به نام خدا

آموزش اضافه کردن ترم چشمه به معادلات ممنتوم در حلگر FSI

نسخه: foam-extend-3.1

توسعه دهنده:	آرش محبوبی دوست: کارشناسی ارشد، مهندسی مکانیک- تبدیل انرژی	
تهیه کننده مستند:	آرش محبوبی دوست	
تاریخ تنظیم سند:	۱۳ / ۵/ ۱۳۹۴	

فهرست مطالب

چکیده

در این مستند ابتدا یک بررسی موردی برای جریان تراکم‌ناپذیر آرام دو بعدی حول ایرفویل انعطاف‌پذیر NACA0012 دارای حرکت نوسانی پیچشی انجام شده است. نحوه ران، بدست آوردن گراف‌های مهم و تمامی متغیرهای ورودی لازم همراه با محل اعمال آنها، به دقت معرفی شده‌اند. سپس به آموزش اضافه کردن ترم چشمه به حلگر FSI پرداخته شده و چگونگی اعمال آن برای تست کیس قبلی توضیح داده خواهد شد. در نهایت خروجی‌ها و چگونگی استخراج آنها معرفی خواهند شد. با استفاده از این مستند کاربر قادر خواهد بود تا یک مسئله کامل FSI را با یک حلگر موجود اجرا نموده و همچنین یک ترم چشمه را به هر حلگر دلخواه دیگر اضافه کرده و نتایج دلخواه را استخراج نماید.

کلمات کلیدی: انعطاف‌پذیر، پلاسما، پیچشی، تعامل سازه-سیال.

فصل ۱: راهنمای کاربری

این مستند براساس نسخه توسعه‌یافته اپن‌فوم[1] foam-extend-3.1 می‌باشد. بنابراین ابتدا لازم است تا این نسخه از نرم‌افزار بر روی سیستم عامل لینوکس کامپایل شده و قابل دسترس باشد که به طور خلاصه به این موارد اشاره خواهد شد. سپس به آماده‌سازی مسئله جریان تراکم‌ناپذیر آرام دو بعدی حول ایرفویل انعطاف‌پذیر NACA0012 دارای حرکت نوسانی پیچشی می‌پردازیم و با حلگر FSI موجود آن را اجرا نموده و نتایج را استخراج خواهیم کرد. طول وتر ایرفویل ۱ متر و عدد رینولدز جریان ۱۲۰۰۰ می‌باشد که سرعت جریان آزاد ۰/۱۵۰۳۱۵۴ متربرثانیه را نتیجه می‌دهد.

۱-۱. نصب foam-extend-3.1 و راه‌اندازی حلگر fsiFoam

ابتدا سیستم عامل MeNoTu که بر اساس نسخه Ubuntu 12.04 می‌باشد را از سایت http://sourceforge.net/projects/menotu/ دانلود کرده و نصب نمایید. سپس طبق دستورات موجود در https://openfoamwiki.net/index.php/Installation/Linux/foam-extend-3.1/Ubuntu#Ubuntu_12.04 نسخه اوپن‌فوم مورد نظر را کامپایل نمایید. حال فقط با وارد کردن دستور fe31 در ترمینال، نسخه foam-extend-3.1 اجرا شده و قابل استفاده می‌باشد. معمولا مطابق با آنچه در راهنمای کاربری خود اوپن‌فوم نیز ذکر شده است، پس از نصب هر نسخه از اوپن‌فوم دو دستور زیر را اجرا می‌کنیم. دستور اول دایرکتوری کاربر را ایجاد می‌کند و دستور دوم توتوریال‌ها را از پوشه اصلی به دایرکتوری run کپی می‌کند.

[1] OpenFOAM

```
mkdir –p $FOAM_RUN
cp –r $FOAM_TUTORIALS $FOAM_RUN
```

با اجرای این دستورات دایرکتوری foam ایجاد شده و دایرکتوری به نام کاربر (مثلا arash-3.1) در داخل پوشه foam مشاهده می‌شود.

حلگر FSI موردنظر به خودی خود در این نسخه از اوپن‌فوم موجود نیست. بنابراین باید آن را دانلود و کامپایل نمود. برای این کار ابتدا از سایت http://openfoamwiki.net/index.php/File:Fsi_31.tar.gz فایل مربوطه را دانلود کنید. روش نصب حلگر به طریق زیر می‌باشد:

- فایل دانلودی را در پوشه home/arash/foam/arash-3.1/run کپی کرده و از حالت فشرده خارج نمایید.
- سپس ترمینال لینوکس را باز کرده و دستور fe31 را وارد کنید. با دستور زیر وارد دایرکتوری src از پوشه FluidStructureInteraction شوید:

```
cd /home/arash/foam/arashh-3.1/run/FluidStructureInteraction/src
```

- دستور Allwmake/. را برای کامپایل نمودن برنامه وارد نمایید. این دستور چند دقیقه‌ای به طول می‌انجامد. پس از اتمام فرایند کامپایل حلگر FSI موردنظر به نام fsiFoam قابل استفاده می‌باشد.

۱-۲. اجرای مسئله

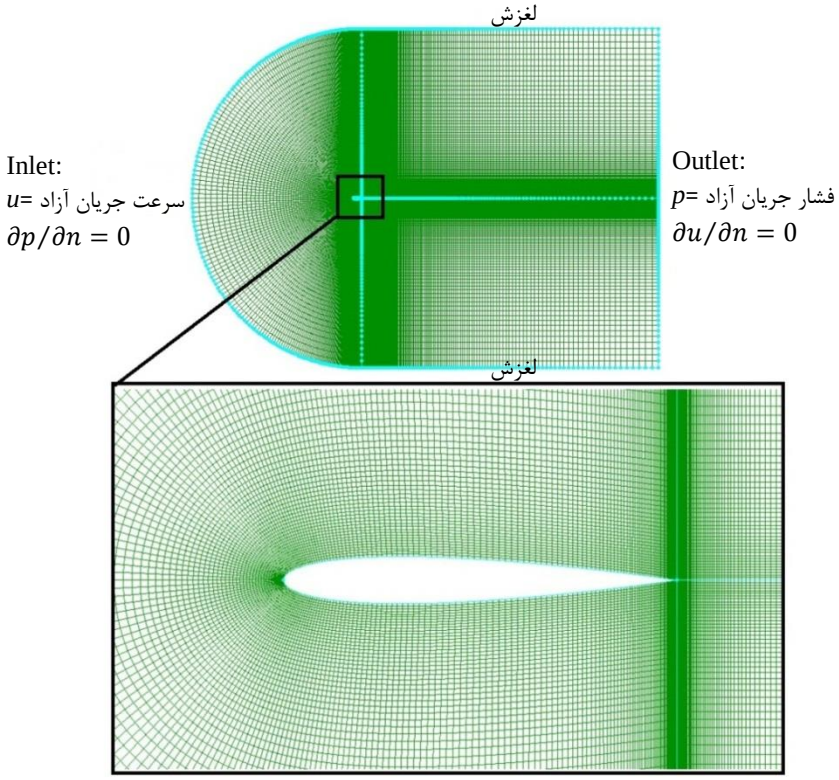
سه مورد توتوریال برای حلگر fsiFoam وجود دارد که شامل 3dTube، beamInCrossFlow و HronTurekFsi3 می‌باشد. برای شروع ابتدا توتوریال HronTurekFsi3 را در قسمت دلخواه کپی نمایید. فولدر fluid شامل فایل‌های موردنیاز برای محاسبه سیال و solid شامل فایل‌های موردنیاز برای محاسبه جامد می‌باشد. اجرای فایل‌های Links* دو محاسبه را در سطح اشتراک بین حوزه‌های سیال و جامد به یکدیگر لینک می‌کند. در این توتوریال در دایرکتوری‌های 0، constant و system فایل‌های لینک با نام solid مشاهده می‌شود که بدین معناست که لینک‌های مربوطه ایجاد شده‌اند. اگر لینکی ایجاد نشده باشد در پوشه HronTurekFsi3 دستور زیر را اجرا کنید تا طبق دستورات موجود در فایل makeSerialLinks، لینک‌های مربوطه ایجاد شود:

```
./makeSerialLinks fluid solid
```

در پوشه fluid سه دایرکتوری 0، constant و system مشاهده می‌شود. کلیه فایل‌های ورودی در این سه پوشه قرار دارد. حال باید این کیس را مطابق با مسئله خود تغییر داده و از آن استفاده نماییم.

۱-۲-۱. شبکه‌بندی و شرایط مرزی

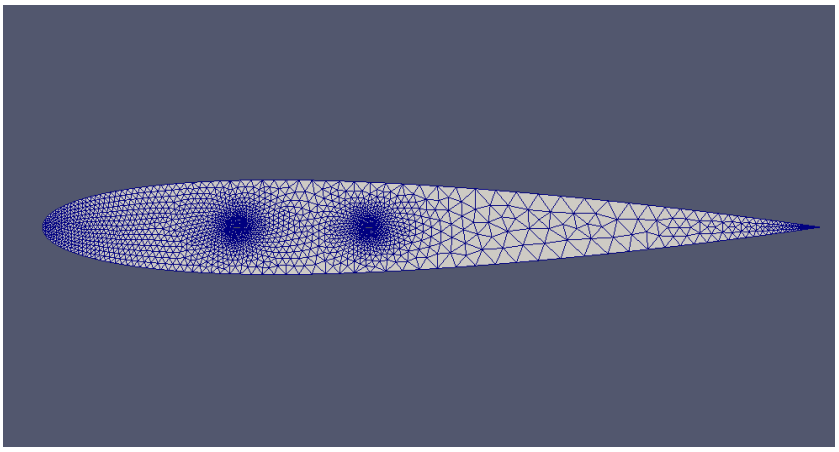
بدلیل اینکه شبکه‌بندی حول ایرفویل با استفاده از ابزار blockMesh آسان نمی‌باشد می‌توان از نرم‌افزارهای تجاری دیگر مانند GAMBIT و Pointwise استفاده نمود که در این مستند از نرم‌افزار Pointwise استفاده نموده‌ایم. یک شبکه باسازمان Cشکل برای مدلسازی جریان حول ایرفویل ایجاد شده است. شبکه و شرایط مرزی در شکل ۱ نشان داده شده است. در امتداد سطح ایرفویل شرط مرزی عدم لغزش به کار رفته است. مرز ورودی و بالا و پایین ۱۲/۵ برابر وتر از سطح ایرفویل و مرز خروجی در ۲۱برابر وتر واقع شده است. برای مرزهای بالا و پایین شرط مرزی لغزش استفاده شده است. برای ورودی، سرعت بر اساس عدد رینولدز ۱۲۰۰۰ محاسبه شده و برای فشار گرادیان صفر قرار داده شده است. در مرز خروجی فشار برابر فشار جریان آزاد است و گرادیان سرعت صفر می باشد.



شکل (۱) شبکه و شرایط مرزی

روش شبکه‌بندی cشکل حول ایرفویل در یک فیلم آموزشی قرار داده شده است و از توضیح دادن آن در این قسمت صرفنظر می‌کنیم.

شبکه ایجاد شده برای ناحیه جامد، یک شبکه بی‌سازمان می‌باشد. چون مسئله حاضر یک مسئله دوبعدی است، برای ایجاد حرکت نوسانی پیچشی دو قید، یکی در مرکز جرم و دیگری در ۰/۲۵ وتر ایجاد شد و دو حرکت سینوسی با دامنه مختلف به آنها اعمال شده است. شبکه ناحیه جامد در شکل ۲ نشان داده شده است.



شکل (۲) محدوده محاسباتی ناحیه جامد

پس از ایجاد شبکه، فایل polyMesh حاصل از نرم‌افزار Pointwise را به جای فایل polyMesh موجود در توتوریال HronTurekFsi3 قرار دهید. حال هندسه توتوریال به هندسه موردنظر ما تبدیل شده است و باید شرایط مرزی و دیگر پارامترها را بر اساس مسئله خود تغییر دهیم. در هنگام نامگذاری شرایط مرزی دقت کنید که پچ مربوط به سیال airfoil و پچ مربوط به جامد airfoilsolid نامگذاری شده‌اند.

۱-۲-۲. شبیه‌سازی جریان سیال

در این بخش بطور خلاصه به معرفی برخی از فایل‌های ورودی مهم برای شبیه‌سازی جریان سیال می‌پردازیم:

- جریان تراکم‌ناپذیر و آرام می‌باشد.
- حلگر سیال از الگوریتم PISO استفاده می‌کند.
- مش دینامیکی بدلیل جابجایی سطح اشتراک FSI موردنیاز می‌باشد.
- شبکه‌های داخلی مش سیال موقعیت خود را هنگام حرکت سطح اشتراک FSI تنظیم می‌کنند.

۱-۲-۲-۱. پوشه 0

در پوشه 0 شرایط مرزی و اولیه جریان مشخص می‌شود. فایل p وU ، شرایط مرزی فشار و سرعت و motionU برای تنظیم حرکت شبکه می‌باشد. شرایط مرزی را بصورت زیر تعریف کنید:

- U

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

internalField   uniform (0.1503154 0 0);

boundaryField
{
    airfoil
    {
        type            movingWallVelocity;
        value           uniform (0 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform (0.1503154 0 0);
    }
    upAndDown
    {
        type symmetryPlane;
    }
    frontAndBack
    {
        type            empty;
    }
}
```

- p

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

internalField   uniform 0;

boundaryField
{
```

```
    airfoil
    {
        type extrapolatedPressure;
        value uniform 0;
    }
    outlet
    {
        type fixedValue;
        value uniform 0;
    }
    inlet
    {
        type extrapolatedPressure;
        value uniform 0;
    }
    upAndDown
    {
        type symmetryPlane;
    }
    frontAndBack
    {
        type empty;
    }
}
```

- motionU

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

internalField   uniform (0 0 0);

boundaryField
{
    airfoil
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    outlet
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
```

```
    inlet
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    upAndDown
    {
        type symmetryPlane;
    }
    frontAndBack
    {
        type            empty;
    }
}
```

۱-۲-۲-۲. پوشه constant

- پوشه constant شامل فایل polyMesh می‌باشد که شبکه‌بندی حوزه سیال در آن ایجاد می‌شود.
- فایل dynamicMeshDict شامل تنظیمات مش دینامیکی می‌باشد:

```
dynamicFvMesh dynamicMotionSolverFvMesh;
twoDMotion      yes;
solver laplace;   //velocityLaplacian;
diffusivity quadratic inverseDistance 1(airfoil);
frozenDiffusion yes;
distancePatches
(
    airfoil
);
```

حلگر laplace معادله لاپلاسین را حل می‌کند. در خط چهارم در داخل پرانتز نام پچ‌های سیال سطح اشتراک FSI نوشته می‌شود و عدد ۱ تعداد این پچ را نشان می‌دهد.

- در فایل flowProperties مدل جریان مشخص می‌شود که شامل سه مدل consistentIcoFlow، icoFlow برای جریان آرام و pisoFlow برای جریان آشفته می‌باشد. icoFlow معادل با حلگر icoDyMFoam بدون ddtPhiCorr می‌باشد. consistentIcoFlow معادل با icoDyMFoam با ddtPhiCorr می‌باشد. pisoFlow معادل با حلگر pisoFoam با دینامیک مش می‌باشد. این فایل را بدون تغییر حفظ می‌کنیم.
- در فایل fsiProperties تنظیمات مربوط به اتصال حلگرها انجام می‌شود:

```
solidPatch airfoilsolid;
solidZone airfoilsolidZone;
fluidPatch airfoil;
fluidZone airfoil Zone;
relaxationFactor 0.01;
interfaceDeformationLimit 0;
outerCorrTolerance 1e-6;
nOuterCorr 50;
interpolatorUpdateFrequency 0;
couplingScheme Aitken;
//couplingScheme IQN-ILS;
couplingReuse 0;
coupled yes; // Will be swithed to yes by the function object
```

- در فایل transportProperties خواص سیال مانند دانسیته و ویسکوزیته تعیین می‌شود.

```
nu          nu [0 2 -1 0 0 0 0] 1.252628e-5;

rho         rho [1 -3 0 0 0 0 0] 1;
```

۱-۲-۲-۳. پوشه system

این پوشه شامل سه فایل controlDict، fvSchemes و fvSolution می‌باشد. فایل controlDict تمامی تنظیمات کنترل زمانی و ورودی و خروجی‌ای که در حلگر سیال مشخص شده‌اند را دربرمی‌گیرد. فایل fvSchemes شامل تمامی طرح‌های گسسته‌سازی عددی مورداستفاده برای حل دیورژانس‌ها، لاپلاسین‌ها و گسسته‌سازی زمانی است. فایل fvSolution نیز حلگر متغیرهای موجود در مسئله و خطای موردنیاز برای همگرایی آنها و همچنین ضرایب underrelaxation را در صورت نیاز مشخص می‌نماید. حلگر laplace در فایل dynamicMeshDict برخلاف velocityLaplacian حرکت شبکه را بر اساس روش المان محدود حل می‌کند و برای حل motionU نیاز به فایل tetFemSolution در پوشه system دارد.

- controlDict

```
application     fsiFoam;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         63;
deltaT          0.01;
writeControl    adjustableRunTime;
writeInterval   0.5;
```

```
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression uncompressed;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;
adjustTimeStep  yes;
maxCo           1.0;
functions
(
  pointHistory
  {
     type pointHistory;
     functionObjectLibs
     (
       "libpointHistory.so"
     );
     region solid;
     refHistoryPoint (1.0 0.0 0.0);
  }

  hronTurekReportAirfoil
  {
     type hronTurekReportAirfoil;
     functionObjectLibs
     (
       "libhronTurekReportAirfoil.so"
     );
  }
);

libs (
       "libOpenFOAM.so"
       "libgroovyBC.so"
       "libsimpleSwakFunctionObjects.so"
       "libswakFunctionObjects.so"
);
```

- در مورد توابع pointHistory و hronTurekReportAirfoil و همچنین کتابخانه‌هایی که در انتهای این فایل آورده شده است در بخش‌های بعد توضیح خواهیم داد.

- fvSchemes

```
ddtSchemes
{
    default backward;
}
gradSchemes
{
    default leastSquares;
}
divSchemes
{
    default none;
    div(phi,U)   Gauss linearUpwind cellLimited leastSquares 1;
}
laplacianSchemes
{
    default none;
    laplacian(nu,U) Gauss linear skewCorrected 1;
    laplacian((1|A(U)),p) Gauss linear skewCorrected 1;
    laplacian(diffusivity,cellMotionU) Gauss linear skewCorrected 1;
}
interpolationSchemes
{
    default linear;
    interpolate(U) skewCorrected linear;
}
snGradSchemes
{
    default skewCorrected 1;
}
fluxRequired
{
    default no;
    p;
}
```

- fvSolution

```
solvers
{
    p
    {
        solver          GAMG;
```

```
    tolerance       1e-08;
    relTol          0;
    minIter         1;
    maxIter         1000;
    smoother        GaussSeidel;
    nPreSweeps      0;
    nPostSweeps     2;
    nFinestSweeps   2;
    scaleCorrection true;
    directSolveCoarsest false;
    cacheAgglomeration true;
    nCellsInCoarsestLevel 20;
    agglomerator    faceAreaPair;
    mergeLevels     1;
}

cellMotionU
{
    solver          GAMG;
    tolerance       1e-08;
    relTol          0;
    minIter         1;
    maxIter         100;
    smoother        GaussSeidel;
    nPreSweeps      0;
    nPostSweeps     2;
    nFinestSweeps   2;
    scaleCorrection true;
    directSolveCoarsest false;
    cacheAgglomeration true;
    nCellsInCoarsestLevel 20;
    agglomerator    faceAreaPair;
    mergeLevels     1;
}

U
{
    solver          PBiCG;
    preconditioner  DILU;
    tolerance       1e-08;
    relTol          0;
    minIter         1;
}
```

```
}
```

- tetFemSolution

```
solvers
{
motionU
{
   solver          PCG;
   preconditioner  DIC;
   tolerance       1e-08;
   relTol          0.001;
}
}
```

۱-۲-۳. شبیه‌سازی جامد

در این بخش بطور خلاصه به معرفی برخی از فایل‌های ورودی مهم برای شبیه‌سازی جامد می‌پردازیم:

۱-۲-۳-۱. شرایط مرزی

در پوشه 0 دو فایل D و pointD شرایط مرزی مربوط به حلگر جامد می‌باشند. اعمال حرکات نوسانی به قیدهای ایجاد شده در هندسه در فایل D صورت می‌پذیرد. برای ایجاد حرکت پیچشی باید قیدی که در مرکز جرم قرار دارد ثابت بوده و معادله حرکت سینوسی به قید موجود در ۰/۲۵ وتر اعمال شود. دقت کنید به دلیل استفاده از شرایط مرزی غیر یکنواخت می‌بایست برنامه swak4Foam مخصوص اپن‌فوم foam-extend-3.1 دانلود و نصب شود. در فایل D از شرط مرزی نوع groovyBC برای اعمال حرکت سینوسی استفاده شده است. لذا لازم است ابتدا کتابخانه‌های مربوط به این شرط مرزی در فایل controlDict در پوشه system وارد شوند. این کتابخانه‌ها شامل موارد زیر می‌باشند که در انتهای controlDict نیز قابل مشاهده‌اند:

```
libs (
        "libOpenFOAM.so"
        "libgroovyBC.so"
        "libsimpleSwakFunctionObjects.so"
        "libswakFunctionObjects.so"
);
```

- D

```
dimensions      [0 1 0 0 0 0 0];

internalField   uniform (0 0 0);
boundaryField
{
   airfoilsolid      //fsiPatchName
   {
      type          tractionDisplacement;     //The BC type
      traction       uniform (0 0 0);       //Externally imposed traction
      pressure        uniform 0;             //Externally imposed pressure
      value          uniform (0 0 0);       // Externally imposed displacement
   }
   pitchPA
   {

      type          groovyBC;
      variables "pi=3.14159265359;" ;
      valueExpression "vector(0,0.011887558*sin(1.5061*time()),0) " ;
      value          uniform (0 0 0);

   }
   CGPA
   {
      type fixedValue;
      value uniform (0 0 0);
   }
   frontAndBack
   {
      type          empty;
   }
}
```

کشش نیرو به ازای واحد سطح بر روی یک صفحه است، که شامل مؤلفه‌های نرمال و برشی است.

- pointD

```
dimensions      [0 1 0 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
   airfoilsolid
   {
      type          calculated;
      value          uniform (0 0 0);
```

```
    }
    pitchPA
    {
        type            calculated;
        value           uniform (0 0 0);
    }
    CGPA
    {
        type            calculated;
        value           uniform (0 0 0);
    }
    frontAndBack
    {
        type            empty;
    }
}
```

۱-۲-۳-۲. تعیین خواص جامد

فایل‌های موجود در دایرکتوری constant، یعنی rheologyProperties و stressProperties خواص جامد را تعیین می‌کنند.

- rheologyProperties

```
planeStress no;    //‘yes’ for 2D cases, ‘no’ for 3D cases.
rheology
{
    type  linearElastic;    //The structure is of linear elasticity.
    rho   rho [1 -3 0 0 0 0 0]  1320;   //Density.
    E     E [1 -1 -2 0 0 0 0] 2.75e6;     //Young’s modulus.
    nu    nu [0 0 0 0 0 0 0] 0.35;      // Poisson’s ratio.
}
```

۱-۲-۳-۳. طرح‌های عددی حلگر جامد

- fvSchemes

```
d2dt2Schemes
{
    default none;
    d2dt2(D) backward;
}
ddtSchemes
{
```

```
    default none;
    ddt(D) backward;
}
```

✓ D مربوط به جابجایی $\vec{u}$ در معادلات جامد است.

```
laplacianSchemes
{
    default none;
    laplacian(DD,D) Gauss linear skewCorrected 1;  // Gauss linear corrected
}
snGradSchemes
{
    default none;
    snGrad(D) skewCorrected 1;    // corrected
}
```

مزیت: مناسب برای کیفیت مش بد
عیب: زمان محاسباتی زیاد

✓ DD به معنی افزایش جابجایی $\delta\vec{u}$ در معادلات جامد است.

```
interpolationSchemes
{
    default none;
    interpolate(mu) linear;         // leastSquares
    interpolate(lambda) linear;    // leastSquares
}
```

مزیت: مناسب برای کیفیت مش بد
عیب: زمان محاسباتی زیاد

۱-۲-۳-۴. الگوریتم ماتریس حلگر جامد

- fvSolution

```
solvers
{
    D
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-09;
        relTol          0.1;
    }
}
relaxationFactors
{
}
```

۱-۲-۴. اجرای حلگر

همانطور که قبلا ذکر شد، در تست کیس یک پوشه به نام hronTurekReport مشاهده می‌کنید که شامل دو تابع کتابخانه‌ای pointHistory برای استخراج مختصات نقطه دلخواه جابجا شده و تابع hronTurekReport برای محاسبه نیروی لیفت و درگ می‌باشد که در انتهای فایل controlDict مربوط به حوزه سیال فراخوانی می‌شوند. داده‌های حاصل از این توابع در پوشه‌ای به نام history در حین ران ذخیره می‌شوند. تابع hronTurekReport برای محاسبه لیفت و درگ مربوط به مسئله HronTurek [۱] می‌باشد. بنابراین قبل از اجرای حلگر باید آن را متناسب با مسئله خود تغییر دهیم. همچنین نام کتابخانه را نیز تغییر می‌دهیم. برای این کار ابتدا نام دو فایل hronTurekReport.C و hronTurekReport.H را به hronTurekReportAirfoil.C و hronTurekReportAirfoil.H تغییر می‌دهیم. همچنین تمامی اسامی hronTurekReport موجود در محتوای این فایل‌ها را به hronTurekReportAirfoil تغییر می‌دهیم. در فایل hronTurekReportAirfoil.C کلمه cylinder را به airfoil تغییر می‌دهیم. از خط ۹۵ تا ۱۲۵ را غیرفعال می‌کنیم. چون در مسئله HronTurek مجموع ضرایب مربوط به میله و استوانه محاسبه می‌شود که در اینجا فقط یک پچ ایرفویل وجود دارد. در نهایت در فایل files که در پوشه make قرار دارد نیز hronTurekReport را به hronTurekReportAirfoilتغییر می‌دهیم. سپس ترمینال را در پوشه hronTurekReportAirfoil باز کرده و دستورات زیر را وارد می‌کنیم:

```
$wclean
$wmake libso
```

اکنون این تابع کامپایل شده و قابل استفاده برای محاسبه نیروی لیفت و درگ می‌باشد.

به دلیل اینکه نسبت چگالی جامد به سیال بزرگتر از ۱۰۰۰ می‌باشد می‌توان مسئله FSI را از نوع اتصال ضعیف در نظر گرفت و از حلگر weakFsiFoam استفاده کرد. لذا ابتدا در فایل controlDict، مقابل application بجای fsiFoam، weakFsiFoam قرار می‌دهیم. در فایل setBatch موجود در پوشه fluid بجای plate، airfoil و در فایل setBatch موجود در پوشه solid بجای plate، airfoilsolid را قرار می‌دهیم که بترتیب بصورت زیر درمی‌آیند.

```
faceSet airfoilZone new patchToFace airfoil
quit
```

```
faceSet airfoilsolidZone new patchToFace airfoilsolid
quit
```

در نهایت در فایل Allrun موجود در پوشه fluid عبارات مربوط به blockMesh و کامپایل تابع hronTurekReport را حذف می‌کنیم که بصورت زیر درمی‌آید:

```
#!/bin/sh
 #Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

#Get application name
application=`getApplication`

#runApplication -l log.blockMesh.solid blockMesh -region solid
runApplication -l log.setSet.solid setSet -case ../solid -batch ../solid/setBatch
runApplication -l log.setToZones.solid setsToZones -case ../solid -noFlipMap

#runApplication blockMesh
runApplication setSet -batch setBatch
runApplication setsToZones -noFlipMap

#Build hronTurekReport function object
#wmake libso ../hronTurekReport

runApplication $application
```

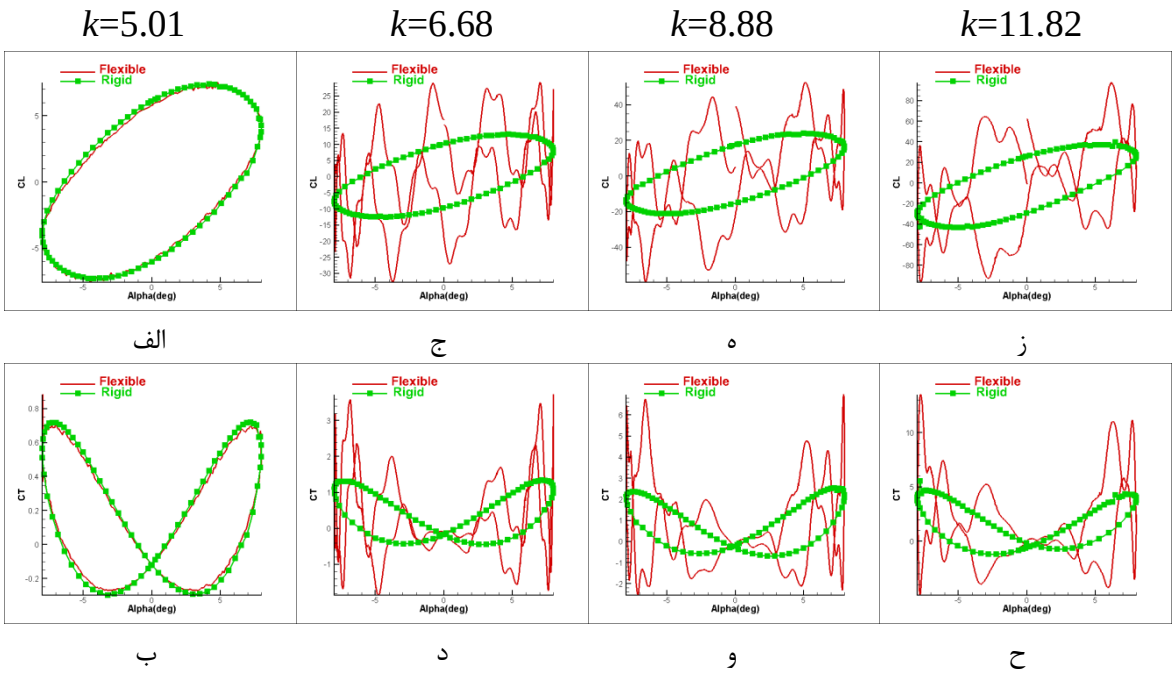
اکنون با اجرای دستور ./Allrun. مسئله شروع به حل می‌کند.

۱-۲-۵. پس پردازش

ابزار پس پردازش می‌تواند paraFoam یا سایر نرم‌افزارهای تجاری باشد. با اجرای دستور paraFoam نتایج هر دو حوزه سیال و جامد بطور اتوماتیک لود می‌شود. برای استفاده از این نتایج در نرم‌افزار tecplot نیز دو روش وجود دارد. یکی استفاده از دستور foamToTecplot360 می‌باشد. اما نسخه‌های جدید tecplot با فراخوانی فایل controlDict مستقیما قادر به لود کردن نتایج OpenFOAM می‌باشند.

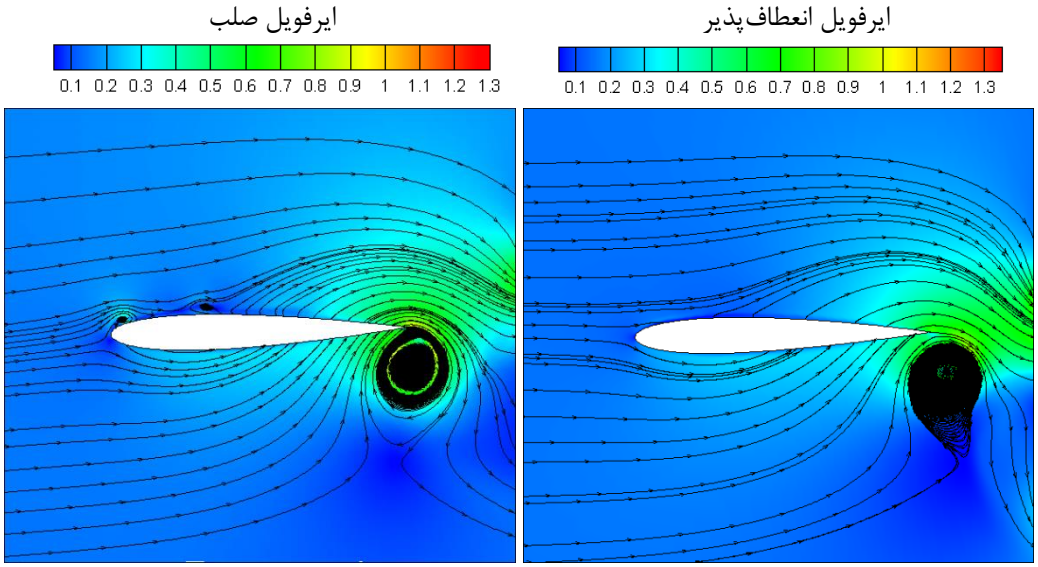
شکل ۳ مقایسه نمودارهای ضریب لیفت و درگ ایرفویل صلب و انعطاف‌پذیر را برای دامنه نوسان $A = 8°$ و فرکانس کاهشی $k = \omega c/2U_\infty = 5.01,6.68,8.88,11.82$ نشان می‌دهد. همانطور

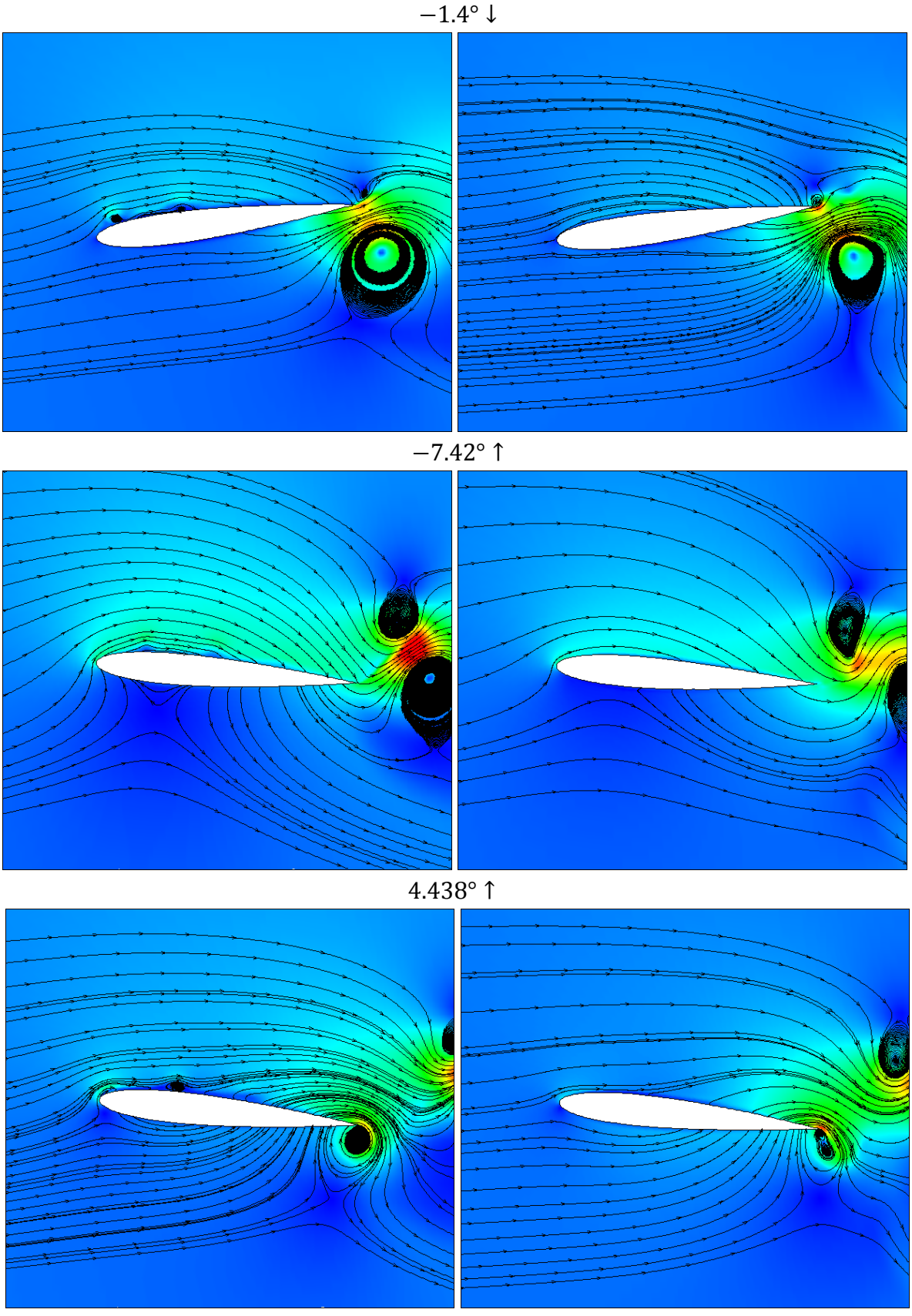
که از این نمودارها مشخص است، در فرکانس‌های بسیار پایین منحنی‌های لیفت و درگ بر هم منطبق‌اند و با افزایش فرکانس و افزایش ارتعاشات ناشی از جریان که بدلیل اثرات انعطاف‌پذیری است، منحنی‌ها دچار نوسانات می‌شود.



شکل (۳) مقایسه مشخصه‌های ایرفویل صلب و انعطاف‌پذیر در دامنه پیچشی ؛ الف ، ج ، ه ، ز ضریب لیفت و ب ، د ، و ، ح ضریب تراست بر حسب دامنه نوسان

شکل ۴ نیز مقایسه کانتور سرعت را برای دو ایرفویل صلب و انعطاف‌پذیر در $A = 8°$ و $k =$ 11.82 نشان می‌دهد. در هیچ یک از فرکانس‌ها جدایش جریان رخ نداد و جریان در سطح ایرفویل کاملا متصل ماند. همچنین مشاهده می‌شود که انعطاف‌پذیری باعث از بین رفتن گردابه‌های کوچک سطح بالایی شده است و باعث افزایش لیفت و تراست در این فرکانس شده است.





شکل (۴) مقایسه کانتورهای سرعت دو ایرفویل صلب و انعطاف‌پذیر در k=11.82

فصل ۲: مستندات آموزشی

در این فصل ابتدا مقدمه‌ای بر کنترل جریان روی ایرفویل را مطرح می‌کنیم. سپس به مدلسازی کنترل جریان محرک پلاسمایی و معادلات مربوط به آن می‌پردازیم. در ادامه نحوه اعمال این معادلات به حلگر FSI مورداستفاده با جزئیات تمام مورد بحث و بررسی قرار می‌گیرد.

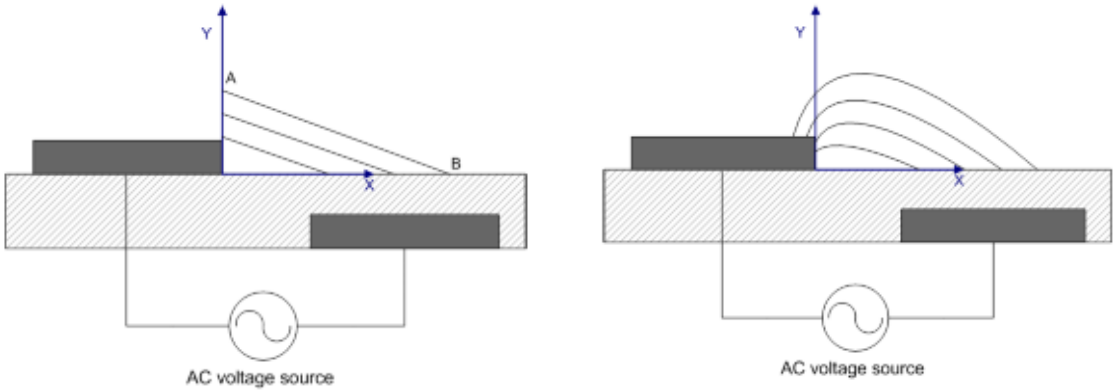
کنترل جریان روی سطح ایرفویل اهمیت زیادی در طراحی وسایل نقلیه هوایی دارد. محرک‌های مختلفی برای کنترل جریان وجود دارد اما اخیرا محرک پلاسمای تخلیه مانع دی الکتریک توجه محققان را به عنوان یک دستگاه آیرودینامیکی کنترل جریان به خود جلب کرده است. این محرک معمولا از یک الکترود در معرض هوا و یک الکترود تعبیه شده در قسمت زیرین بدنه تشکیل شده است، که توسط یک ورق دی‌الکتریک از هم جدا می‌شوند. الکترودها در ولتاژ و فرکانس بالا انرژی می‌گیرند، که باعث می‌شود هوا در طول الکترود تعبیه شده یونیزه شود که در نهایت یک جت کوچک در نزدیکی سطح تولید می‌کند. از مزایای محرک پلاسما این است که چون کاملا الکترونیکی است به اجزای متحرک نیاز ندارد، پاسخ سریع و جرم بسیار کمی دارد، نیاز به توان ورودی کمی دارد و شبیه‌سازی عددی آن آسان است.

۲-۱. مدلسازی پلاسما

پلاسما در اثر اضافه کردن مقدار کافی انرژی به یک گاز مولکولی تولید می‌شود. این انرژی توسط اعمال ولتاژ به الکترودها از طریق میدان الکتریکی، به الکترون‌ها و یون‌های اطراف منتقل می‌شود. در فرکانس‌های بالا، ذرات باردار به سرعت‌های بالایی می‌رسند. اگرچه الکترون‌ها سرعت بیشتری نسبت به یون‌ها دارند، اما به دلیل جرم کوچکشان، تأثیر مستقیمشان بر انتقال ممنتوم بسیار کوچک است. بنابراین

انتقال ممنتوم بین پلاسما و محیط از طریق یون تحقق می‌یابد. فشار الکترواستاتیکی ناشی از میدان الکتریکی سیال را به سمت جلو می‌راند. برای مدلسازی پلاسما مدل‌های مختلفی ارائه شده است که عبارتند از: مدل الکترواستاتیکی، مدل مدار المان-فشرده، مدل نیروی خطی، مدل جریان پتانسیل که در این پایان‌نامه از مدل نیروی خطی استفاده شده است[۲].

جایارامان و همکاران [۳] از یک الگوی ساده خطوط میدان الکتریکی تولید شده بین دو الکترود استفاده کردند. در واقع می توان در شکل ۱-الف دید که خطوط میدان الکتریکی در کاتد متمرکز و در سراسر آند توزیع شده‌اند.



الف: آرایش محرک با شکل تقریبا واقعی خطوط میدان الکتریکی **ب**: آرایش محرک با شکل خطی خطوط میدان الکتریکی

شکل (۱) مقایسه خطوط میدان الکتریکی مدل نیروی خطی و خطوط میدان الکتریکی اصلی[۲]

این فرض منجر به مدلسازی خطوط میدان الکتریکی بصورت موازی، به جز در فضای کوچک نزدیک کاتد می‌شود. این ساده‌سازی تغییر میدان جریان را خطی‌سازی می‌کند و بنابراین هیچ محاسبه دقیقی از میدان الکتریکی نیاز نیست. تغییر میدان الکتریکی فرض‌شده می‌تواند به صورت زیر نوشته شود:

$$|E| = E_0 - k_1 x - k_2 y \qquad (۱)$$

که در آن E_0 میدان الکتریکی در ماده دی الکتریک در فاصله بین دو الکترود است و می‌تواند به صورت $E_0 = U_a/d$ تقریب زده شود که در آن d فاصله جدایی بین دو الکترود در جهت x است و U_a جذر ریشه میانگین ولتاژ اعمال شده است. علاوه بر این k_1 و k_2 ثابت‌های مثبت هستند. علامت دو ثابت اطمینان حاصل می‌کند که شدت میدان الکتریکی وقتی که در امتداد جهت مثبت محورها حرکت می‌کنیم کاهش می‌یابد. در شکل ۱ می‌توان مقایسه بین خطوط میدان الکتریکی واقعی و خطوط میدان الکتریکی خطی را دید. توجه داشته باشید که خط AB در شکل ۱(ب) مرز سیال پلاسما با استفاده از تقریب خطی است. فرض می‌شود که قدرت میدان الکتریکی خارج از این خط به اندازه کافی برای یونیزاسیون هوا قوی نیست و بنابراین هیچ پلاسمایی خارج از این خط وجود ندارد. این امر توسط ولتاژ شکست دی الکتریک

تعریف می‌شود. ولتاژ شکست حداقل ولتاژ مورد نیاز برای هدایت الکتریکی اجباری بخش دی الکتریک است. متوسط زمانی نیرو در کل چرخه کامل بصورت زیر در نظر گرفته می‌شود:

$$\mathbf{F}_x = \vartheta\alpha\rho_c e_c \Delta t E_x \delta, \quad \mathbf{F}_y = \vartheta\alpha\rho_c e_c \Delta t E_y \delta \qquad (۲)$$

$$E_x = \frac{Ek_2}{\sqrt{k_1^2+k_2^2}}, \quad E_y = \frac{Ek_1}{\sqrt{k_1^2+k_2^2}} \qquad (۳)$$

$$\begin{cases} \delta = 1 & for\ E > E_{cr} \\ \delta = 0 & for\ E \le E_{cr} \end{cases} \qquad (۴)$$

که در آن ϑ (معکوس دوره زمانی) فرکانس ولتاژ اعمال شده، α ضریب راندمان برخورد، ρ_c چگالی عددی الکترون، e_c بار الکترون، Δt زمان تخلیه پلاسماست که بسیار کوچکتر از مقیاس زمانی سیال است، E قدرت میدان الکتریکی می‌باشد. تابع δ به این دلیل وارد شده است که نیرو تنها در مناطقی که پلاسما در آن وجود دارد عمل کند. E_{cr} قدرت میدان الکتریکی شکست یا همان E_b می‌باشد. معادلات نیروی حجمی به طور مستقیم می‌تواند به عنوان یک ترم چشمه به معادلات ناویر استوکس اضافه شود. بزرگترین نقطه ضعف این مدل فرض کاهش قدرت میدان الکتریکی بطور خطی وقتی به دور از لبه داخلی الکترود حرکت می‌کنیم می‌باشد. جدول ۱ مقادیر پارامترهای مورد استفاده برای تعیین نیروی پلاسما را می‌دهد.

جدول (۱) پارامترهای انتخاب شده برای مدلسازی نیروی پلاسما

پارامتر	مقدار	واحد
ϑ	3000	Hz
ρ_c	1.03e+17	$\mathbf{1/m^3}$
e_c	1.602e-19	coulombs
Δt	67e-06	sec
E_b	30e+05	$\boldsymbol{V/m}$
U_a	4000	**Volts**

بنابراین برای جریان دو بعدی آرام تراکم‌ناپذیر و ناپایا معادلات پیوستگی و ممنتوم بصورت زیر نوشته می‌شود:

$$\nabla.(\rho \mathbf{v}) = 0 \qquad (۵)$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\frac{\partial \boldsymbol{u}}{\partial x} + \boldsymbol{v}\frac{\partial \boldsymbol{u}}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 \boldsymbol{u}}{\partial x^2} + \frac{\partial^2 \boldsymbol{u}}{\partial y^2}\right) + \mathbf{F}_x \qquad (۶)$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + F_y \qquad (۷)$$

۲-۲. اعمال معادلات پلاسما به حلگر

اکنون به چگونگی اعمال معادلات ذکر شده در بخش قبل به حلگر FSI می‌پردازیم.

تغییرات اعمال شده ناشی از تحریک پلاسما در دو فایل icoFlow.H و icoFlow.C که در آدرس FluidStructureInteraction/src/fluidStructureInteraction/flowModels/icoFlow قرار دارند باید انجام گیرد.

۲-۲-۱. تغییرات فایل icoFlow.C

ابتدا باید متغیرهایی که در معادلات پلاسما موجودند تعریف شوند. نیروی ناشی از پلاسما، برداری و دارای دو مؤلفه در راستای x و y می‌باشد که به مراکز سلول‌ها اعمال می‌شوند. بنابراین بعد از معرفی متغیر phi در فایل icoFlow.C عبارات زیر را وارد می‌کنیم:

```
const volVectorField& centres =(U_.mesh().C()),    //تعریف سرعت مرکز سلول
volScalarField xx_((centres & vector(1,0,0))),     // تعریف مختصات x مرکز سلول
volScalarField yy_((centres & vector(0,1,0))),     // تعریف مختصاتy مرکز سلول
```

سپس متغیرهای نیروی حجمی در معادله ۲ را تعریف می‌کنیم. نحوه تعریف متغیرها در این نسخه از اپن‌فوم کمی تغییر یافته است. مثلا بعد از اسم هر متغیر یک علامت _ مشاهده می‌شود. این متغیرها و نحوه تعریف آنها عبارتند از:

- sw: متغیر برای ایجاد switch می‌باشد. این متغیر با نسبت دادن مقدار ۱ به سلول، پلاسما را به مختصاتی که توسط کاربر تعیین می‌شود اعمال می‌کند و برای خارج از مختصات داده شده مقدار ۰ را لحاظ می‌کند که به معنی عدم وجود پلاسما می‌باشد.

```
sw_
(
    IOobject
   (
      "sw",
```

```
            runTime().timeName(),
            mesh,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        mesh
    ),
```

- E: میدان الکتریکی می‌باشد. طبق فرمول ۱ داریم:

```
    E_
    (
        IOobject
        (
            "E",
            runTime().timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        sw_*(E0_-k1_*xx_-k2_*yy_)
    ),
```

- Ex: میدان الکتریکی در راستای x می‌باشد.

```
    Ex_
    (
        IOobject
        (
            "Ex",
            runTime().timeName,()
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        (mag(E_)*k2_)/sqrt((k1_*k1_)+(k2_*k2_))
    ),
```

- Ey: میدان الکتریکی در راستای y می‌باشد.

```
    Ey_
    (
```

```
        IOobject
        (
          "Ey",
          runTime().timeName(),
          mesh,
          IOobject::NO_READ,
          IOobject::AUTO_WRITE
        ),
        (mag(E_)*k1_)/sqrt((k1_*k1_)+(k2_*k2_))
    ),
```

- Fx: نیروی الکتریکی در راستای x می‌باشد.

```
    Fx_
    (
        IOobject
        (
          "Fx",
          runTime().timeName(),
          mesh,
          IOobject::NO_READ,
          IOobject::AUTO_WRITE
        ),
        nufrequency_*alpha_*rhoc_*ec_*deltat_*Ex_
    ),
```

- Fy: نیروی الکتریکی در راستای y می‌باشد.

```
    Fy_
    (
        IOobject
        (
          "Fy",
          runTime().timeName(),
          mesh,
          IOobject::NO_READ,
          IOobject::AUTO_WRITE
        ),
        nufrequency_*alpha_*rhoc_*ec_*deltat_*Ey_
    ),
```

- F: بردار نیروی الکتریکی ناشی از پلاسما می‌باشد.

```
F_
(
    IOobject
    (
      "F",
       runTime().timeName,()
       mesh,
       IOobject::NO_READ,
       IOobject::AUTO_WRITE
    ),
    Fx_*vector(1,0,0)+Fy_*vector(0,1,0)
),
```

متغیرهایی که مقادیر آنها در جدول ۱ داده‌اند در فایل transportProperties در پوشه constant در قسمت fluid فراخوانی می‌شوند. لذا همانطور که دو متغیر nu و rho در فایل icoFlow.C بعد از معرفی transportProperties فراخوانی شده‌اند، متغیرهای کمّی پلاسما نیز به همان صورت در زیر آنها فراخوانی می‌شوند.

```
nufrequency_(transportProperties_.lookup("nufrequency ")),
alpha_(transportProperties_.lookup("alpha")),
rhoc_(transportProperties_.lookup("rhoc")),
ec_(transportProperties_.lookup("ec")),
deltat_(transportProperties_.lookup("deltat")),
E0_(transportProperties_.lookup("E0")),
Eb_(transportProperties_.lookup("Eb")),
k1_(transportProperties_.lookup("k1")),
k2_(transportProperties_.lookup("k2"))
```

سپس معادله نیروی حجمی را بصورت زیر به معادله ممنتوم اضافه می‌کنیم:

```
    fvVectorMatrix UEqn
(
    fvm::ddt(U_)
+   fvm::div(phi_, U_)
-   fvm::laplacian(nu_, U_)
==    F_
);
```

۲-۲-۲. تغییرات فایل icoFlow.H

در فایل icoFlow.H در بخش Class icoFlow Declaration و قسمت Private data بعد از اعلان rho عبارات زیر را وارد می‌کنیم.

```
dimensionedScalar nufrequency_;
dimensionedScalar alpha_;
dimensionedScalar rhoc_;
dimensionedScalar ec_;
dimensionedScalar deltat_;
dimensionedScalar E0_;
dimensionedScalar Eb_;
dimensionedScalar k1_;
dimensionedScalar k2_;
```

همچنین در قسمت Member Functions بعد از اعلان rho عبارات زیر را وارد می‌کنیم.

```
const dimensionedScalar& nufrequency()
{
    return nufrequency_;
}

const dimensionedScalar& alpha()
{
    return alpha_;
}

const dimensionedScalar& rhoc()
{
    return rhoc_;
}

const dimensionedScalar& ec()
{
    return ec_;
}

const dimensionedScalar& deltat()
{
    return deltat_;
}
```

```
const dimensionedScalar& E0()
{
    return E0_;
}

const dimensionedScalar& Eb()
{
    return Eb_;
}

const dimensionedScalar& k1()
{
    return k1_;
}

const dimensionedScalar& k2()
{
    return k2_;
}
```

حال حلگر جدید آماده کامپایل می‌باشد. به آدرس FluidStructureInteraction /src رفته و ابتدا دستور Allwclean/. و سپس Allwmake/. را زده و منتظر کامپایل حلگر جدید می‌مانیم. اکنون باید از حلگر جدید در تست کیس قبلی استفاده نماییم که در بخش بعد توضیح داده شده است.

۲-۳. اجرای حلگر جدید برای تست کیس ایرفویل

برای اجرای حلگر تغییریافته در تست کیس قبلی ابتدا باید محل اعمال نیروی پلاسما را تعیین نماییم. این کار از طریق فایل setFieldDict در پوشه system در قسمت fluid صورت می‌پذیرد. دستور setFields به یک مجموعه انتخاب‌شده از سلول‌ها یا پچ‌های صفحات از طریق یک دیکشنری مقادیر اعمال می‌کند.

```
defaultFieldValues
(
    volScalarFieldValue sw 0
```

```
);
regions
(
    boxToCell
    {
        box (0.980186 0 -1) (1 0.005 0);
        fieldValues
        (
            volScalarFieldValue sw 1
        );
    }
);
```

این فایل محل اعمال نیروی پلاسما در لبه فرار ایرفویل می‌باشد. متغیر sw که قبلا تعریف شده است در تمامی حوزه حل به جز مختصات محدوده داده شده مقدار صفر دارد.

همچنین یک فایل sw در پوشه 0 نیاز می‌باشد.

```
dimensions      [0 1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    frontAndBack
    {
        type            empty;
    }

    airfoil
    {
        type            zeroGradient;
    }
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    upAndDown
    {
        type            symmetryPlane
```

```
    }
}
```

حال کیس آماده ران می‌باشد. لذا ابتدا دستور setFields را وارد کرده تا پلاسما به ناحیه موردنظر اعمال شود و سپس دستور weakFsiFoam را برای اجرای مسئله وارد می‌کنیم.

منابع

[1] Turek, S. and Hron, J., “Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow”, Springer, 2006.
[2] Bouchmal, A., “Modeling of Dielectric-Barrier Discharge Actuator Implementation, validation and generalization of an electrostatic model”, March 2011, Delft University of Technology.
[3] Jayaraman, B. and Shyy, W., “Modeling of dielectric barrier discharge-induced fluid dynamics and heat transfer”, Progress in Aerospace Sciences, 44:139-191, 2008.