知乎 首发于 **有所思**

用GdbOF追踪laplacian函数在OpenFOAM中的定义过程

陈与论
答案一直很简单

关注他

19 人赞同了该文章

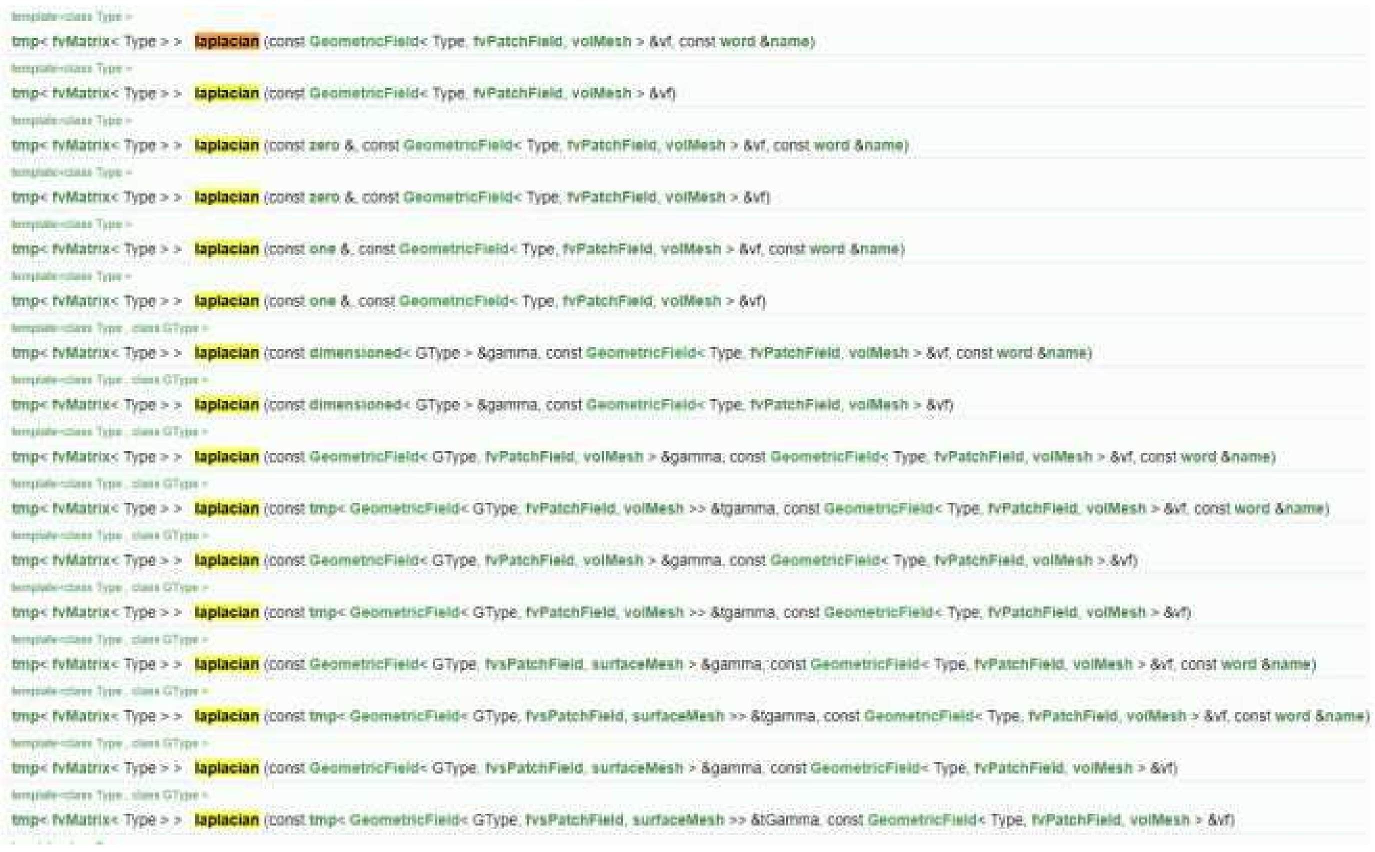
查看OpenFOAM源代码的时候经常遇到这样的问题，看着看着就绕回去了。今天我特地以laplacian函数为例，追踪一下其在OpenFOAM中的定义过程，搞清楚这些函数都是怎么定义的：

以laplacianFoam为例：

OpenFOAM-2.4.x/applications/solvers/basic/laplacianFoam/laplacianFoam.C

```
  solve
            (
                fvm::ddt(T) - fvm::laplacian(DT, T)
            );
```

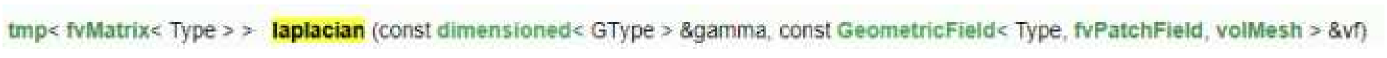
1. 找fvm命名空间下的laplacian函数：



```
template<class Type >
tmp< fvMatrix< Type > >  laplacian (const GeometricField< Type, fvPatchField, volMesh > &vf, const word &name)
template<class Type >
tmp< fvMatrix< Type > >  laplacian (const GeometricField< Type, fvPatchField, volMesh > &vf)
template<class Type >
tmp< fvMatrix< Type > >  laplacian (const zero &, const GeometricField< Type, fvPatchField, volMesh > &vf, const word &name)
template<class Type >
tmp< fvMatrix< Type > >  laplacian (const zero &, const GeometricField< Type, fvPatchField, volMesh > &vf)
template<class Type >
tmp< fvMatrix< Type > >  laplacian (const one &, const GeometricField< Type, fvPatchField, volMesh > &vf, const word &name)
template<class Type >
tmp< fvMatrix< Type > >  laplacian (const one &, const GeometricField< Type, fvPatchField, volMesh > &vf)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const dimensioned< GType > &gamma, const GeometricField< Type, fvPatchField, volMesh > &vf, const word &name)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const dimensioned< GType > &gamma, const GeometricField< Type, fvPatchField, volMesh > &vf)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const GeometricField< GType, fvPatchField, volMesh > &gamma, const GeometricField< Type, fvPatchField, volMesh > &vf, const word &name)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const tmp< GeometricField< GType, fvPatchField, volMesh >> &tgamma, const GeometricField< Type, fvPatchField, volMesh > &vf, const word &name)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const GeometricField< GType, fvPatchField, volMesh > &gamma, const GeometricField< Type, fvPatchField, volMesh > &vf)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const tmp< GeometricField< GType, fvPatchField, volMesh >> &tgamma, const GeometricField< Type, fvPatchField, volMesh > &vf)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const GeometricField< GType, fvsPatchField, surfaceMesh > &gamma, const GeometricField< Type, fvPatchField, volMesh > &vf, const word &name)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const tmp< GeometricField< GType, fvsPatchField, surfaceMesh >> &tgamma, const GeometricField< Type, fvPatchField, volMesh > &vf, const word &name)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const GeometricField< GType, fvsPatchField, surfaceMesh > &gamma, const GeometricField< Type, fvPatchField, volMesh > &vf)
template<class Type , class GType >
tmp< fvMatrix< Type > >  laplacian (const tmp< GeometricField< GType, fvsPatchField, surfaceMesh >> &tGamma, const GeometricField< Type, fvPatchField, volMesh > &vf)
```

▲ 赞同 19 ▼ 11 条评论 分享 喜欢 收藏 申请转载 …

2. 我们要找的fvm::laplacian(k,T)中，k是**dimensioned**类，T是**GeometricField**类，所以应该选这个：

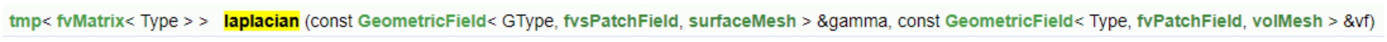

tmp< fvMatrix< Type > > laplacian (const dimensioned< GType > &gamma, const GeometricField< Type, fvPatchField, volMesh > &vf)

跳转到相关页面：

```
template<class Type, class GType>
tmp<fvMatrix<Type>>
laplacian
(
    const dimensioned<GType>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    const GeometricField<GType, fvsPatchField, surfaceMesh> Gamma
    (
        IOobject
        (
            gamma.name(),
            vf.instance(),
            vf.mesh(),
            IOobject::NO_READ
        ),
        vf.mesh(),
        gamma
    );

    return fvm::laplacian(Gamma, vf);
}
```

可以看出最后返回到fvm::laplacian(Gamma, vf)，就是说这个函数是个空壳，干活的是别的函数；

3. 继续找


tmp< fvMatrix< Type > > laplacian (const GeometricField< GType, fvsPatchField, surfaceMesh > &gamma, const GeometricField< Type, fvPatchField, volMesh > &vf)

```
template<class Type, class GType>
```

▲ 赞同 19 ▼ 11 条评论 分享 喜欢 收藏 申请转载 ···

```
    const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    return fvm::laplacian
    (
        gamma,
        vf,
        "laplacian(" + gamma.name() + ',' + vf.name() + ')'
    );
}
```

这里有：

```
"laplacian(" + gamma.name() + ',' + vf.name() + ')'
=
laplacian(DT,T)
```

指的就是fvSchemes文件里要定义的差分格式。

fvSchemes：

```
laplacianSchemes
{
    default         none;
    laplacian(DT,T) Gauss linear corrected;
}
```

4. 继续找

tmp< fvMatrix< Type > > laplacian (const GeometricField< GType, fvPatchField, volMesh > &gamma, const GeometricField< Type, fvPatchField, volMesh > &vf, const word &name)

```
template<class Type, class GType>
tmp<fvMatrix<Type> >
laplacian
(
    const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf,
    const word& name
```

```
        vf.mesh(),
        vf.mesh().laplacianScheme(name)
    )().fvmLaplacian(gamma, vf);
}
```

可以看出前面2-3都是空壳，最终都会跳转到4这里，然后转另一个函数laplacianScheme

5. laplacianScheme::New

static tmp< laplacianScheme< Type, GType > > **New** (const fvMesh &mesh, Istream &schemeData)
Return a pointer to a new laplacianScheme created on freestore. More...

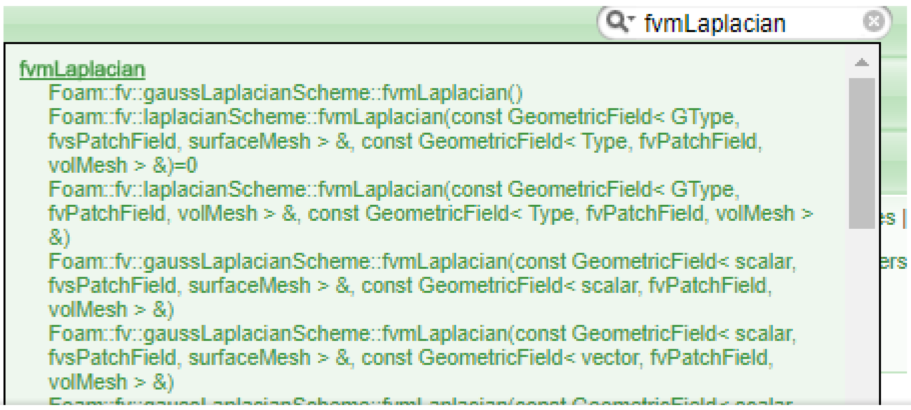
```
template<class Type, class GType>
tmp<laplacianScheme<Type, GType> > laplacianScheme<Type, GType>::New
(
    const fvMesh& mesh,
    Istream& schemeData
)
{
    if (fv::debug)
    {
        Info<< "laplacianScheme<Type, GType>::New(const fvMesh&, Istream&) : "
               "constructing laplacianScheme<Type, GType>"
            << endl;
    }

    if (schemeData.eof())
    {
        FatalIOErrorIn
        (
            "laplacianScheme<Type, GType>::New(const fvMesh&, Istream&)",
            schemeData
        )   << "Laplacian scheme not specified" << endl << endl
            << "Valid laplacian schemes are :" << endl
            << IstreamConstructorTablePtr_->sortedToc()
            << exit(FatalIOError);
    }

    const word schemeName(schemeData);

    {
        FatalIOErrorIn
        (
            "laplacianScheme<Type, GType>::New(const fvMesh&, Istream&)",
            schemeData
        )   << "Unknown laplacian scheme " << schemeName << nl << nl
            << "Valid laplacian schemes are :" << endl
            << IstreamConstructorTablePtr_->sortedToc()
            << exit(FatalIOError);
    }

    return cstrIter()(mesh, schemeData);
}
```

可以看出cstrIter返回了一个关于定义过差分方法的laplacianScheme对象。

6. fvmLaplacian(gamma, vf)

```
template<class Type, class GType>
tmp<fvMatrix<Type> >
laplacianScheme<Type, GType>::fvmLaplacian
(
    const GeometricField<GType, fvPatchField, volMesh>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    return fvmLaplacian(tinterpGammaScheme_().interpolate(gamma)(), vf);
}
```

这里的fvmLaplacian不是调用他自己，而是调用另一个命名空间下的同名函数：



fvmLaplacian
Foam::fv::gaussLaplacianScheme::fvmLaplacian()
Foam::fv::laplacianScheme::fvmLaplacian(const GeometricField< GType, fvsPatchField, surfaceMesh > &, const GeometricField< Type, fvPatchField, volMesh > &)=0
Foam::fv::laplacianScheme::fvmLaplacian(const GeometricField< GType, fvPatchField, volMesh > &, const GeometricField< Type, fvPatchField, volMesh > &)
Foam::fv::gaussLaplacianScheme::fvmLaplacian(const GeometricField< scalar, fvsPatchField, surfaceMesh > &, const GeometricField< scalar, fvPatchField, volMesh > &)
Foam::fv::gaussLaplacianScheme::fvmLaplacian(const GeometricField< scalar, fvsPatchField, surfaceMesh > &, const GeometricField< vector, fvPatchField, volMesh > &)

cstrIter的定义是:

```
typename IstreamConstructorTable::iterator cstrIter =
IstreamConstructorTablePtr_->find(schemeName);
```

但是实际上我没找到IstreamConstructorTable这个对象。在源代码里找到的关于IstreamConstructorTable的代码都是“调用”，没有“定义”的部分。

我意识到这是一个技术难点，需要攻克。

//--

通过查阅资料，我大致判断这是C++中的一种被称作“**runtime construction**”的操作。也就是编译的时候不构建对象，运行的时候再根据外部文件内容定义其结构的方法。那么IstreamConstructorTable可能就是使用这种方法会自动定义出来的一个对象。其原型可能就是Istream类。假如换一个A类使用runtime construction，就会自动产生一个名为AConstructorTable的类。

为了求证，我翻阅了一些资料：

OpenFOAM guide/runTimeSelection mechanism
openfoamwiki.net

https://www.cfd-online.com/Forums/openfoam-...
www.cfd-online.com

还是看不太懂。以我目前的理解，大致是因为C++对面象对象编程支持得还是不够好，所以OpenFOAM用了一种比较迂回的方式实现了对于class的虚拟函数（virtual function）操作。这个迂回的方式就是构建指针（point）和哈希列表（Hash tables）。

> Any function defined with the virtual key word in a base class and redefined in its derived class is a virtual function. Without the virtual key word, the function is hidden.

> pointers to all the virtual functions owned by that class. When a virtual function is called, the correct function is found by following the vpointer and looking it up on the vtable.
> Unfortunately, a vtable cannot be constructed until the object in question is fully manifested. In other words, no virtual constructors.

看不懂定义不要紧，看得懂例子就自然懂了。抱着这样的心态，我继续看后面的例子。

https://openfoamwiki.net/index.php/OpenFOAM_guide/runTimeSelection_...
openfoamwiki.net

OpenFOAM guide/runTimeSelection mechanism

OpenFOAM guide/runTimeSelection mechanism
openfoamwiki.net

In Base.C (or BaseNew.C):

```
Foam::tmp<Foam::Base> Foam::Base::New(const dictionary& dict)
{
    // omitting error catching and debug statements

    word DerivedType(dict.lookup("type"));

    typename MrConstructorTable::iterator cstrIter
        = MrConstructorTablePtr_->find(DerivedType);

    return cstrIter()(dict);
}
```

可以看出，在4.3.4节中出现的这一段代码跟我们遇到的laplacianScheme::New基本是一致的。那么可以认为laplacianScheme.C就对应着文中的Base.C，其目的就是在构建好的哈希表中选择正确的scheme方法，真正不同的scheme方法如何操作数字，这方面的代码应该在Derived.C文件里。

> Lastly, all **Derived** classes have one extra line in their **Derived.C**:

```
Base::addMrConstructorToTable<DerivedType>
    addDerivedTypeMrConstructorToBaseTable_;
```

既然这样，我们就搜一搜github，看看哪些C文件里有这行文字。

laplacianScheme.H

```
laplacianScheme<Type, GType>::                                         \
        addIstreamConstructorToTable<SS<Type, GType> >                    \
    add##SS##Type##GType##IstreamConstructorToTable_;
```

找来找去只找到laplacianScheme自己的头文件里有这样一行定义。可能教程有错吧。

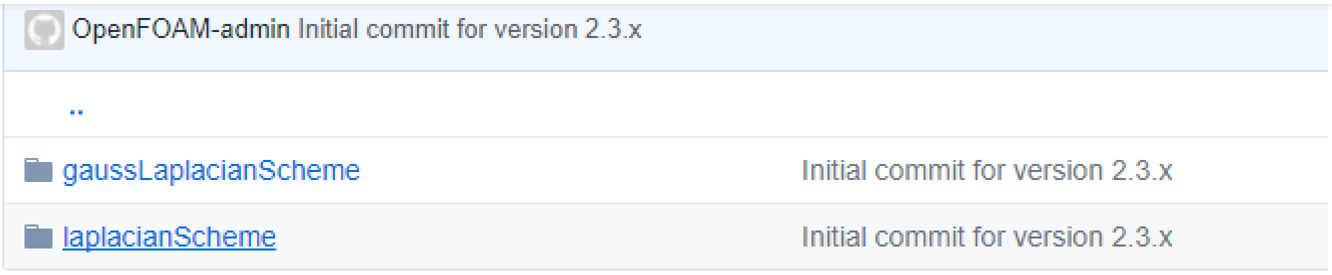
关于runtime construction的研究就到此为止了，以后有空再看吧。现在先继续laplacianScheme的研究。

//--

虽然还不知道这个cstrIter()(mesh, schemeData)返回的具体过程，但我已经知道了他想要返回的是一个具体的laplacianScheme。

```
$foamSearch $FOAM_TUTORIALS fvSchemes laplacianSchemes.default
    default         Gauss linear corrected;
    default         Gauss linear limited corrected 0.33;
    default         Gauss linear limited corrected 0.5;
    default         Gauss linear orthogonal;
    default         Gauss linear uncorrected;
```

查询可用的laplacianSchemes类型，发现5种都是Gauss类型。


OpenFOAM-admin Initial commit for version 2.3.x

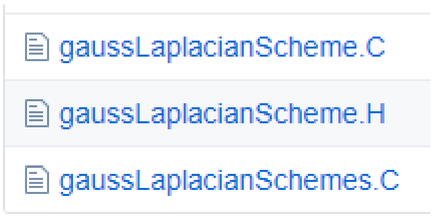
..

gaussLaplacianScheme	Initial commit for version 2.3.x
laplacianScheme	Initial commit for version 2.3.x

查看代码，发现只有一个gaussLaplacianScheme子文件，所以我认为这里的gaussLaplacianScheme就是laplacianScheme这个base类的derived类。

7. fvmLaplacian(tinterpGammaScheme_().interpolate(gamma)(), vf)

gaussLaplacianScheme文件夹下有3个文件：


gaussLaplacianScheme.C
gaussLaplacianScheme.H
gaussLaplacianSchemes.C

其中gaussLaplacianSchemes.C和gaussLaplacianScheme.C里面都定义了fvmLaplacian，其中一个定义矩阵的source项，一个定义矩阵的matrix项。

//

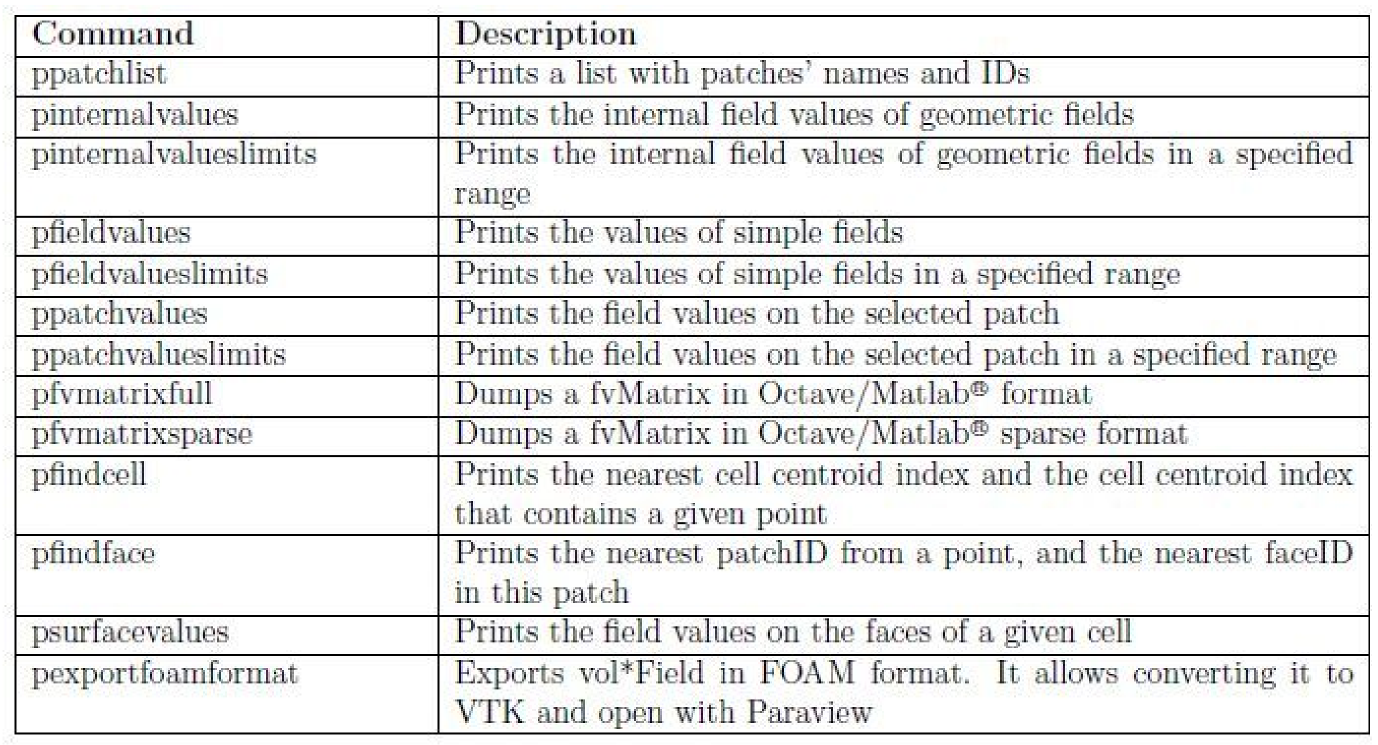
新的思路：

大家也看到了，光从源文件入手非常困难，花时间非常多。所以我打算换个思路，从debug入手，用gdbOF查看具体是调用哪些函数的。

几个常用gdb命令：

```
info locals //显示所有局部变量，OpenFOAM里别用，会爆炸
info variables/classes/functions/selectors            //顾名思义，但都别乱用，容易爆炸
info break  //显示所有breakpoints
info source //显示当前位置的文件调用关系，非常有用，尤其是被调用的文件是link的时候
n           //next，单个文件里的下一行
s           //step，下一步，能够跳转进入function内部
```

```
b           //设置breakpoint，基本格式：break 49;设定指定文件：b gaussLaplacianScheme.C:5
delete 1    //删除代号为1的断点
clear       //删除所有断点
pfvmatrixfull this matrix.txt    //专门用于输出矩阵，一般要先
ctrl+c      //卡住的时候用来跳出
set logging on/off          //用来将gdb命令显示的结果输出到默认的gdb.txt文件里
where       //告诉你你现在在哪个文件里
l           //list，显示附近的代码。可显示特点行号前后10行代码，如：list 12
watch       //表示设置一个监视点，当所指定的表达式EXPR的值被修改了，则程序会停止。
rwatch      //表示设置一个监视点，当所指定的表达式EXPR的值被读取了，则程序会停止。
awatch      //表示设置一个监视点，当所指定的表达式EXPR的值被读取或修改了，都会让程序停止。
p var       //显示变量var
p var@n     //显示数组变量var的前n个元素
whatis var  //显示变量var的类型
```



Command	Description
ppatchlist	Prints a list with patches' names and IDs
pinternalvalues	Prints the internal field values of geometric fields
pinternalvalueslimits	Prints the internal field values of geometric fields in a specified range
pfieldvalues	Prints the values of simple fields
pfieldvalueslimits	Prints the values of simple fields in a specified range
ppatchvalues	Prints the field values on the selected patch
ppatchvalueslimits	Prints the field values on the selected patch in a specified range
pfvmatrixfull	Dumps a fvMatrix in Octave/Matlab® format
pfvmatrixsparse	Dumps a fvMatrix in Octave/Matlab® sparse format
pfindcell	Prints the nearest cell centroid index and the cell centroid index that contains a given point
pfindface	Prints the nearest patchID from a point, and the nearest faceID in this patch
psurfacevalues	Prints the field values on the faces of a given cell
pexportfoamformat	Exports vol*Field in FOAM format. It allows converting it to VTK and open with Paraview

还有一个理论上很好用的命令:

Compiling and Injecting Code
sourceware.org

```
compile code
```

可以这样简单地插入一小段代码，输出当前某个变量的信息，但是这个命令在我的ubuntu16.04上总是运行不好，会提示No compiler support for language c++，所以我没用，以后可能会换个机子试一试。

还是对我们之前翻译文章里的那个简单的热传导的例子（源文件可下载）：

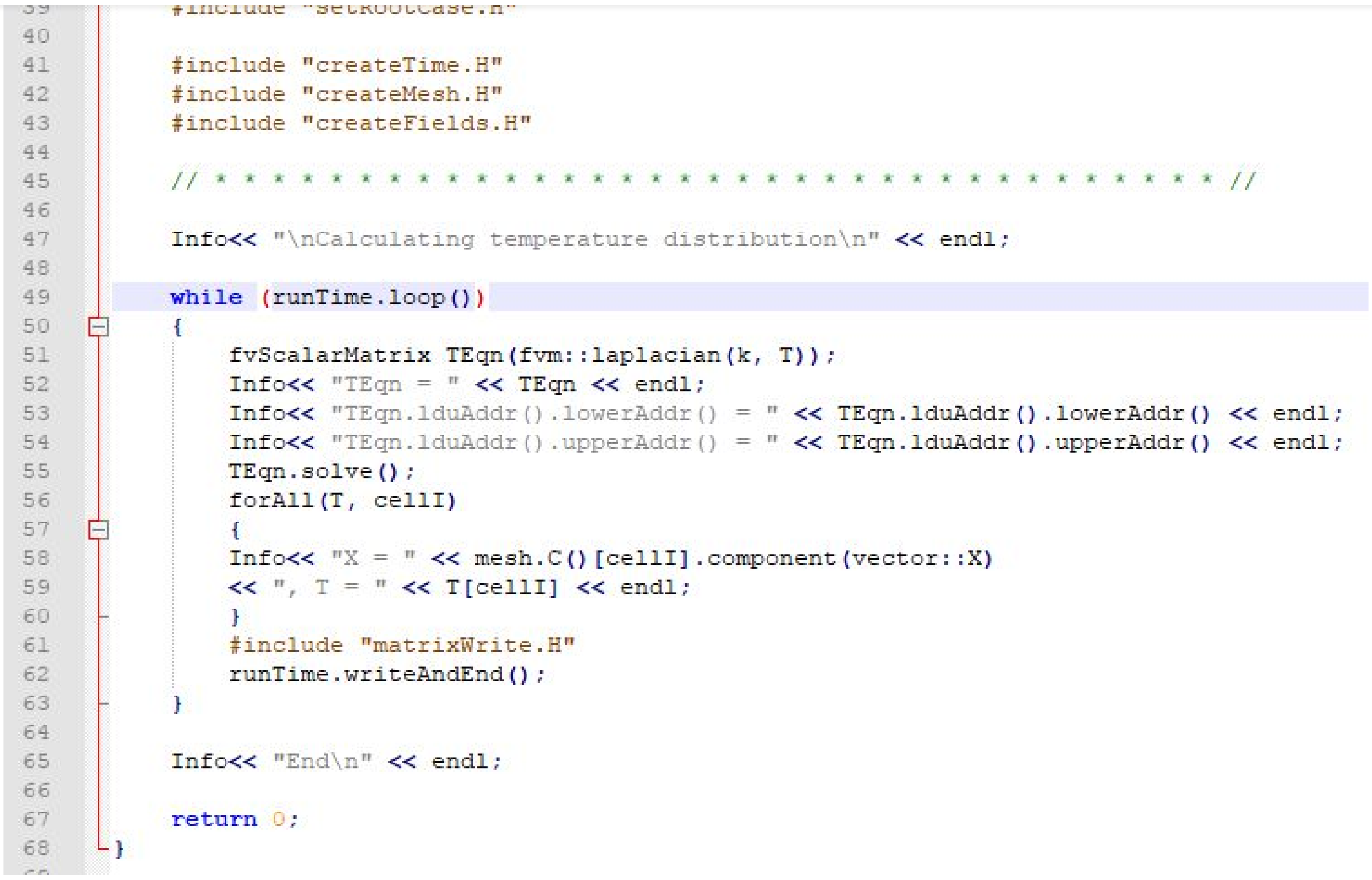
陈与论：【翻译】OpenFOAM中的空间离散化与矩阵系数【未完成】
zhuanlan.zhihu.com

1. 进入gdb模式：

```
yzhu25@ubuntu:~/Desktop/OpenFOAM_Practice1_20140316/testCase1$ gdb mylaplacianFoam
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from mylaplacianFoam...done.
(gdb)
```

2. 设置在49行断点：

```
(gdb) b 49
Breakpoint 1 at 0x41fb46: file mylaplacianFoam.C, line 49.
(gdb)
```



```
    #include "setRootCase.H"

    #include "createTime.H"
    #include "createMesh.H"
    #include "createFields.H"

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    Info<< "\nCalculating temperature distribution\n" << endl;

    while (runTime.loop())
    {
        fvScalarMatrix TEqn(fvm::laplacian(k, T));
        Info<< "TEqn = " << TEqn << endl;
        Info<< "TEqn.lduAddr().lowerAddr() = " << TEqn.lduAddr().lowerAddr() << endl;
        Info<< "TEqn.lduAddr().upperAddr() = " << TEqn.lduAddr().upperAddr() << endl;
        TEqn.solve();
        forAll(T, cellI)
        {
        Info<< "X = " << mesh.C()[cellI].component(vector::X)
        << ", T = " << T[cellI] << endl;
        }
        #include "matrixWrite.H"
        runTime.writeAndEnd();
    }

    Info<< "End\n" << endl;

    return 0;
}
```

3. 开始运行，直到断点停下

```
(gdb) run
Starting program: /home/yzhu25/OpenFOAM/yzhu25-2.4.0/platforms/linux64GccDPDebug/bin/m
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
/*---------------------------------------------------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
Build  : 2.4.0-dcea1e13ff76
Exec   : /home/yzhu25/OpenFOAM/yzhu25-2.4.0/platforms/linux64GccDPDebug/bin/mylaplacia
Date   : Jan 20 2018
Time   : 23:40:41
Host   : "ubuntu"
PID    : 4246
Case   : /home/yzhu25/Desktop/OpenFOAM_Practice1_20140316/testCase1
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Create time

Create mesh for time = 0

Reading field T

Reading transportProperties

Reading diffusivity k


Calculating temperature distribution


Breakpoint 1, main (argc=1, argv=0x7fffffffcdc8) at mylaplacianFoam.C:49
warning: Source file is more recent than executable.
49          while (runTime.loop())
(gdb)
```

4. 向前跑一行

```
(gdb) n
51              fvScalarMatrix TEqn(fvm::laplacian(k, T));
(gdb)
```

5. 此时我们想进入laplacian函数，看看如何操作的，所以向前跑一步

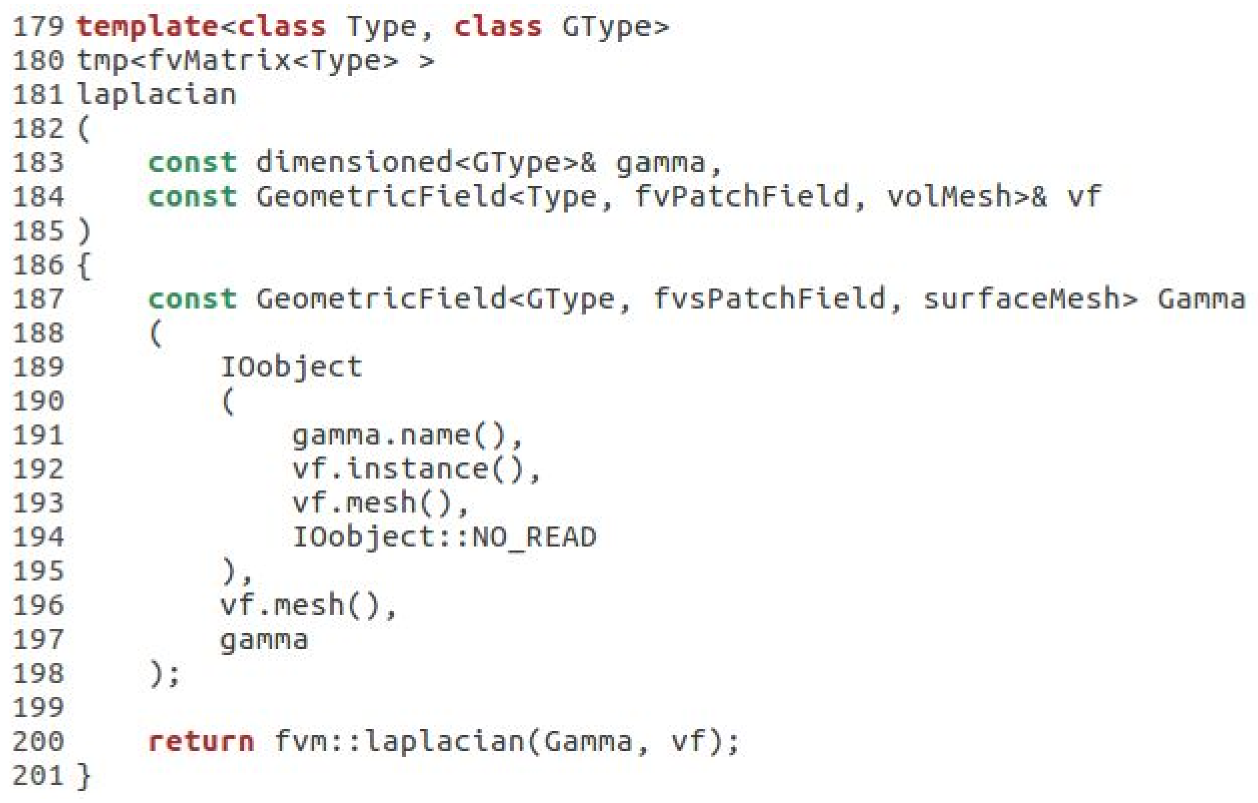
```
(gdb) s
Foam::fvm::laplacian<double, double> (gamma=..., vf=...) at /home/yzhu25/OpenFOAM/Open
181     laplacian
(gdb)
```

6. 用where和info source查询当前位置：

```
(gdb) info source
Current source file is /home/yzhu25/OpenFOAM/OpenFOAM-2.4.0/src/finiteVolume/lnInclude
Compilation directory is /home/yzhu25/Desktop/OpenFOAM_Practice1_20140316/testCase1/my
Located in /home/yzhu25/OpenFOAM/OpenFOAM-2.4.0/src/finiteVolume/finiteVolume/fvm/fvmL
Contains 342 lines.
Source language is c++.
Producer is GNU C++ 5.4.0 20160609 -m64 -mtune=generic -march=x86-64 -ggdb3 -O0 -ftemp
Compiled with DWARF 2 debugging format.
Includes preprocessor macro info.
(gdb)
```

可以看见，我们成功定位到了laplacian函数的位置：

```
/home/yzhu25/OpenFOAM/OpenFOAM-2.4.0/src/finiteVolume/finiteVolume/fvm/fvmLaplacian.C
```



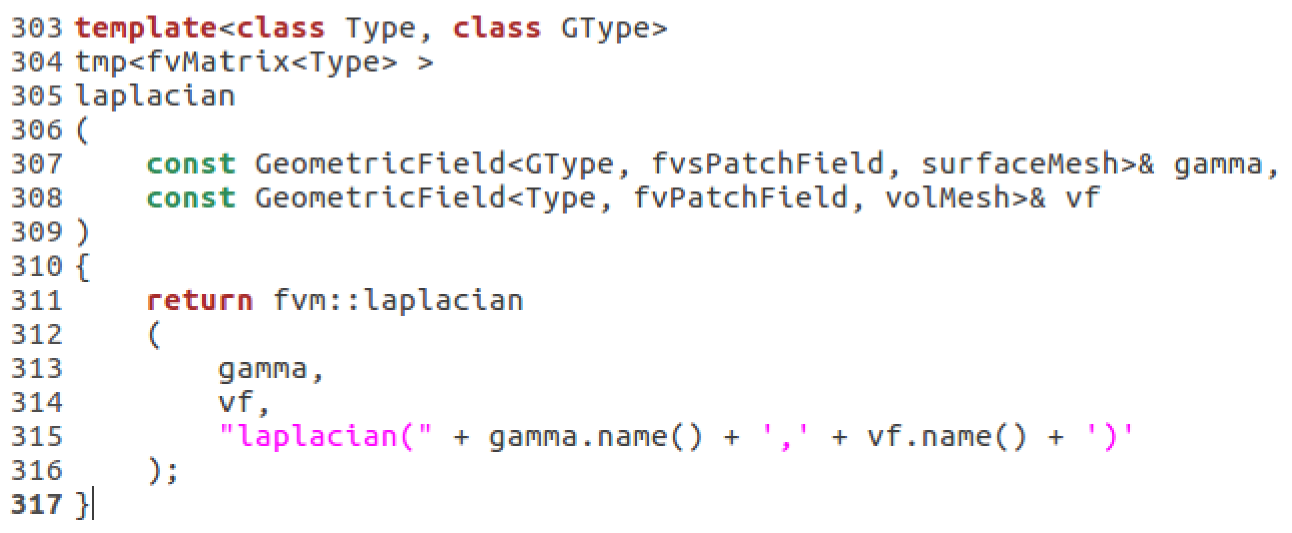
```
179 template<class Type, class GType>
180 tmp<fvMatrix<Type> >
181 laplacian
182 (
183     const dimensioned<GType>& gamma,
184     const GeometricField<Type, fvPatchField, volMesh>& vf
185 )
186 {
187     const GeometricField<GType, fvsPatchField, surfaceMesh> Gamma
188     (
189         IOobject
190         (
191             gamma.name(),
192             vf.instance(),
193             vf.mesh(),
194             IOobject::NO_READ
195         ),
196         vf.mesh(),
197         gamma
198     );
199
200     return fvm::laplacian(Gamma, vf);
201 }
```

7. 继续追：

```
(gdb) br 199
Breakpoint 2 at 0x4231b5: file /home/yzhu25/OpenFOAM/OpenFOAM-2.4.0/src/finiteVolume/l
```

```
200         return fvm::laplacian(Gamma, vf);
(gdb) s
Foam::fvm::laplacian<double, double> (gamma=..., vf=...) at /home/yzhu25/OpenFOAM/Open
305     laplacian
(gdb) where
#0  Foam::fvm::laplacian<double, double> (gamma=..., vf=...) at /home/yzhu25/OpenFOAM/
#1  0x00000000004231d5 in Foam::fvm::laplacian<double, double> (gamma=..., vf=...) at
#2  0x000000000041fb7d in main (argc=1, argv=0x7fffffffcdc8) at mylaplacianFoam.C:51
(gdb) list
300     }
301
302
303     template<class Type, class GType>
304     tmp<fvMatrix<Type> >
305     laplacian
306     (
307         const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
308         const GeometricField<Type, fvPatchField, volMesh>& vf
309     )
(gdb)
```

可以看见，正如我们之前预测的那样，我们进入了前文中第3步找到的函数。



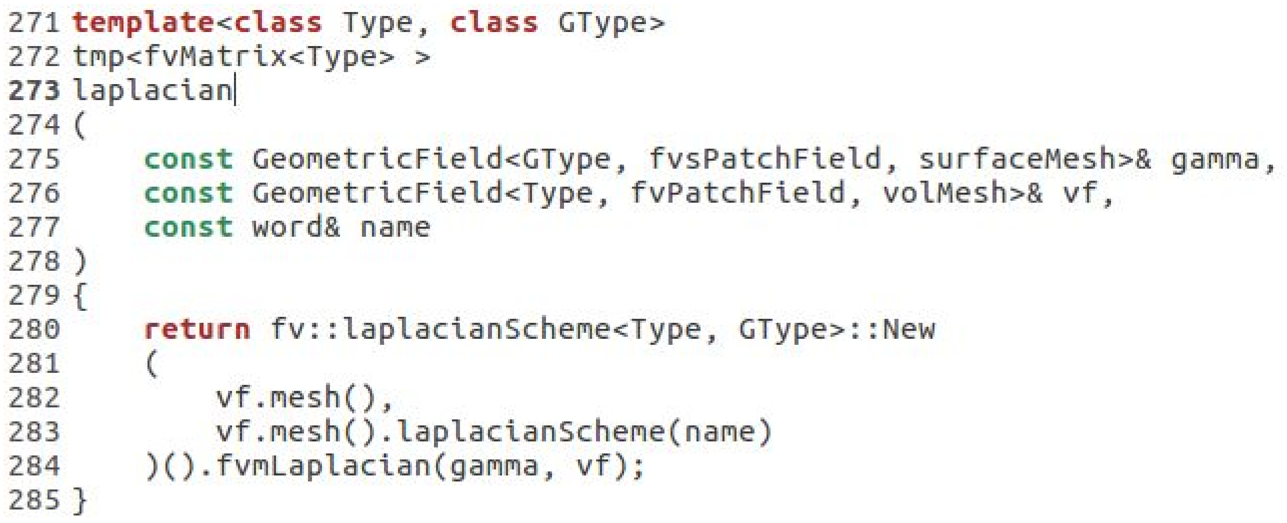
```
template<class Type, class GType>
tmp<fvMatrix<Type> >
laplacian
(
    const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    return fvm::laplacian
    (
        gamma,
        vf,
        "laplacian(" + gamma.name() + ',' + vf.name() + ')'
    );
}
```

8. 继续追：

这一部分比较长，我略去了，注意多用fin，不要用return。

```
271     template<class Type, class GType>
272     tmp<fvMatrix<Type> >
273     laplacian
274     (
275         const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
276         const GeometricField<Type, fvPatchField, volMesh>& vf,
277         const word& name
(gdb) where
#0  Foam::fvm::laplacian<double, double> (gamma=..., vf=..., name=...) at /home/yzhu25
#1  0x0000000000425bb4 in Foam::fvm::laplacian<double, double> (gamma=..., vf=...) at
#2  0x00000000004231d5 in Foam::fvm::laplacian<double, double> (gamma=..., vf=...) at
#3  0x000000000041fb7d in main (argc=1, argv=0x7fffffffcdc8) at mylaplacianFoam.C:51
(gdb)
```

可以看见，正如我们之前预测的那样，我们进入了前文中第4步找到的函数。



```
271 template<class Type, class GType>
272 tmp<fvMatrix<Type> >
273 laplacian
274 (
275     const GeometricField<GType, fvsPatchField, surfaceMesh>& gamma,
276     const GeometricField<Type, fvPatchField, volMesh>& vf,
277     const word& name
278 )
279 {
280     return fv::laplacianScheme<Type, GType>::New
281     (
282         vf.mesh(),
283         vf.mesh().laplacianScheme(name)
284     )().fvmLaplacian(gamma, vf);
285 }
```

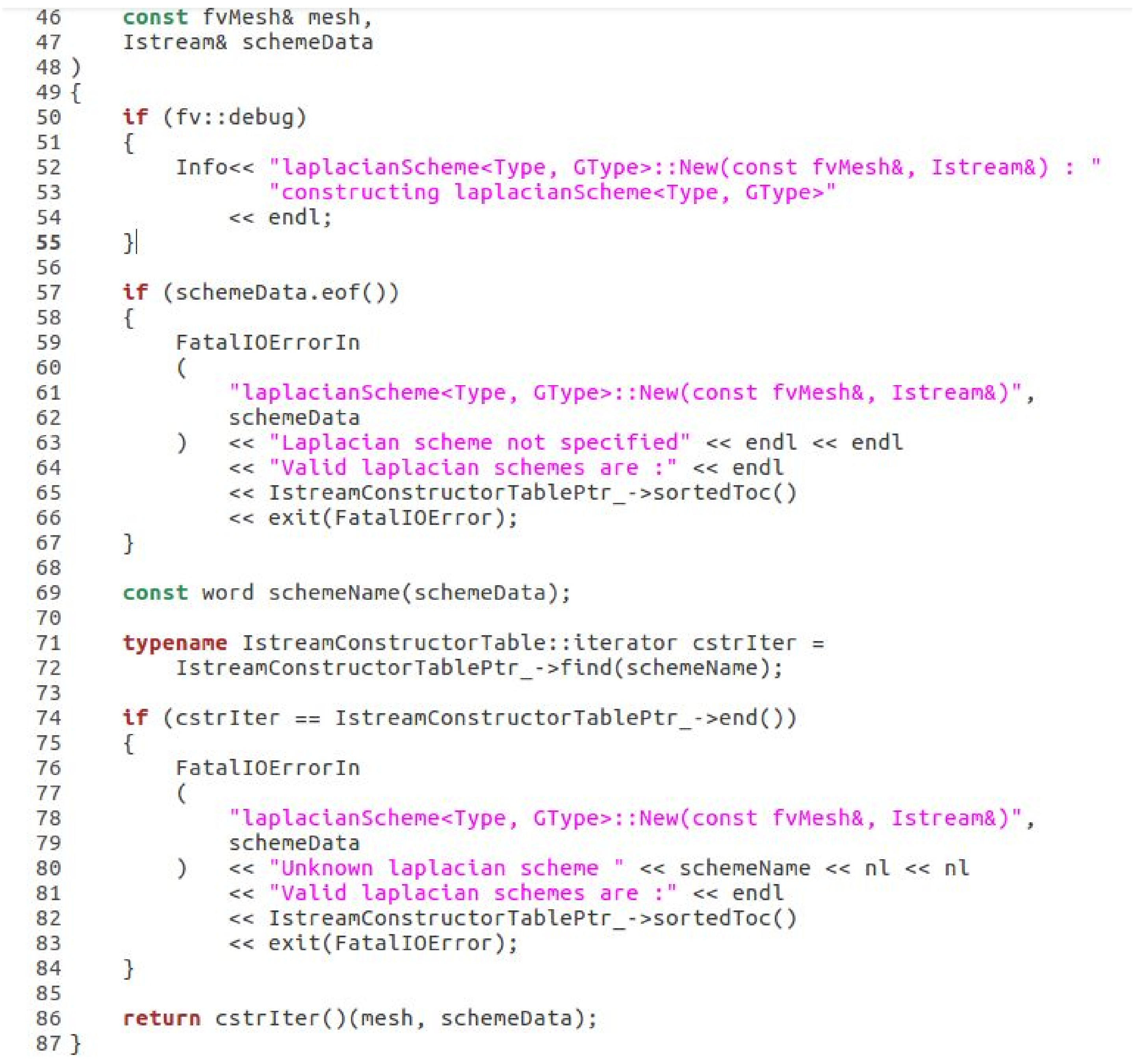
9. 继续追

```
(gdb) n
284         )().fvmLaplacian(gamma, vf);
(gdb) s
Foam::fv::laplacianScheme<double, double>::New (mesh=..., schemeData=...) at /home/yzh
44      tmp<laplacianScheme<Type, GType> > laplacianScheme<Type, GType>::New
(gdb) where
#0  Foam::fv::laplacianScheme<double, double>::New (mesh=..., schemeData=...) at /home
#1  0x0000000000428920 in Foam::fvm::laplacian<double, double> (gamma=..., vf=..., nam
#2  0x0000000000425bb4 in Foam::fvm::laplacian<double, double> (gamma=..., vf=...) at
```

```
Compilation directory is /home/yzhu25/Desktop/OpenFOAM_Practice1_20140316/testCase1/my
Located in /home/yzhu25/OpenFOAM/OpenFOAM-2.4.0/src/finiteVolume/finiteVolume/laplacia
Contains 131 lines.
Source language is c++.
Producer is GNU C++ 5.4.0 20160609 -m64 -mtune=generic -march=x86-64 -ggdb3 -O0 -ftemp
Compiled with DWARF 2 debugging format.
Includes preprocessor macro info.
(gdb)
```

可以看见，正如我们之前预测的那样，我们进入了前文中第5步找到的函数：

```
/src/finiteVolume/finiteVolume/laplacianSchemes/laplacianScheme/laplacianScheme.C
```

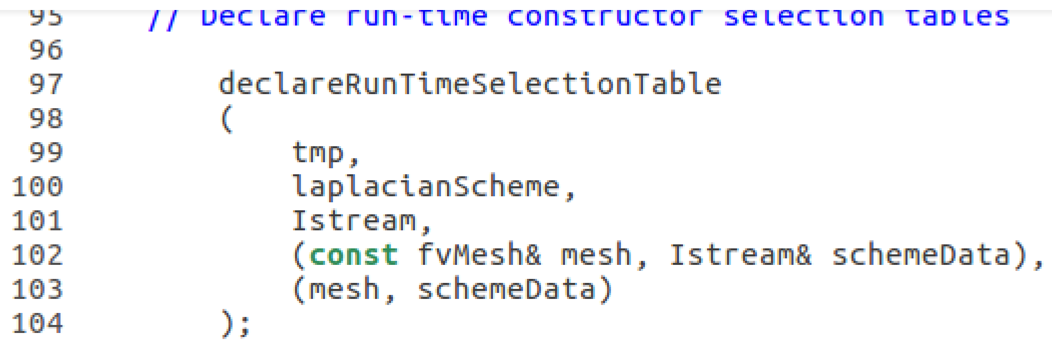


```
    const fvMesh& mesh,
    Istream& schemeData
)
{
    if (fv::debug)
    {
        Info<< "laplacianScheme<Type, GType>::New(const fvMesh&, Istream&) : "
               "constructing laplacianScheme<Type, GType>"
            << endl;
    }

    if (schemeData.eof())
    {
        FatalIOErrorIn
        (
            "laplacianScheme<Type, GType>::New(const fvMesh&, Istream&)",
            schemeData
        )   << "Laplacian scheme not specified" << endl << endl
            << "Valid laplacian schemes are :" << endl
            << IstreamConstructorTablePtr_->sortedToc()
            << exit(FatalIOError);
    }

    const word schemeName(schemeData);

    typename IstreamConstructorTable::iterator cstrIter =
        IstreamConstructorTablePtr_->find(schemeName);

    if (cstrIter == IstreamConstructorTablePtr_->end())
    {
        FatalIOErrorIn
        (
            "laplacianScheme<Type, GType>::New(const fvMesh&, Istream&)",
            schemeData
        )   << "Unknown laplacian scheme " << schemeName << nl << nl
            << "Valid laplacian schemes are :" << endl
            << IstreamConstructorTablePtr_->sortedToc()
            << exit(FatalIOError);
    }

    return cstrIter()(mesh, schemeData);
}
```

10. 继续追：

/src/OpenFOAM/lnInclude/HashTableI.H:377

```
373     template<class T, class Key, class Hash>
374     inline T&
375     Foam::HashTable<T, Key, Hash>::iterator::operator()()
376     {
377         return this->object();
378     }
```



```
 95     // Declare run-time constructor selection tables
 96
 97         declareRunTimeSelectionTable
 98         (
 99             tmp,
100             laplacianScheme,
101             Istream,
102             (const fvMesh& mesh, Istream& schemeData),
103             (mesh, schemeData)
104         );
```

接着以run-time constructor的方式构建了一个新的类，根据gdb反馈信息，这里97行的声明，等价于：

```
Foam::fv::laplacianScheme<double, double>::addIstreamConstructorToTable  \
<Foam::fv::gaussLaplacianScheme<double, double> >::New (mesh=..., schemeData=...) \
at lnInclude/laplacianScheme.H:97
```

这就调用了gaussLaplacianScheme.H文件：

```
Foam::fv::gaussLaplacianScheme<double, double>::gaussLaplacianScheme \
(this=0x7c4e40, mesh=..., is=...) at
finiteVolume/laplacianSchemes/gaussLaplacianScheme/gaussLaplacianScheme.H:91
```

这里 laplacianScheme是调用的laplacianScheme.H:118里的函数：

11. 继续追，进入gaussLaplacianSchemes.C

```
(gdb) s
```

可以看到，现在我们进入了之前观看源代码分析时停止的地方。当时在gaussLaplacian文件夹下的gaussLaplacianSchemes.C和gaussLaplacianScheme.C里面都定义了fvmLaplacian函数，我不能确定代码执行了哪一个。现在gdb告诉我，两个都执行了。C++先执行的是gaussLaplacianSchemes.C文件里的declareFvmLaplacianScalarGamma(scalar);语句。然后declareFvmLaplacianScalarGamma里面再套用gaussLaplacianScheme.C里定义的函数。

```
declareFvmLaplacianScalarGamma(scalar);
```

这个语句在该文件偏上一点的位置刚刚被定义过:

可以看出，这个函数里面有对source项的直接赋值。

对矩阵的直接赋值，在50行调用了函数fvmLaplacianUncorrected，这个函数则是在gaussLaplacianScheme.C里定义的：

至此laplacian函数的定义过程就完全破解了。与Moukalled等人出版的《The Finite Volume Method in Computational Fluid Dynamics》中的相关章节(8.10.2)对比一致。

知乎 首发于 **有所思**

编辑于 2018-12-20

「真诚赞赏，手留余香」

赞赏

还没有人赞赏，快来当第一个赞赏的人吧！

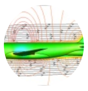
计算流体力学（CFD） openfoam

文章被以下专栏收录

有所思

与PhD们交流读博期间的所思所感，呼吁社会关爱

关注专栏



OpenFOAM学习记录

OpenFOAM在超算平台上的移植优化，欢迎交流

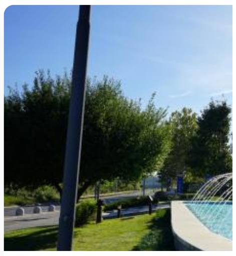
关注专栏

推荐阅读

如何在 OpenFOAM 中插入自己写的外部 python 函数

为了在RANS模型中插入一个自定义的函数，我试着在OpenFOAM中插入python代码，现在我把这个过程记录下来，供有需要的朋友们参考，也给自己留个记录。 首先是创建一个新的solver基本上只要涉...

陈与论



fvc::flux代码

海兮吾槊

▲ 赞同 19 ▼ 11 条评论 分享 喜欢 收藏 申请转载 ···

11 条评论 切换为时间排序

写下你的评论...

JingHe 2018-01-22

如此执着的钻研，着实令人佩服！我每次像这样追到一定程度感觉差不多是这样也就不管了哈哈，，，话说博主主要用OF做哪个方向模拟啊？我是做湍流和结冰的，刚开始博一，有机会还希望可以多多交流啊~

赞

陈与论 (作者) 回复 JingHe 2018-01-22

我也是做湍流的，多交流

赞

波尔 回复 JingHe 04-03

现在还做结冰方向吗？

赞

cjthermo 2018-06-21

刚想说你第六步的函数找错了，结果发现你换gdb找到了。你原先找的那个gamma的数据类型不对。给scalar gamma后openfoam直接构造surfaceScalarField 就不用从volScalarField差值了。

赞

毛博 2018-09-18

感谢答主！！ 正在分析数据结构，感觉被答主拉着跑进了新天地！！

赞

无色花开 2019-04-17

请问一下gdbOF支持Openfoam6吗

赞

无色花开 2019-04-18

不管支不支持我已经弃疗了... 换成openfoam v5.x就能用了..

赞

陈与论 (作者) 回复 无色花开 2019-04-19

▲ 赞同 19 ▼ 11 条评论 分享 喜欢 收藏 申请转载 ···

陈与论 (作者) 回复 无色花开 2019-04-19

还有5也不一定好，我室友去年用5算得结果不收敛，换成我的2.4就好了，所以我从来不升级。

赞

展开其他 1 条回复

Micro 09-21

深度好文，值得大家学习！！

赞

赞同 19 11 条评论 分享 喜欢 收藏 申请转载 ···