مدلسازی سیال با ویژگی های متغیر با دما

(حل میدان دما در داخل حفره)

در بسیاری از فرآیندهای فیزیکی تغییر دما به حدی است که فرض سیال با ویژگی های فیزیکی ثابت صحیح نیست. در این موارد معمولا از یک مقدار میانگین برای مدلسازی استفاده میشود. اما مدل دقیق تر میتواند تغییرات ویژگی های فیزیکی با دما را نیز در نظر بگیرد.

۱-۱- تعریف مساله

در این مساله یک سیال تراکم ناپذیر درون حفره ای دوبعدی قرار دارد. ویسکوزیته دینامیکی سیال برابر با $0.01\frac{m^2}{s}$ است و صفحه ی بالایی حفره با سرعت ۱ متر بر ثانیه به سمت راست حرکت میکند.

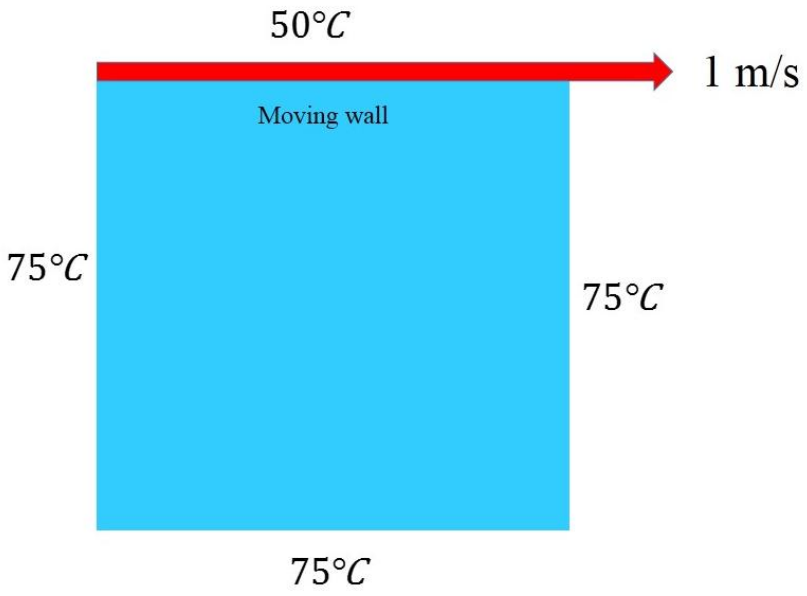
$$D = D_0 - \alpha(T - T_0)$$

$$D_0 = 0.01$$

$$T_0 = 50°C$$

$$a = 0.001\frac{m^2}{s°c}$$

$$\frac{\partial T}{\partial t} + \nabla\cdot(\vec{U}T) - \nabla\cdot(D\nabla T) = 0$$



ما میدان فشار و جریان را برای این مساله در فصل های گذشته با استفاده از حلگر icoFoam بدست آوردیم. در این فصل خواسته ما توزیع دما در این مساله با شرایطی است که در تصویر بالا آمده است. در این مساله ضریب پخش حرارتی رابطه ی خطی با دما به صورت زیر دارد:

$$D = D_0 - \alpha(T - T_0) \qquad (۱-۱)$$

که در اینجا D_0 ضریب پخش حرارتی و T_0 دمای مرجع است.

۱-۲- استراتژی حل

ما در این مساله دنبال حلگری هستیم که علاوه بر حل جریان سیال غیرقابل تراکم و انتقال حرارت، تغییرات ضریب پخش حرارتی نسبت به دما را نیز در نظر بگیرد. با جستجو در حلگرهایی که به حل جریان غیر قابل تراکم میپردازند به این نکته پی میبریم که برخی حلگرها نظیر pimpleFoam و icoFoam به حل جریان غیرقابل تراکم میپردازند ولی توانایی حل معادله انتقال حرارت در آنها وجود ندارد. میبینیم که حلگری که هم تراکم ناپذیر باشد و هم به حل انتقال حرارت بپردازد عملا وجود ندارد. بنابراین تنها راه حل ما ایجاد یک حلگر جدید بر پایه حلگرهای موجود است که تمام خواسته های ما را براورده کند. همانطور که میدانید OpenFOAM یک نرم افزار متن باز است و ما کاملا به کدهای منبع دسترسی داریم.

اولین قدم در نوشتن حلگر جدید استخراج معادلات حاکم است. تا پیش از این ما فقط به عنوان یک کاربر از اپن فوم استفاده کردیم و نیازی ندیدیم که معادلات جریان و انتقال حرارت و ... را به طور کامل بشناسیم. اما اکنون چون باید معادله جدیدی را به حلگر اضافه کنید و حلگر جدیدی بنویسید باید با معادلات و نحوه اعمال آنها در اپن فوم آشنا شوید.

برای حل این مساله ما باید قدم های زیر را برداریم:

- استخراج معادلات حاکم بر مساله.
- کدخوانی حلگر icoFoam به عنوان نزدیکترین حلگر به مساله ما.
- نوشتن حلگر جدید که معادله دما را در خود حل میکند.
- حل مساله با حلگر جدید.

۱-۳- استخراج معادلات

$$\nabla \cdot (\vec{U}) = 0 \qquad (۱-۲)$$

$$\frac{\partial \vec{U}}{\partial t} + \nabla \cdot (\vec{U}\vec{U}) - \nabla \cdot (\upsilon \nabla \vec{U}) = -\frac{1}{\rho}\nabla P \qquad (۱-۳)$$

$$\frac{\partial T}{\partial t} + \nabla \cdot (\vec{U}T) - \nabla \cdot (D\nabla T) = 0 \qquad (۱-۴)$$

که ρ چگالی، υ ویسکوزیته سینماتیکی و D ضریب پخش حرارتی است.

۱-۴- کدخوانی

برای دستیابی به کدهای منبع به زیر پوشه application به آدرس زیر بروید:

Opt/openfoam220/applications/solvers/incompressible/icoFoam

هر حلگر از یک فایل سورس با پسوند C. و تعدادی فایل هدر با پسوند H. تشکیل شده است و حاوی پوشه Make است. در فایل make دو فایل با نام های files و options قرار دارد.

فایل files: این فایل از دو بخش تشکیل شده است. در بخش نخست فهرستی از فایل های سورسی (فایل های با پسوند C.) که باید کامپایل شوند آورده شده و در بخش دوم، نام فایل اجرایی (باینری) قرار دارد و مکان ذخیره سازی فایل باینری قرار دارد و مکان ذخیره سازی فایل باینری مشخص شده است.

icoFoam.C

EXE = $(FOAM_APPBIN)/icoFoam

فایل باینری برنامه در آدرس زیر قرار دارد:

/opt/openfoam220/platforms/linuxGccDPOpt/bin

فایل options: این فایل از دو بخش تشکیل شده است. در بخش نخست، آدرس فایل های include و در بخش دوم فهرستی از کتابخانه هایی که در طول برنامه مورد استفاده قرار میگیرند آورده شده است.

EXE_INC = \

-I $ (LIB_SRC) / finiteVolume/lnInclude

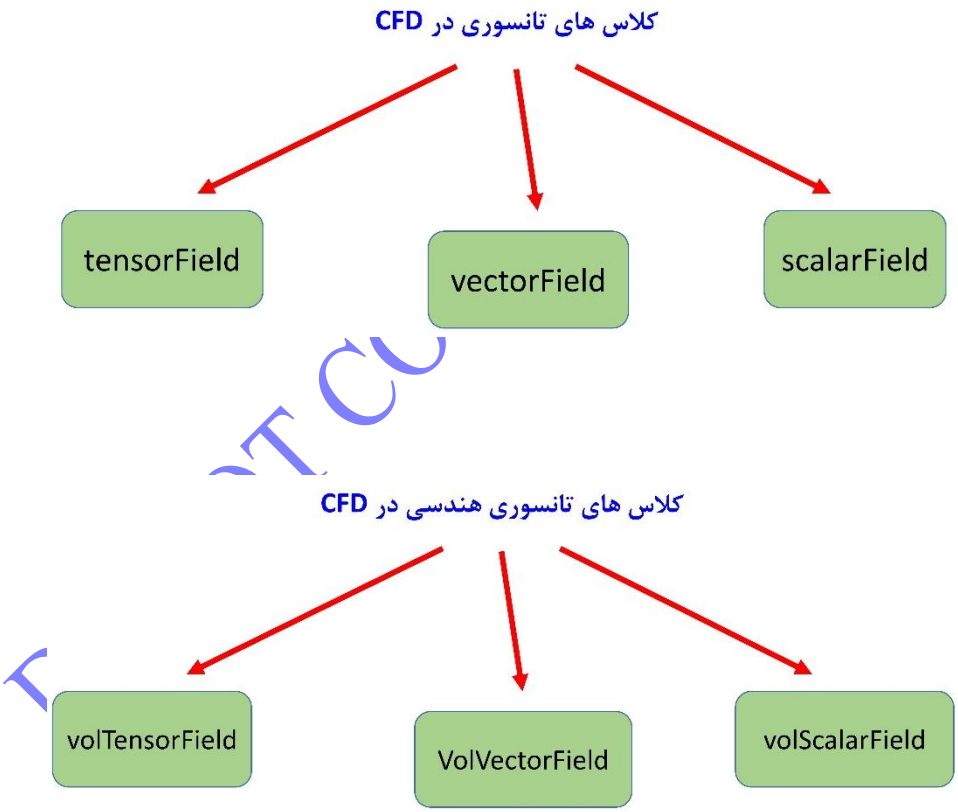
EXE_LIBS = \

-lfiniteVolume

۱-۵- اعمال میدان های تانسوری :

بیشتر بخش های دینامیک سیالات می تواند توسط محاسبات تانسوری شرح داده شود.که مرتبه ی این تانسورهای میتواند صفر و ۱ و ۲ باشد که به ترتیب نشانگر اسکالر، بردار و تانسور می باشند. بنابراین سه کلاس پایه ای ایجاد شده است :scalarFields و vectorField و tensorFields .

سطح بعدی تانسورها به عنوان "میدان های تانسور هندسی" یا geometricTensorFields شناخته میشوند و شامل اطلاعات موقعیتی نیز می شوند که کلاس های قبلی فاقد آن هستند. این سری از کلاس ها برای سه مرتبه از تانسورها به صورت volScalarField ، volVectorField و volTensorField اعمال شده اند.



ذکر این نکته ضروری است که باید مابین به عنوان مثال میدان اسکالر (scalarField) و volScalarField تفاوت قائل شد هرچند که،به کامپایلر اجازه داده شده است که scalarField+volScalarField را به عنوان یک عمل قبول کند،که مناسب نمی باشد، بنابراین به جای آن از تقابل داده ها با یکدیگر استفاده شده است.

معمولا دو نوع از کلاس های مشتق تانسوری در این فوم اعمال شده است :

Fvc (**finite volume calculus**):که داده های از پیش تعیین شده را به صورت صریح ارزیابی می کند (explicit) و یک میدان تانسوری هندسی را بر میگرداند.

Fvm (**finite volume method**): که یک تعریف ماتریسی از عملیات را ارایه میدهد.

به بیان ساده تر fvc ترم های اضافه شده به معادله را به عنوان ترم چشمه در نظر میگیرد و با آن ها به صورت explicit برخورد میکند یعنی از گام زمانی قبل یا تکرار قبلی آن ،استفاده می کند.

Fvm ترم های اضافی وابسته به متغیر را درون تانسور ضرایب قطری جا می دهد و آن را به صورت حل همزمان یعنی implicit حل میکند.

Fvc → Finite volume calculus → explicit

Fvm → Finite volume method → implicit

معادله غیر خطی → $Ax = B$

همه ی مشتق های تانسوری ممکن که در این فوم اعمال شده اند : $\frac{\partial}{\partial t}, \nabla\cdot, \nabla, \nabla\times$.به علاوه اپراتور لاپلاسین به طور مجزا تعریف شده است یعنی مجزا از اینکه لاپلاسین برابر $\nabla\cdot\nabla$ است، تعریف شده. یکی از مشکلات که در مسائل

عددی باید از پس آن برآییم انتخاب Scheme (طرح یا رویه) دیفرانسیل گیری است که برای محاسبه ی مشتق ها مورد استفاده قرار میگیرد.

مشتق زمانی $\frac{\partial}{\partial t}$ می تواندبه صورت زیر استفاده شود:

volVectorField dUdt = fvc::ddt(U , EI)

که ورودی دوم مشخص میکند که کدام scheme دیفرانسیل گیری استفاده شده است (در این کیس Euler Implicit استفاده شده).

در FVM ، ترم های دیورژانس به وسیله ی انتگرال سطحی بر روی حجم های کنترل δ_{v_i} معرفی شده است. بنابراین فراخوانی تابع دیورژانس به صورت div(phi , Q) است،که phi شار سطحی است، میدانی که مقادیر بر روی صفحات سلول ذخیره شده اند، و Q، مشخصه ای است که به وسیله ی این شار، منتقل شده است، و Q میدانی است که مقادیر آن بر روی مراکز سول ذخیره شده اند. به همین دلیل است که این عملیات نمیتواند به صورت div(phi*Q) انجام گیرد و فراخوانی گردد زیرا که شار phi به صورت میدان سطحی است و Q به صورت میدان حجمی .

عملگر لاپلاس به صورت یک فراخوانی جدید اعمال شده نسبت به عملگرهای دیورژانس و گرادیان چون تعریف عددی آن ها متفاوت است.

فرم های مختلفی از ترم چشمه نیز اعمال شده است. یک ترم چشمه میتواند صریح (Explicit) باشد در کیس های با نوع خاصی از معادلات که در این فرم ترم چشمه در سمت چپ و در قسمت ماتریس معلومات قرار میگیرد($Ax = B \quad in \quad B$)،یا میتواند به صورت ضمنی تعریف شود (implicit) با ورودی های ماتریسی که ضرایب معادلات بر روی ضرایب قطری ماتریس A اثر میکنند.

ساخت یک ترم چشمه ی صریح به وسیله ی عملیات + و – بدست می آید و به صورت fvm + volScalarField تعریف می شود.ساخت یک ترم چشمه ی ضمنی با استفاده از تابع Sp(a,Q) بدست می آید،که متغیر مستقل Q را در نظر میگیرد که باید حل شود.

مفهوم ترم های صریح و ضمنی

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\vec{U}\alpha) = \dot{m}'''\alpha$$

$$\frac{\partial T}{\partial t} + \nabla \cdot (\vec{U}T) - \nabla \cdot (D\nabla T) = \alpha(1-\alpha).\dot{m}''' \cdot H_{LG}$$

$$Ax = B$$

اثری از متغیر اصلی معادله که به صورت ماتریسی حل میشود در آن نیست و عدد بدست آمده از آن باید در سمت راست و ماتریس معلومات قرار گیرد. ← explicit

متغیر اصلی معادله در آن ترم چشمه وجود دارد و باید در ماتریس ضرایب قرار گیرد ← implicit

به عنوان یک مثال،معادلات پایستگی جرم را در نظر بگیرید:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\phi) = 0 \quad , \quad \phi = \rho U \qquad (۵-۱)$$

```
fvScalarMatrix    rhoEqn
(
    Fvm::ddt(rho) + fvc::div(phi)
);
```

که شار phi قبلا ارزیابی شده است،و این عبارت به وسیله ی دستور زیر حل میشود:

```
rhoEqn.solve();
```

برای کامل کردن مساله،اپراتور = = تعریف شده است تا برابری یا معادل بود نترم های دو طرف معادله را از نظر ریاضی بیان کند.این اپراتور سبب می شود که کد به صورت اتوماتیک معادله را بازچینی کند(تمام ترم های ضمنی به درون

ماتریس می روند و تمام ترم های صریح سهم بردار منبع میشوند.برای اینکه این عمل امکان پذیر باشد،اپراتورهای انتخاب شده باید اولویت پایین تری داشته باشند و این دلیل استفاده از = = است.

۱-۶- توضیح حلگر scalarTransportFoam:

یکی از ساده ترین حلگرها و یک نقطه ی شروع عالی در درک حلگر اپن فوم،حلگر scalarTransportFoam است،این حلگر،حل معادله ی advection-diffusion یک اسکالر ناپایا را امکان پذیر می کند:

$$\frac{\partial T}{\partial t}+\nabla\cdot(UT)-D_T\nabla^2T=0 \qquad (۶-۱)$$

که T کمیت دمای اسکالر،U میدان سرعت و D_T ضریب نفوذ است.

مفهوم کد	متن کد
خط ۱: این فراخوانی به منظور در دسترس داشتن ابزارهای fvm برای کاربردهای گسسته سازی و استفاده از ابزارهای CFD صورت می گیرد. خط ۲-۱۱ : تابع اصلی آغاز میشود.پارامترهای خط فرمان،متغیرهای زمانی و مش تنظیم میشوند. خط ۱۲: "createFields.H" شرایط اولیه را برای U و T و D_T میخواند (خطوط ۴۷-۱۰۷ را نگاه کنید).این فایل همچنین شامل createPhi.H نیز می باشد (خطوط ۱۰۸-۱۳۴) و که شارسطحی phi را می خواند و محاسبه میکند. خط۱۷: یک پیام را نشان میدهد که به معنای شروع محاسبات است. خط ۱۹: این فرمان ماکزیمم و مینیموم عدد کورانت را مشخص می کند. خط ۲۱-۳۸ : سیکل اصلی زمانی است که به وسیله ی runtime کنترل می شود. خط ۲۳: این فرمان زمان حقیقی را چاپ می کند.	1 #include "fvCFD.H" 2 3 // * * * * * * * * * * * * * * * * * * 4 5 int main(int argc, char *argv[ ]) 6 { 7 8 # include "setRootCase.H" 9 10 # include "createTime.H" 11 # include "createMesh.H" 12 # include "createFields.H" 13 14 15 // * * * * * * * * * * * * * * * * * * 16 17 Info<< "\nCalculating scalar transport\n" << endl; 18 19 # include "CourantNo.H" 20 21 for (runTime++; !runTime.end(); runTime++)

```
22      {
23          Info<< "Time = " <<
runTime.timeName() << nl << endl;
24
25          # include "readSIMPLEControls.H"
26
27              for (int nonOrth=0;
nonOrth<=nNonOrthCorr; nonOrth++)
28                  {
29                   solve
30                   (
31                      fvm::ddt(T)
32                   + fvm::div(phi, T)
33                   - fvm::laplacian(DT, T)
34                    );
35                  }
36
37              runTime.write();
38       }
39
40   Info<< "End\n" << endl;
41
42   return(0);
43  }
44
45
46
47 // ************************************ //
```

خط ۲۵: شامل readSIMPLEControls.H میشود که اجازه میدهد تا تعداد تصحیح کننده های نامتعامد،خوانده شود.

خط ۲۱-۳۸: این فرمان زمان حقیقی را چاپ می کند.

خط ۲۳: این فرمان زمان حقیقی را چاپ می کند.

خط ۲۷-۳۵: همانطور که قبلا شرح داده شده اسـت،ترم نفوذ و ترم لاپلاسین به وسیله ی گرادیان صفحه ای گسسته شده اند.این گسسته سازی به مصحح های نا متعامد نیاز دارد در کیس هایی که مش های نامتعامد استفاده شده است.سپس این حلقه به دفعاتی که مشخص شده است به وسیله ی nNonOrthCorr این اصلاح را اعمال می کند.توجه شود که تابع solve در همان زمان ،همان مساله را حل می کند.این نیاز به توضیح دارد که :

Fvm::ddt(T)
+ fvm::div(phi,T)
–fvm::laplacian(DT,T)

بیان می کند که سه سیستم از معادلات جمع آوری شده است،یکی برای هر ترم.هر ترم ماتریس،مقادیر حدسی و مقادیر سمت راست مخصوص به خود را دارد.چون متد fvm فقط در ماتریس شرکت میکند و T میدان حدسی است،این ترم ها زمانی که جمع میشوند،یک سیستم از معادلات را با سمت راست برابر با صفر تشکیل می دهند،زیرا اپراتور == استفاده نشده است که این نشان دهنده ی عدم وجود ترم چشمه است.در هرگام زمانی سمت راست شروع به صفر شدن می کند اما ما n گام اصلاح تعامد داریم.

خط ۳۷ : این خط نشان دهنده ی نوشتن میدان بر روی هارد دیسک است.

خط ۳۹-۴۳: نشان دهنده ی پایان محاسبات و برگرداندن کنترل به سیستم است.

۱-۷- ادامه توضیح حلگر icoFoam

icoFoam.C:

در پوشه icoFoam، فایل سورس icoFoam.C قرار دارد. این فایل بدنه ی اصلی حلگر را تشکیل میدهد. یکی از ویژگی های اپن فوم OpenFOAM استفاده هوشمندانه از زبان برنامه نویسی شی گرای C++ است. توجه کنید که اپن فوم نه کاملا برنامه نویسی C++ است نه کاملا به صورت نرم افزارهای تجاری مانند فلوئنت. بلکه اپن فوم به گونه ای مابین این دو حالت است و در واقع با استفاده هوشمندانه از کتابخانه های وسیع و کامل و فراخوانی آنها در مواقع نیاز یک برنامه نویسی ساده و جامع را ارائه میکند.

icoFoam با عبارت Int main (int arg() char*argv []) آغاز می شود که دو عبارت درون پرانتز تعداد پارامترها و پارامترهای حقیقی هستند که هنگام اجرای دستور icoFoam استفاده می شوند.

کیس به وسیله هدرهایی که قبل از گام زمانی آمده است مقدار دهی اولیه می شود که همه این هدرها غیر از CreateFields.H از بخش src فراخوانی می شود.

CreateFields.H در دایرکتوری icoFoam واقع شده است و همه متغیرهایی را که در icoFoam استفاده می شوند مقدار دهی اولیه می کند و همچنین تعریف متغیرها در این فایل انجام می گیرد.

در خطوط readPisoControls.H و courantNO.H زیر دیکشنری PISO در فایل fvsolution در کیس مورد نظر خوانده می شود و عدد کورانت محاسبه شده و چاپ می شود.

ایجاد ماتریس معادلات خطی:

```
fvVectorMatrix      UEqn
(
    fvm:: ddt(U)
  +fvm::div(phi,U)
   -fvm:: laplacian(nu,U)
```

fvm: ترم های ضمنی (implicit)

Fvc: ترم های صریح (explicit)