軸の定理でシフトさせている．
        centreOfMass     (0.01 0 -0.005); // 重心．回転中心と一致させる．重心を(0.0101, 0, -0.0089)移動させている．
        transform        (1 0 0    0 1 0    0 0 1) $centreOfMass;          // 参照座標系からみた物体座標系の位置姿勢．
        joint          // COM 運動方程式を選択したので，必ず 6Dof かつ Pxyz, Rxyz の順番
          {
            type                composite;
            joints (
                {  type Pxyz; }
                {  type Rxyz;        }    );
        }
        patches            (butterfly);     // この剛体の流体力を計算する境界
        innerDistance    100;               // これの内側の格子が剛体と共に並進する
        outerDistance    101;               // これの内側の格子が変形する
    }
}

restraints
{
}
```

. transportProperties

```
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       transportProperties;
}

transportModel  Newtonian;      /* ニュートン流体指定   */

nu                  nu [ 0 2 -1 0 0 0 0 ] 1e-05;          /* 空気の動粘性係数．  ν＝μ／ρ．これもザックリなので注意  */
//rho               rho [1 -3 0 0 0 0 0 ] 1.293;          使っていないようだ.
```

.turbulenceProperties

```
FoamFile
{
    version      2.0;
```

```
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}

simulationType  laminar;        /* 層流モデルに設定*/
```

2.5 境界条件の設定（0.org ディレクトリ）

ソルバが，初期境界条件の中身を書き換えたりする可能性があるときには，0.org を作っておき，0 としてコピーしてから計算を開始する．なお，蝶に大きな初速度がある場合には，simpleFoam などのソルバで予め定常計算して用意しておく．

. p

```
/*  圧力境界  */

FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    #includeEtc "caseDicts/setConstraintTypes"

    "(stationarySides|ground)"
    {
        type            zeroGradient;
    }

    top
    {
        type            fixedValue;     /* 天井に大気圧 0Pa を設定．0.4m が無限遠設定は近すぎるかも．  */
        value           $internalField;
    }

    butterfly
    {
        type            zeroGradient;
    }

    butterflySides    // 重合格子の境界
    {
        type            overset;
    }
}
```

. U

```
/*  速度境界  */

FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
```

```
}

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    #includeEtc "caseDicts/setConstraintTypes"

    "(stationarySides|top)"
    {
        type            zeroGradient;
    }

    ground
    {
        type            uniformFixedValue;      /* 地面は壁 */
        uniformValue    (0 0 0);
    }

    butterfly
    {
        type            movingWallVelocity;
        value           uniform (0 0 0);
    }

    butterflySides// 重合格子の境界
    {
        type            overset;
    }
}
```

. pointDisplacement /* 移動境界用の条件 */

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       pointVectorField;
    object      pointDisplacement;
}

dimensions      [0 1 0 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    ".*"            // この記号は，「全て」という意味であるが，そのあとの指定で上書きされるのだろうか？
    {
        type            uniformFixedValue;
        uniformValue    (0 0 0);
    }

    butterfly
    {
        type            calculated;     /*  剛体ソルバで計算された値が入力される */
    }
```

```
    butterflySides// 重合格子の境界
    {
        patchType           overset;
        type                zeroGradient;
    }
}
```

. zoneID

```
FoamFile
{
    version         2.0;
    format          ascii;
    class           volScalarField;
    object          zoneID;
}

dimensions          [0 0 0 0 0 0 0];

internalField     uniform 0;

boundaryField
{
    #includeEtc "caseDicts/setConstraintTypes"

    butterflySides
    {
        type                overset;
    }

    ".*"
    {
        type                zeroGradient;
    }
}
```

2.6 ライブラリの作成

rigidBodyDynamcis 系クラスは，物体座標系の原点（回転中心）に基づいて運動方程式を立てているが，これが自由運動に対応していないようなので，重心周りの運動方程式を立てなおす．詳細は，flyingfish16 以降，もしくは，「剛体の落下」の資料を参照のこと．ここでは，以下のコマンドでライブラリ（***ButterfltCOM）を作成しておく．

- Libclean

で過去のライブラリを消しておく．

- Librun

でコンパイルし，ライブラリを作成しておく．$(FOAM_USER_LIBBIN)で指定された場所に，

libforcesButterflyCOM
librigidBodyDynamicsButterflyCOM
librigidBodyMeshMotionButterflyCOM

が作成されている．

2.7 Linux での計算

コマンドラインからの作業を自動で行うため，以下のテキストファイルを作成する．
. Allrun

```
. ${WM_PROJECT_DIR:?}/bin/tools/RunFunctions        /* restore0Dir コマンドを実行するためのパス設定   */
```

```
cd oversetmesh                          /*  蝶計算領域の作成    */
blockMesh
snappyHexMesh -overwrite

cd ..
blockMesh                               /*  全体計算領域の作成    */
topoSet -dict ./system/topoSetDict2             //  計算空間の中央部を細分化する
refineMesh –overwrite
mergeMeshes . ./oversetmesh –overwrite                  /*  右翅計算領域との合成  */
topoSet                                                 /*  二つ領域のセルの名前を付ける  */
restore0Dir                                             /* 0 ディレクトリのコピー*/
setFields                                               /*  二つ領域のナンバリング  */
overPimpleDyMFoam > pimple.log          /*  流体計算ソルバを並列実行    */
```

以上より，上記で paraview で確認するために作成したファイルを一旦消してから Allrun を実行する．
重合格子を使う場合，重なり合っている境界近傍の格子のサイズは同じぐらいでないと流れを正しく伝えられない．1 秒計算するのに，並列計算なしで 6 時間ぐらいかかった．

蝶にかかる力とモーメントは，以下のファイルに保存されている．全体にかかる力ベクトルの他に，面の法線方向の力（圧力による力）と，接線方向の力（粘性による力）も保存されている．

¥butterfly12¥postProcessing¥wingforce¥0¥force.dat
¥butterfly12¥postProcessing¥wingforce¥0¥moment.dat

```
# Force
# CofR          : (0.000000e+00 0.000000e+00 0.000000e+00)      //  指定した COM の位置になる．
#
# Time                  (total_x total_y total_z)               (pressure_x pressure_y pressure_z)               (viscous_x          viscous_y
viscous_z)
0.00125    (-1.017245e-07 7.239888e-12 2.096161e-04)    (-1.042995e-07 7.651304e-12 2.089465e-04)  (2.574990e-09    -4.114159e-
13 6.695709e-07)
0.00270833    (-2.496131e-07 -2.618675e-12 5.879687e-04)            (-2.880706e-07 -1.258239e-12 5.852565e-04) (3.845750e-08   -
1.360436e-12 2.712244e-06)
0.00453125    (-1.669014e-07 -5.605878e-11 7.050241e-04)            (-3.709558e-07 -5.528058e-11 6.996710e-04) (2.040544e-07   -
7.782046e-13 5.353098e-06)
. . . .
```

なお，この値いには，重力（マルチボディの場合には剛体どうしの反力）は含まれていない．よって，実際の運動は，この流体力に，さらに重力，物体反力を加えた外力によって計算される．

これ以外にも，設定したボディの剛体の重心位置，姿勢が以下に保存されている．

¥butterfly12¥butterflybody.dat