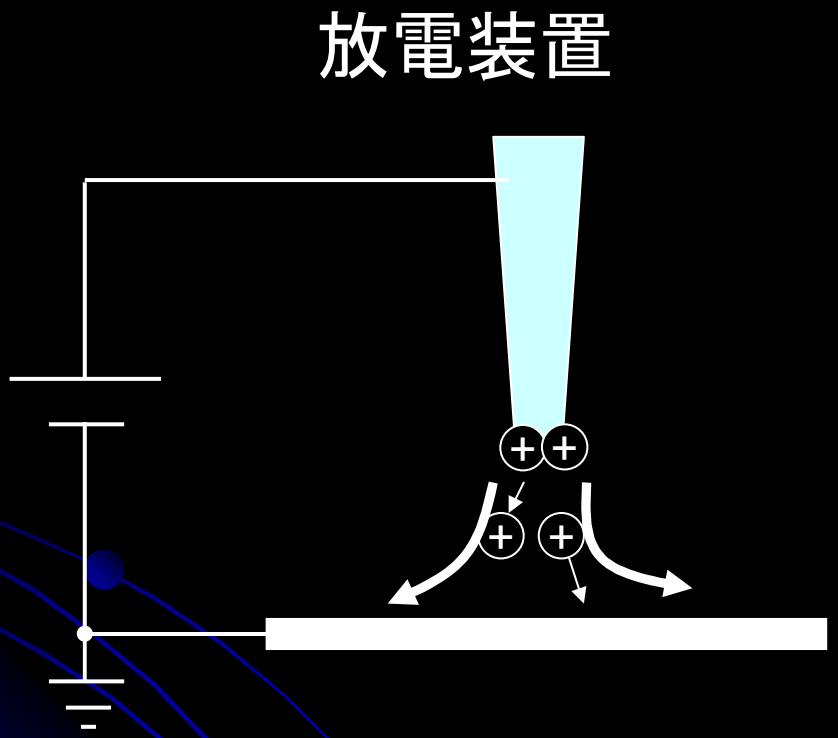
OpenFOAMを利用した
電場-流体の連成ソルバー作成
electrostaticFoam+icoFoam

発表者: K. M

電気流体力学現象について

イオン風の原理



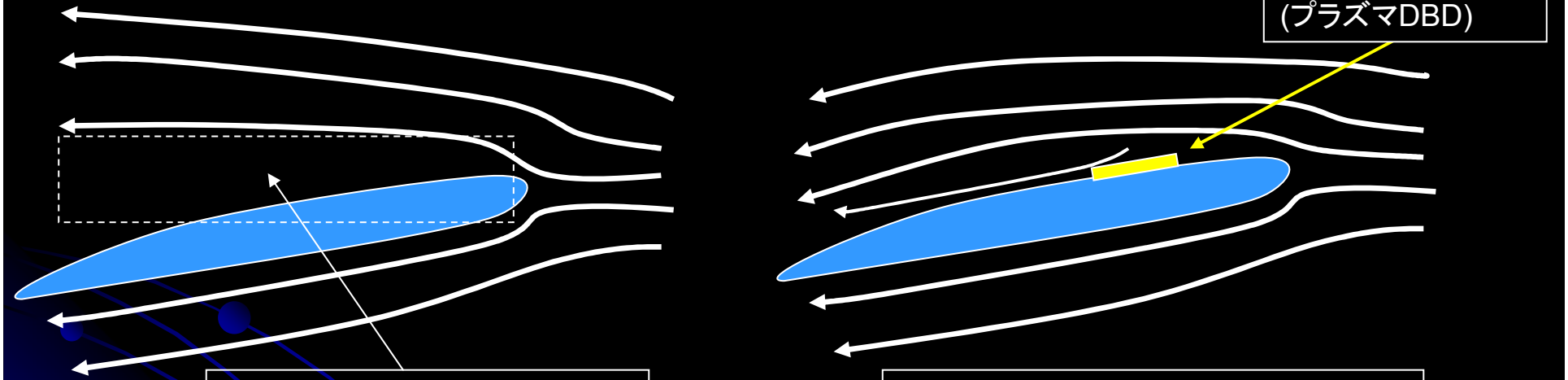
クーロン力で加速された荷電粒子が、空気中の分子を押し出して流れを生じる。

イオン風の工学的応用

プラズマアクチュエーターによる流れの剥離制御

流れ場(イオン風なし)

流れ場(イオン風あり)



流れが剥離
⇒失速の原因

イオン風によるアシストで剥離が減少

電場-流体の連成ソルバー基礎方程式

electrostaticFoamの基礎方程式

$$\nabla^2\phi = -\frac{\rho}{\varepsilon_0}$$

$$\frac{\partial\rho}{\partial t} + \nabla\cdot\mathbf{j} = 0$$

$$\mathbf{j} = \rho\left(-k\nabla\phi\right)$$

ϕ : 静電ポテンシャル

ρ : 電荷密度

ε_0 : 真空の誘電率

$\mathbf{j}$: イオンのドリフト電流

k : イオン移動度

icoFoamの基礎方程式

$$\frac{\partial\mathbf{U}}{\partial t} + \mathbf{U}\cdot\nabla\mathbf{U} = -\nabla\left(\frac{p}{\rho_{air}}\right) + \nu\nabla^2\mathbf{U}$$

$$\nabla\cdot\mathbf{U} = 0$$

ν : 動粘度

$\mathbf{U}$: 流速

ρ_{air} : 空気の密度

電場-流体連成の基礎方程式

$$\nabla^2\phi = -\frac{\rho}{\varepsilon_0}$$

$$\frac{\partial\rho}{\partial t} + \nabla\cdot\mathbf{j} = 0$$

$$\mathbf{j} = \rho\left(-k\nabla\phi + \mathbf{U}\right)$$

対流項

＋

$$\frac{\partial\mathbf{U}}{\partial t} + \mathbf{U}\cdot\nabla\mathbf{U} = -\nabla\left(\frac{p}{\rho_{air}}\right) + \nu\nabla^2\mathbf{U} - \frac{\rho\nabla\phi}{\rho_{air}}$$

$$\nabla\cdot\mathbf{U} = 0$$

クーロン力

電場-流体解析ソルバー作成(1)

ソルバー作成の指針

- あまり多くの変更を行わない(既存ソルバーの最大限活用)
- とりあえず動くことを目標に(シンプルなアルゴリズムを採用)

計算アルゴリズム

対流項を持つ電場ソルバー

$$\nabla^2 \phi = -\frac{\rho}{\varepsilon_0}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{j} = 0$$

$$\mathbf{j} = \rho\left(-k\nabla\phi + \mathbf{U}\right)$$

クーロン力を含む流体ソルバー

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U}\cdot\nabla\mathbf{U} = -\nabla\left(\frac{p}{\rho_{air}}\right) + \nu\nabla^2\mathbf{U} - \frac{\rho\nabla\phi}{\rho_{air}}$$

$$\nabla \cdot \mathbf{U} = 0$$

収束

電場-流体連成の定常解

電場-流体解析ソルバー作成(2)

ソルバー設計

対流項を持つ電場ソルバー

弱連成

クーロン力を含む流体ソルバー

収束

電場-流体連成の定常解

$$\nabla^2\phi = -\frac{\rho}{\varepsilon_0}$$

$$\frac{\partial\rho}{\partial t} + \nabla\cdot\mathbf{j} = 0$$

$$\mathbf{j} = \rho\left(-k\nabla\phi + \mathbf{U}\right)$$

convective_esFoam

$$\frac{\partial\mathbf{U}}{\partial t} + \mathbf{U}\cdot\nabla\mathbf{U} = -\nabla\left(\frac{p}{\rho_{air}}\right) + \nu\nabla^2\mathbf{U} - \frac{\rho\nabla\phi}{\rho_{air}}$$

$$\nabla\cdot\mathbf{U} = 0$$

ico_esFoam

やるべき作業

- 2つのソルバー(convective_esFoam,ico_esFoam)作成
- ソルバーのテスト

流体ソルバーconvective_esFoam作成(1)

-コード作成の準備-

ディレクトリ作成と必要ファイルのコピー

```
$cd  OpenFOAM/OpenFOAM-1.7.1/applications/solvers
$mkdir user_solvers
$cd user_solvers
$mkdir convective_esFoam
$cp –r ../electromagnetics/electrostaticFoam/*  convective_esFoam
$cd convective_esFoam
```

ファイルの編集(Make/files)

オリジナル

electrostaticFoam.C
EXE = $(FOAM_APPBIN)/electrostaticFoam

編集後

electrostaticFoam.C
EXE = $(FOAM_APPBIN)/convective_esFoam

環境変数FOAM_APPBINの設定も忘れずに!

```
$export FOAM_APPBIN=OpenFOAM/OpenFOAM-1.7.1/applications/bin
```

電場ソルバーconvective_esFoam作成(2)

-ソルバー主要部分-

convective_esFoam

$$\nabla^2\phi = -\frac{\rho}{\varepsilon_0}$$

$$\frac{\partial\rho}{\partial t} + \nabla\cdot\mathbf{j} = 0$$

$$\mathbf{j} = \rho\left(-k\nabla\phi + \mathbf{U}\right)$$

改変後

electrostaticFoam.c

```
solve
    (
        fvm::laplacian(esphi) + rho/epsilon0
    );

rhoFlux = -k*mesh.magSf()*fvc::snGrad(esphi);
 solve
    (
        fvm::ddt(rho) + fvm::div(rhoFlux, rho)
      + fvm::div(phi,rho)
    );
```

electrostaticFoam

$$\nabla^2\phi = -\frac{\rho}{\varepsilon_0}$$

$$\frac{\partial\rho}{\partial t} + \nabla\cdot\mathbf{j} = 0$$

$$\mathbf{j} = \rho\left(-k\nabla\phi\right)$$

electrostaticFoam.c

```
solve
    (
        fvm::laplacian(phi) + rho/epsilon0
    );

rhoFlux = -k*mesh.magSf()*fvc::snGrad(phi);
 solve
    (
        fvm::ddt(rho) + fvm::div(rhoFlux, rho)
    );
```

対流項は、div(phi,U)などと表現する(ex. icoFoam.c)。
変数名phiがポテンシャルphiと重複するので改名する

電場ソルバーconvective_esFoam作成(3)

-変数設定-

新しく定義した変数 → createFields.Hにて定義する

流速U:　対流項に登場
静電ポテンシャルesphi:　phiだったが対流項と重複のため改名
対流項を計算する際に必要な変数phi:
rhoFlux: phi改名に伴って編集が必要

createFields.H

```
...
  Info<< "Reading field esphi¥n" << endl;
    volScalarField esphi
    (
        IOobject
        (
            "esphi",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );

...
#   include "createPhi.H"
```

← 静電ポテンシャルesphi追加
対流項のためcreatePhi.H追加

電場ソルバーconvective_esFoam作成(4)

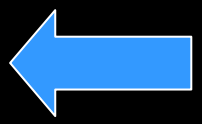
-変数設定-

createFields.H編集

流速U:　対流項に登場
静電ポテンシャルesphi:　phiだったが対流項と重複のため改名
対流項を計算する際に必要な変数phi:
rhoFlux: phi改名に伴って編集が必要

createFields.H(続き)

```
…
 Info<< "Calculating field rhoFlux¥n" << endl;
    surfaceScalarField rhoFlux
    (
        IOobject
        (
            "rhoFlux",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        -k*mesh.magSf()*fvc::snGrad(esphi)
    );
…
```



静電ポテンシャルをesphiに改名

電場ソルバーconvective_esFoam作成(5)

-コンパイル-

Compileの実施

ソースの編集は終わったので、あとはcompileすればいいだけ
コンパイルにはwmakeコマンドを使う.

```
$wmake
```

コンパイルが終了すると、対流項を持つ
静電ソルバーconvective_esFoamが作成される

```
$cd
$cd OpenFOAM/OpenFOAM-1.7.1/applications/bin
$ls
```

ソルバーconvective_esFoamがあることを確認する.
以上で、covective_esFoam作成は終了

電場ソルバーico_esFoam作成(1)

-コード作成の準備-

ディレクトリ作成と必要ファイルのコピー

```
$cd  OpenFOAM/OpenFOAM-1.7.1/applications/solvers/user_solvers
$mkdir ico_esFoam
$cp –r ../incompressible/icoFoam/*  ico_esFoam
$cd ico_esFoam
```

ファイルの編集(Make/files)

オリジナル

icoFoam.C
EXE = $(FOAM_APPBIN)/icoFoam

編集後

icoFoam.C
EXE = $(FOAM_APPBIN)/ico_esFoam

環境変数FOAM_APPBINの設定も忘れずに!

```
$export FOAM_APPBIN=OpenFOAM/OpenFOAM-1.7.1/applications/bin
```

電場ソルバーico_esFoam作成(2)

-ソルバー主要部分-

ico_esFoam

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{U} = -\nabla \left(\frac{p}{\rho_{air}} \right) + \nu \nabla^2 \mathbf{U} - \frac{\rho \nabla \phi}{\rho_{air}}$$

$$\nabla \cdot \mathbf{U} = 0$$

icoFoam

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{U} = -\nabla \left(\frac{p}{\rho_{air}} \right) + \nu \nabla^2 \mathbf{U}$$

$$\nabla \cdot \mathbf{U} = 0$$

改変後

icoFoam.c

```
volVectorField qE =-rho/denMedium)*fvc::grad(esphi);
    fvVectorMatrix UEqn
    (
        fvm::ddt(U)
      + fvm::div(phi, U)
      - fvm::laplacian(nu, U)
        -qE          // with coulomb force
    );
    solve(UEqn == -fvc::grad(p));
```

icoFoam.c

```
fvVectorMatrix UEqn
    (
        fvm::ddt(U)
      + fvm::div(phi, U)
      - fvm::laplacian(nu, U)
    );
  solve(UEqn == -fvc::grad(p));
```

クーロン力項を加えている.空気密度(=denMedium)を入れて設定している.

電場ソルバーico_esFoam作成(3)

-変数設定-

新しく定義した変数 → createFields.Hにて定義する

静電ポテンシャルesphi:　phiだったが対流項と重複のため改名
空気の密度denMedium:

createFields.H

```
…
 dimensionedScalar denMedium
    (
        physicalProperties.lookup("denMedium")
 Info<< "Reading field esphi¥n" << endl;
    volScalarField esphi
    (
        IOobject
        (
            "esphi",
            runTime.timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    );
…
```

← 空気密度denMedium
静電ポテンシャルesphi

電場ソルバー ico_esFoam作成(4)

-コンパイル-

Compileの実施

ソースの編集は終わったので、あとはcompileすればいいだけ
コンパイルにはwmakeコマンドを使う.

```
$wmake
```

コンパイルが終了すると、対流項を持つ
静電ソルバーico_esFoamが作成される

```
$cd
$cd OpenFOAM/OpenFOAM-1.7.1/applications/bin
$ls
```

ソルバーico_esFoamがあることを確認する.

まとめ+コメント

-言わなかったことや言い訳したいこと-

- 偶然できたように装っていますが、一応、Programmer's Guideを読みながら作っています.
- 細かいところで嵌るところがあります.
  変数のタイプ(Surface,Volume)とdiv演算子、laplacian演算子の作用のさせ方.

前回のまとめ

電場-流体の連成ソルバー(弱連成)の作成が完了

ソルバー設計

対流項を持つ電場ソルバー

弱連成

クーロン力を含む流体ソルバー

収束

電場-流体連成の定常解

$$\nabla^2\phi = -\frac{\rho}{\varepsilon_0}$$

$$\frac{\partial\rho}{\partial t} + \nabla\cdot\mathbf{j} = 0$$

$$\mathbf{j} = \rho\left(-k\nabla\phi + \mathbf{U}\right)$$

convective_esFoam

作成完了!

$$\frac{\partial\mathbf{U}}{\partial t} + \mathbf{U}\cdot\nabla\mathbf{U} = -\nabla\left(\frac{p}{\rho_{air}}\right) + \nu\nabla^2\mathbf{U} - \frac{\rho\nabla\phi}{\rho_{air}}$$

$$\nabla\cdot\mathbf{U} = 0$$

ico_esFoam

作成完了!

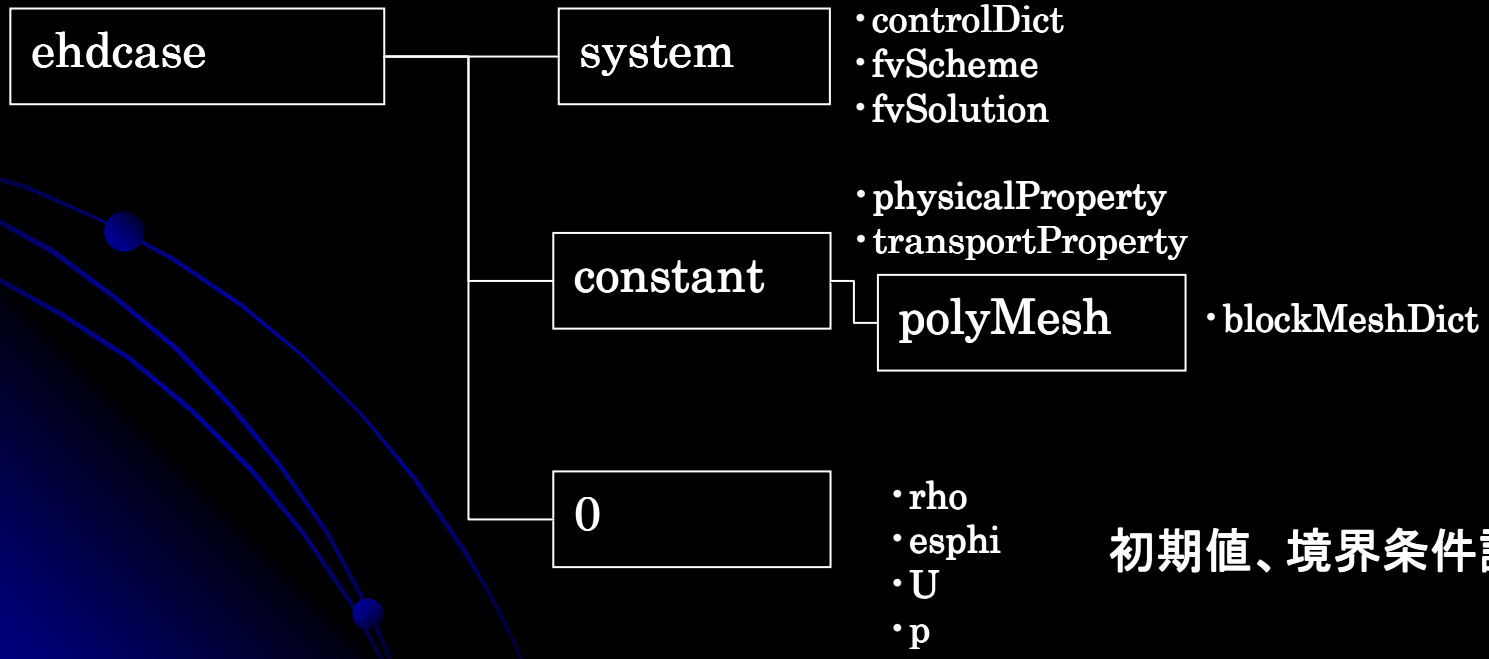
今回やること

テストケースを作成し、自作ソルバーの動作を確認する

イオン風解析テストケースの作成

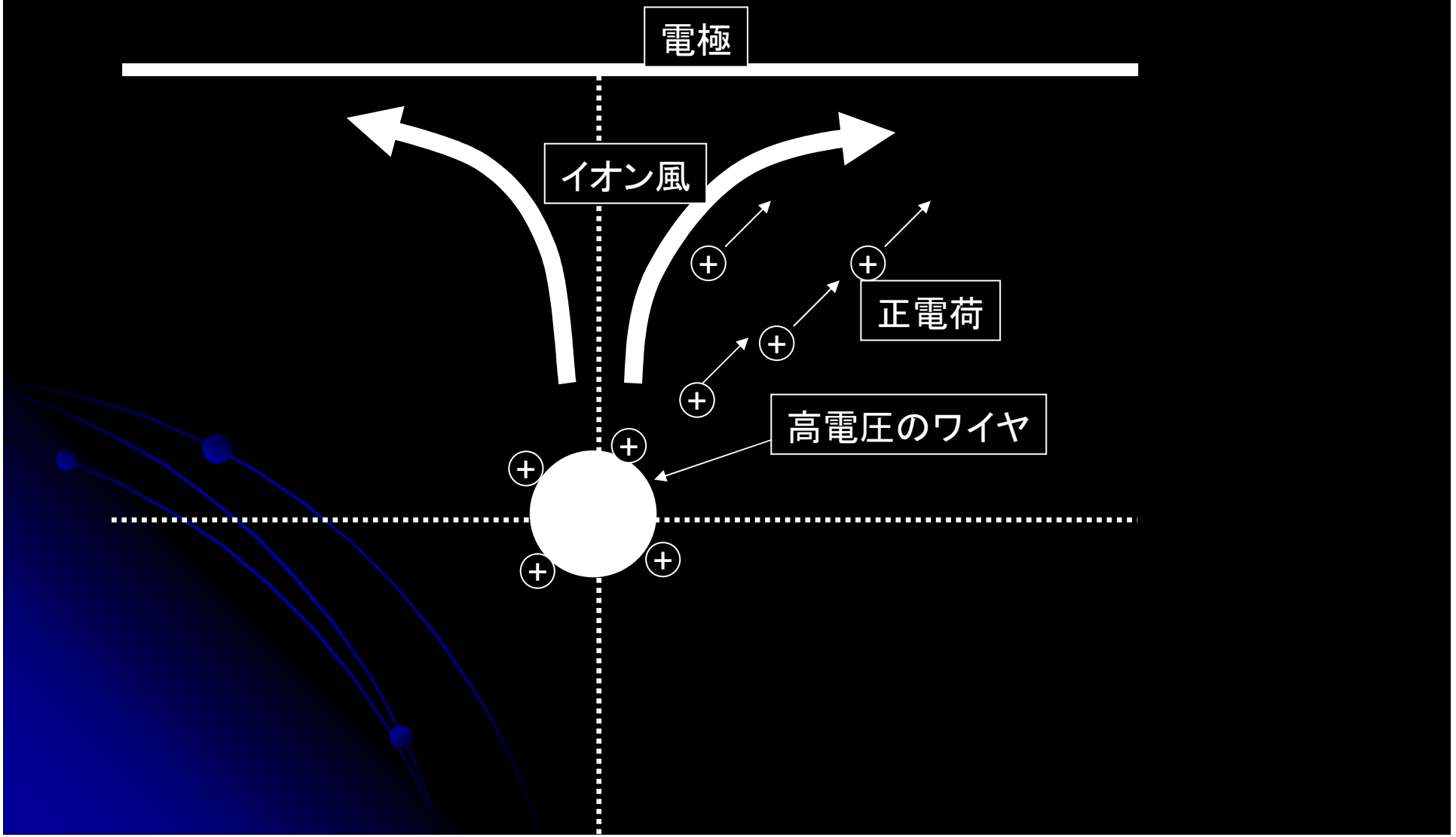
テストケースに必要なもの

- 計算形状
- 境界条件+初期条件
- 物理パラメーターの設定
- 計算スキームの設定(対流項など)



計算形状

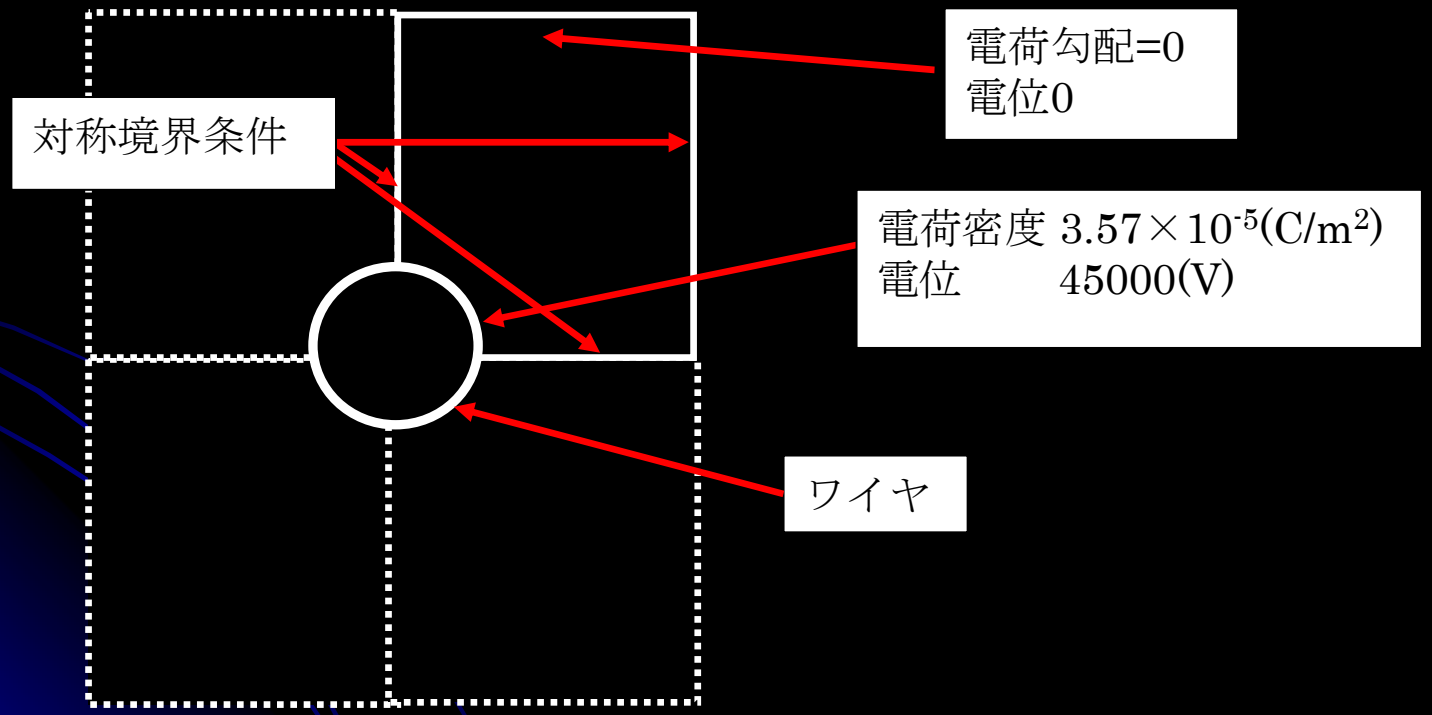
チュートリアル事例:chargedWireを使用



境界条件(1) 電荷rho+静電ポテンシャルesphi

境界条件=electrostaticFoamでの境界条件

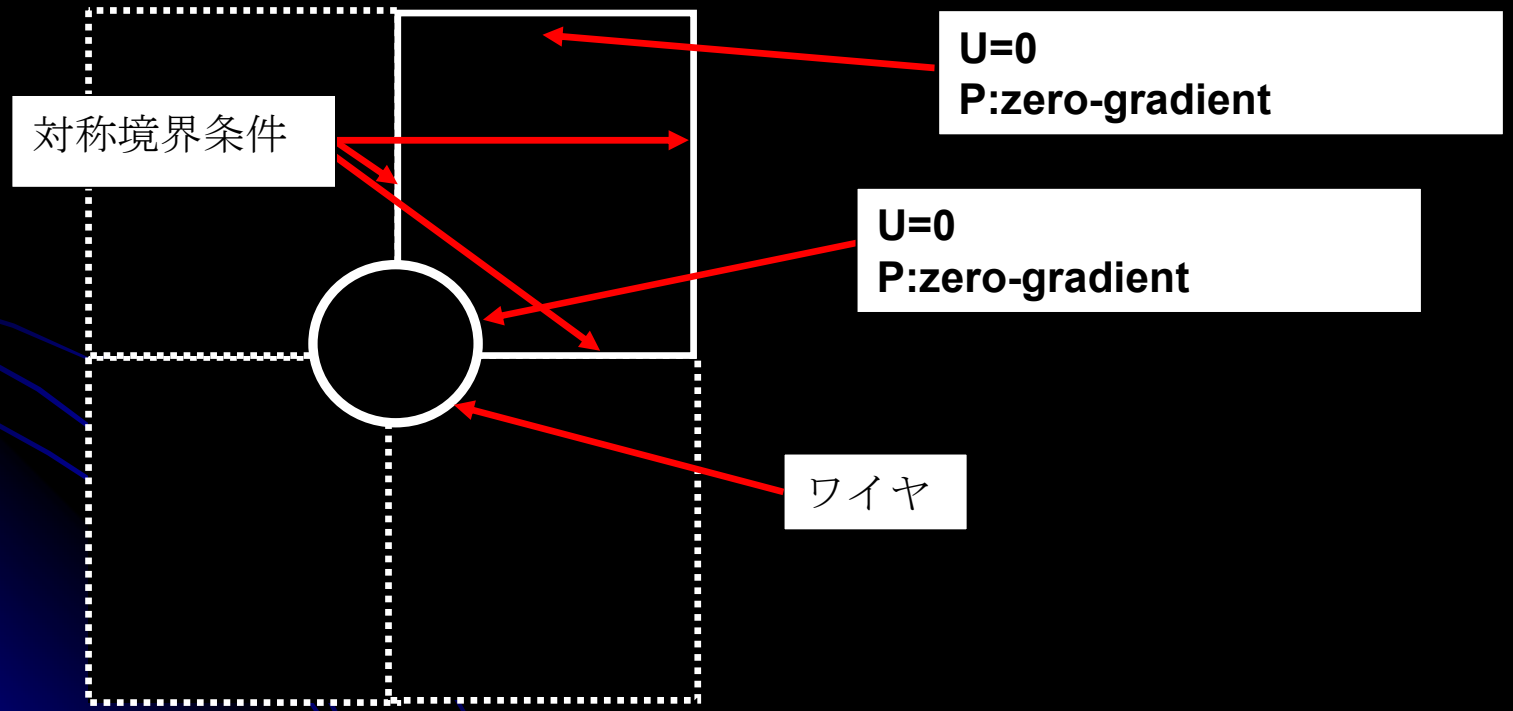
帯電ワイヤ ¼ モデル



境界条件(2) 流速U+圧力p

境界条件=electrostaticFoamでの境界条件

帯電ワイヤ ¼ モデル



テストケース作成
-テスト計算用データのコピー-

ディレクトリ作成と必要ファイルのコピー

恐らくpyfoamにより不要作業が激減するはずだと思いますが、泥臭く作業します。

```
$cd  OpenFOAM/OpenFOAM-1.7.1/tutorials
$mkdir user_case
$cd  user_case
$mkdir ehdcase
$cp –r ../electromagnetics/electrostaticFoam/chargedWire/*  ehdcase/
```

チュートリアルchargedWireを基本にします

```
$cp  ../incompressible/icoFoam/cavity/0/*   ehdcase/0/
$cp ../incompressible/icoFoam/cavity/constant/physicalProperties   ehdcase/constant
```

チュートリアルcavityから圧力と流速の境界条件をコピー

動粘性係数をコピー

境界条件設定(1)

-静電場境界条件作成-

静電場境界条件

恐らくpyfoamにより不要作業が激減するはずだと思いますが、泥臭く作業します。

静電ポテンシャルファイル名をphiからesphiに変更

```
$cd  ehdcase/0
$mv phi esphi
```

ファイルesphi中でも変数名を変更

esphi

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

名前の変更

境界条件設定(2)

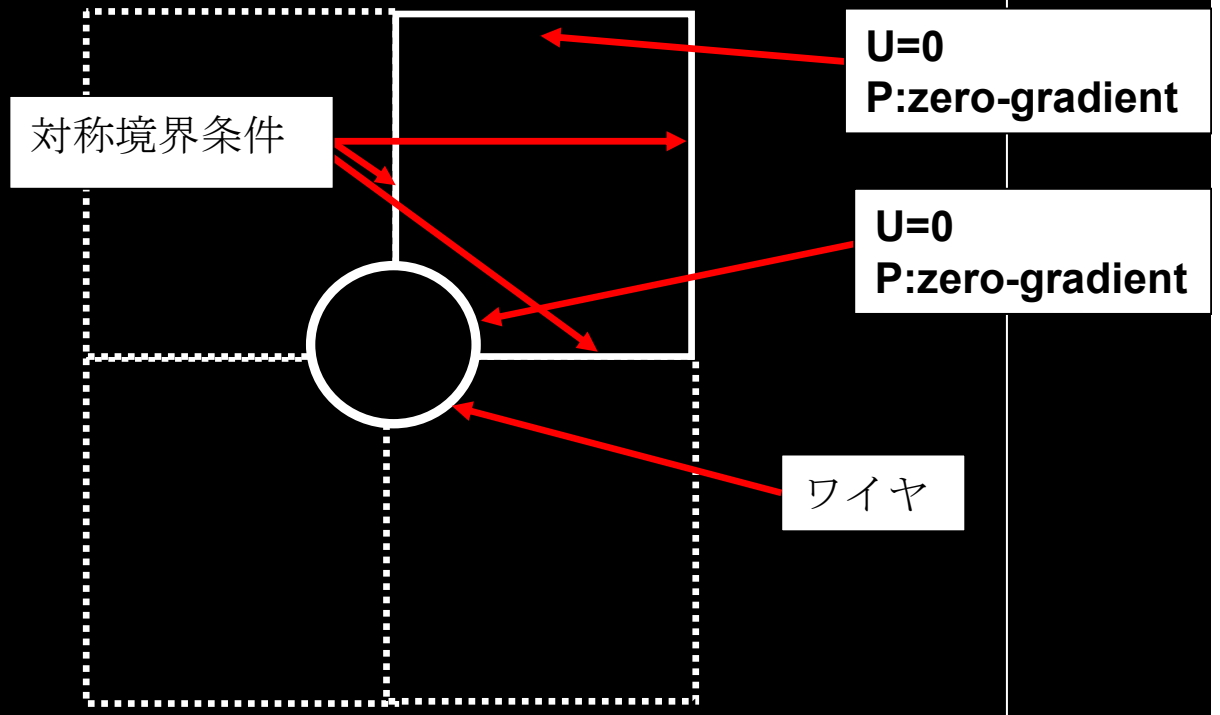
-流体境界条件作成1-

流体境界条件

形状や境界の情報を正しく反映する

p

```
…
boundaryField
{
    left
    {
        type            symmetryPlane;
    }

    right
    {
        type            symmetryPlane;
    }

    down
    {
        type            symmetryPlane;
    }

    up
    {
        type            zeroGradient;
    }

    hole
    {
        type            zeroGradient;
    }

    defaultFaces
    {
        type            empty;
    }
}
```



境界条件設定(3)

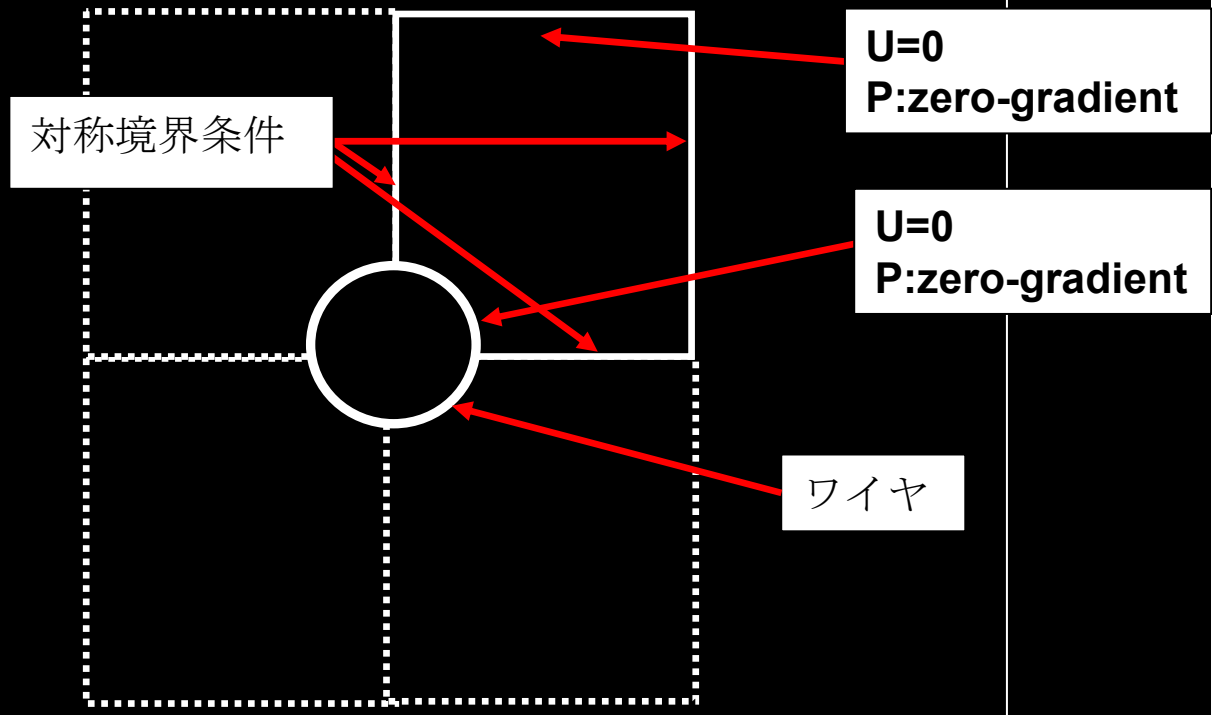
-流体境界条件作成2-

流体境界条件

形状や境界の情報を正しく反映する

U

```
…
boundaryField
{
 left
   {
     type          symmetryPlane;
   }

   right
   {
     type          symmetryPlane;
   }

   down
   {
      type          symmetryPlane;
   }

   up
   {
     type          fixedValue;
     value          uniform (0 0 0);
   }

   hole
   {
     type          fixedValue;
     value          uniform (0 0 0);
   }

   defaultFaces
   {
      type          empty;
   }
```



物理パラメーターの設定

-空気の密度設定-

物理パラメーターの設定

新しく作成した物理パラメーターdenMedium:空気密度を定義

Constant/physicalProperties

```
…
epsilon0        epsilon0 [ -1 -3 4 0 0 2 0 ] 8.85419e-12;

k               k [ -1 0 2 0 0 1 0 ] 0.00016;

denMedium       denMedium [ 1 -3 0 0 0 0 0 ] 1.1850;
…
```

計算スキームの設定(1)

-対流項やdivなど-

電場計算と流体計算の両方を
fvSchemes
fvSolution
で設定する

fvSolution

```
...
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }

    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }


    esphi
```

計算スキームの設定(2)

-対流項やdivなど-

fvSolution(続き)

```
…
esphi
    {
      solver          PCG;
      preconditioner  DIC;
      tolerance       1e-08;
      relTol          0.2;
    }

   rho
    {
      solver          PBiCG;
      preconditioner  DILU;
      tolerance       1e-08;
      relTol          0.2;
    }

}

PISO
{
   nCorrectors     2;
   nNonOrthogonalCorrectors 0;
   pRefCell        0;
   pRefValue       0;
}
…
```

計算スキームの設定(3)

-対流項やdivなど-

fvSchemes

```
…
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(esphi)     Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss  limitedLinear 1;
    div(rhoFlux,rho) Gauss limitedLinear 1;
    div(phi,rho)    Gauss limitedLinear 1;
}

laplacianSchemes
{
    default         none;
    laplacian(nu,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(esphi)  Gauss linear corrected;
}
```

計算スキームの設定(4)

-対流項やdivなど-

fvSchemes(続き)

```
…
interpolationSchemes
{
    default         linear;
    interpolate(HbyA) linear;
}

snGradSchemes
{
    default         corrected;
}

fluxRequired
{
    default         no;
    p               ;
    esphi           ;
}



…
```

計算条件の設定

-時間条件の設定変更-

controlDict

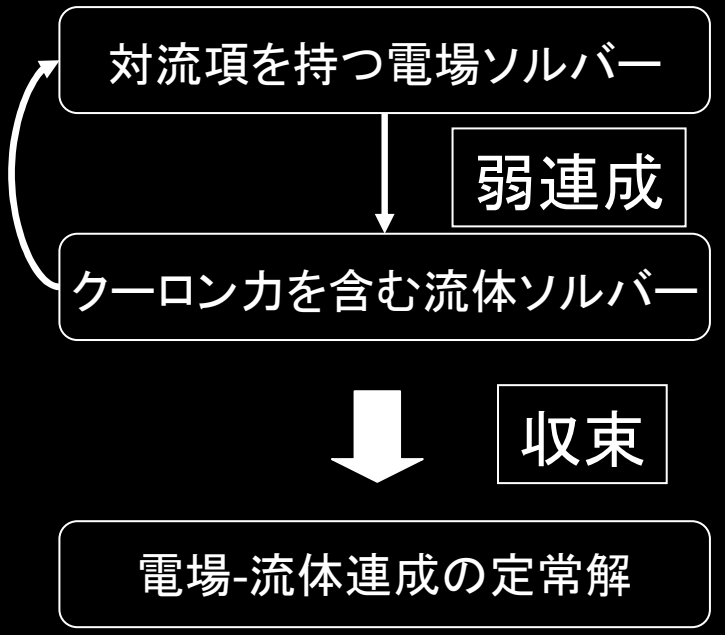
```
…
application     electrostaticFoam;
//startFrom       startTime;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         0.10;
deltaT          1e-6;
…
```

最新の計算時間から計算を
実行するように変えておく

計算実行
-弱連成に基づく非定常解析-

弱連成で計算を実行

```
$cd  OpenFOAM/OpenFOAM-1.7.1/tutorials
$cd user_case
$cd ehdcase
$convective_esFoam
…
…
$ico_esFoam
…
…
$convective_esFoam
…
…
$ico_esFoam
```



落ち着いたら計算途中終了。
別ソルバー立ち上げ。

泥臭いです。この辺はスクリプトで制御できるようにしたいです。
(sh?, perl?, python?)

まとめ+コメント

-言わなかったことや言い訳したいこと-

- 簡易EHDソルバーを作りましたが、まだまだ足りない部分があります。
- 本来必要な誘電体上への電荷降り積もり(壁電荷)境界条件は今回作成していません。境界条件のモデリングは、今後の課題です。
- 今後は、正と負のキャリアを準備したモデルを作成してみたいと思います。