XFEM 扩展有限元教程

廉伟东

2011 年 6 月

目录

1 简介

1. xFiniteElement 这个类的理解，这个类的作用就是输入一个 mEntity 对象，他就能帮你返回对应的 xValueKey，记住不管怎么样，不管输入的什么样的对象点，线，面，体对象等（dim=0，1，2，3），他都会帮你返回点对象的 ValueKey, 也就是说那些对象都被最终分解为点对象，因为有限元计算关心的是节点信息，因此经过这类他就会帮你返回网格中节点对应的 xValueKey，而通过 xValueManager 就可以获得对应的 xDoubleValue。而整个网格对应的 xDoubleValue 和 xValueKey 都是通过 DeclareInterpolation 来创建的。

2. xClassRegion 这个类可以返回 xMesh（mMesh）网格中的点，线，面，体的集合，通过这些对象的 ntity_id，注意这个 entity_id，其实就是在 gmsh 生成网格时对应的 physical index。

3. xField 这个类是连接了 xSpace 和实际的 xDoubleValueManager，因此我们直接可以通过 xField.beginValues(mEntity) 和 xField.endValues(mEntity) 来返回 mEntity 对应的 xDoubleValue，不管 mEntity 的类型是点线面体。同理 xField.beginFcts(mEntity) 返回对应的 shape function。

4. xAssembler 类用来集成全局刚度矩阵和全局载荷向量，在该类中分为集成矩阵还是向量，若是集成矩阵时要提供 I，J，Value 信息，如果要集成载荷阵列则要提供 I,Value 信息。并且实际上 I，J 信息的提供只是需要一个在整体刚度矩阵中的位置信息即可，在当前 XFEM 程序中不采用整型数来编号，而是通过每个节点的 xDoubleValue 的指针，也就是 xDoubleValueManager 中的 Iterator 来进行。

5. xSpaceLagrange,xSpaceComposite,xSpaceFiltered 等对象的主要作用是获取形函数（Fcts）和键值的（femKey)。核心工作就是你可以输入一个 mEntity*e，注意这个 Entity 可以是体，面，线，点各种对象，他会根据定义的 Space 的自由度空间，返回对应节点的 ValueKey 以及 Function。也就是输入一个从大的对象返回对应点对象对应的节点自由度信息. 具体的实现参见代码 xSpaceLagrange::getKeys(mEntity* e,femKeys*keys)，其中涉及了对各种不同的几何对象如何获取对应几何对象的基本节点，然后再根据空间自由度，向量场还是标量场，来生成对应的自由度所对应的 xValueKey。所以仅仅是生成 xValueKey。

6. 当看到 xEval 是就是重载了括号运算符的一个函数对象，返回值的类型就是模板提供的参数。比如 xEvalConst 表示返回常数,xEvalFctAtPoint 返回给定的函数在某点的值；xEvalUnary 或者 xEvalBinary 返回指定的单一参数或者二参数谓词的返回值，也就是计算指定谓词的返回值。xValOperator and xValOperator 数值操作运算符。

7. xEval,xEvalConst,xEvalFctAtPoint,xEvalUnary,xEvalBinary, 当看到这些函数时，要在浅意识有几点认识。第一，只有 xEval 字样就表示要进行一个 Evaluation, 这个求值，是建立在给定每个单元的 xGeoElm m_appro, m_eteg 参数然后返回一个计算的值，计算返回值的类型就是 xEval<T> 中的 T 所指定的参数；第二，如果看到 xEvalUnary and xEvalBinary 就意味着首先要对在构造函数中传入的两个 xEval 类型的派生对象进行求值运算，然后他们分别求值运算的结果，作为 Unary 或者 Binary 的参数，在进行一次 Unary 或者 Binary 操作符的运算，得到的结果作为 xEvalBinary 求值运算的返回值。比如 xEvalBinary< xMult<xTensor4, xTensor2, xTensor2> > eval_stress(hooke, eval_strain);，这条语句的意思就是说，要进行一次求值运算，首先对构造函数传入的参数 hook，和 eval_strain 分别进行他们对应的 operator()(xGeoElement* appro, xGeoElement* integ, returntype& res) 进行求值运算，然后他们的返回的计算结果，分别是 xTensor4 和 xTensor2 类型的，他们分别作为二维谓词 xMult 的两个参数，再进行求值运算，得到的结果作为最终的返回值。因此可以看出整个这个 xEvalBinary 的返回值类型应该是 xMult<xTensor4, xTensor2, xTensor2> 中最后一个参数类型，而 xMult<xTensor4,xTensor2,xTensor2> 头两个参数的类型就分别是 hooke 和 eval_strain 的返回值类型。因此我们可以说 xEvalBinary, 或者 xEvalUnary 的意思就是首先进行 xEval 求值运算，然后返回值在进行 Binary 或者 Unary 操作符对应的运算。

8. 一些概念，对于每个节点都有 xValueKey xDoubleValue ShapeFuction, 通过 xfield 来管理，xvaluemanager。对于组装整体刚度矩阵，需要对每个单元进行积分，求解其单元刚度矩阵，而后装配到整体刚度矩阵。对每个进行积分求解单元刚度矩阵，就需要遍历每个单元，每个单元在遍历其中的节点来进行积分。

9. xMesh 中的 xVertex 对应的就是节点的几何对象，其中的 Id 就是 gmsh 文件中的 nodeid 因此在输出的 dcl.dbg 文件中也就是 xDoubleManager 中输出的调试信息的 Enti id 号。这样一旦出现问题时，可以对照 gmsh 和 dcl.dbg 文件来找到问题。

10. mEntity e_integ and e_appro; xGeoElement geo_integ and geo_appro; 区别就是当一个单元被加强时，zerolevel 与单元相交，这个时候整个单元叫做 e_appro and geo_appro，而那些小的 subelements 我们叫做 e_Integ and geo_integ，这个小的单元是用来获取积分点然后进行积分的。也就是说对整个大的单元的积分，转变成了对小的单元的积分的和。这样在每个小单元内部材料的特性就是一致的。一个大的单元有几个小的 sub 单元构成，但是记住一条规则小的单元的某些在大的单元中间节点上不存在 shapefunction，因为这些 sub-element 只是用来积分更加准确，实际上估计空间中并不存在这些自由度。这些小的子单元和大的单元享有共同形函数，但是他们有自己的积分点和 Jacbian 行列式。具体的积分点是通过 xGeoElement::getUVW 来获取的。如果有获取常规坐标下的积分点要用 getXYZ，遍历积分点可以通过 for(int k = 0; k < nb; k++)geo_integ->setUVW(k);

11. 如何 xspace 类型是 scalar 比如温度，则求得的温度场和场的 Gradient 操作分别是 double 和 xVector 类型; 如果 xspace 的类型是 Vector 比如位移场，则求得的位移场以及场的 Gradient 操作分别为 xVector 和 xTensor2。

12. xFormBiLinear 是用来组装左侧的刚度矩阵的，而 xFormLinear 是用来生成对应的载荷阵列的，也就是右侧的向量。如何将双线性形式和线性形式组装到对应的矩阵？通过 Assemble 这个函数就可以将对应的线性和双线形式组装到对应的矩阵和向量中。这个函数是重载函数，有多种版本形式.

13. Assemble 双线性形式的程序流程如下：首先要确定你的 test function 的估计函数和你的 shapefunction 是否一致，这里可以指定不同的函数空间，对于常规固体力学问题往往采用伽辽金法来建立有限元格式，因此 test function 的估计空间和求解的估计空间是一致的。在程序中通过参数 test and trial 来进行区分，如果选取一致那么就属于加辽金法。首先遍历你输入的积分区域（一般可是边，面，体），这样程序实际就是用来处理积分区域中的每个子区域。对于每个小的子区域，也就是一条边，一个面，一个体，程序中的变量就是 e_integ，另外就是获取估计函数，因为在 xfem 中需要对一个有两种材料的单元进行积分，而这个单元就是通过 levelset 来分割的，因此我们要对这种单元进行分割，然后分别对两侧积分，要注意这只是一种积分策略，实际上我们的估计空间中根本不存在这样两个子单元，而只有整个单元，因此对这样的子单元积分时就要用到包含这个子单元的大的单元的形函数来积分。这样这个积分对象的形函数就不是自身节点对应的形函数了。所以这就是为什么在程序中经常看到有 e_integ 和 e_appro，分别表示就是积分对象以及这个积分对象的估计函数。这样对于每个小的子区域，我们可以通过 xFiniteElement 这个对象获得相应的 Shape Functions，这样就获得对应的刚度矩阵了，比如对于一个二维三角形单元区域，有六个自由度，如果加上加强节点就有 12 个自由度，这样组合出来对应就是一个 12X12 的矩阵，矩阵每个元素都对应两个下标，这样你能标定把每个元素按照他们在整体刚度矩阵中的位置，组装上去了。要注意两点，第一即使是对于一个单元里面的 subdomain 进行积分，那么形函数个数取决于单元区域，而不是取决于 subdomain，我们只是从 subdomain 上获得准去的材料特性以及高斯

积分点，这样即使是对于这个比自己小的 subdomain，那么刚度矩阵仍然是 12X12 大小的。s

14. xFormBiLinear 是如何工作的呢？他的工作非常简单就是当你给我 subdomain 对象即 e_integ 以及对应的估计函数对象 e_appro，以及对应的形函数，他就会对每个高斯积分点处刚度矩阵就行累加，获得最终的 subdomain 对整体刚度矩阵的贡献。主要在 xFormBiLinear 中，在对每个高斯积分点积分之前，它都会把对应的高斯积分点的坐标放置在 xGeoElem(e_appro) 中，这样就方便我们调用了。但是对于每个高斯积分点如何积分呢，是采用双线性形式，但是双线性形式格式是什么样的呢？利用 C++ 语言中的虚拟函数这样我们就从 xFormBiLinear 中实现派生了大量的其它子类，比如 xFormBiLinearWithLaw, xFormBiLinearWithoutLaw 等。

15. xFormBiLInearWithoutlaw 中 accumulate_png 是首先返回在某个积分点处各个形函数对应的值（如何有 Operator，就应用那个 operator，一般是 gradient 操作符），分别对左项和右侧项执行这个操作，叫做 values_left, and values_right。获取该高斯积分点的权值和对应的雅阁比行列式的值 wdet，然后将获取的 values_left 12 和 values_right 12，分别相乘并乘以 wdet，这样最终就获取了，12X12 的刚度矩阵。

16. 关于如何从一个 Entity 访问与其相关的 Entity，比如从节点访问和节点相连的单元。mEntity* e; e->size(level); e 是一个基类指针，可以指向 vertex，edge，face，volume 等，level 表示想获取的对象的 size，比如 0 获取当前对象所包含的点的个数，1 表示想当前对象包含的线的个数，同理 2 表示面，3 表示体；如果当前对象是点，想获取的对象的类型是面，那么这是他就直接给你返回与这个点相连的面的个数。

2 常用代码

1. 填充一个新的 xField 对象

   下面的代码用来 fill 一个新的 field，有的时候经常需要

```
1  void fillField(xRegion& reg, xField &field, xEval<↩
       xTensor2> & eval_stress)
2  {
3      for(xIter it=reg.begin(0); it!=reg.end(0); ++it){
4          mEntity *e = *it; mVertex *v = (mVertex*) e;
5          int nelem=e->size(2); double value=0.;
6          for(int i=0;i<nelem;++i) {
7              mEntity {*}elem=e->get(2,i);
8              xGeomElem geo_appro(elem); xTensor2 stress;
9              eval_stress(geo_appro,geo_appro,stress);
10             value+=stress.vonMises();
11         }
12         field.setVal(e, val/nelem);
13         v->point()(0);
14     }
15 }
```

2. dfd