基于变分原理的有限元软件包构建

徐飞

2019 年 8 月 1 日

摘 要

本文介绍一种基于变分原理的有限元软件包构造方法. 有限元软件包的结构一般包括网格, 有限元空间, 形成刚度矩阵, 线性求解这几部分. 我们主要介绍一种形成刚度矩阵这个过程的方法, 该方法基于变分离散形式, 对于一个变分方程, 用户只需提供变分形式涉及到的导数类型及组合形式, 就可以方便的形成刚矩阵. 同时边界条件以及误差估计也可以统一的进行处理.

关键词. 有限元方法, 有限元软件包, 离散变分形式.

1 引言

随着科学技术的发展, 快速的交通工具、精密的航空航天设备、大规模的建筑物、大跨度的桥梁和大功率的发电机组等新事物不断取得进步. 这一切都要求工程师在设计阶段就能精确地预测出产品和工程的技术性能, 需要对结构的静、动力强度以及温度场、流场、电磁场和渗流等技术参数进行分析计算, 模拟出在实际环境中可能遇到的问题, 以便节省研发时间. 上世纪60年代在计算机技术和数值分析方法支持下发展起来的有限元方法 (FEM，Finite Element Method)则为解决这些复杂的工程分析计算问题提供了有效的途径. 有限元方法首先在结构力学领域提出, 最初应用于航空航天业, 随着近年计算机技术快速发展和普及, 这项技术已进入机械制造业、建筑业、汽车制造业、船舶制造工业、计算机及电子通信、材料工程等各行各业. 在工程界, 有限元方法已经成为应用最广泛的数值分析方法之一. 有限元模拟的意义具体表现在以下几个方面: 1．缩短产品设计和分析的循环周期; 2．增加产品和工程的可靠性; 3．采用优化设计，降低材料的消耗或成本; 4．在产品制造或工程施工前预先发现潜在的问题; 5．模拟各种试验方案，减少试验时间和经费.

有限元法的基本思想是将连续的求解区域离散为一组有限个、且按一定方式相互联结在一起的单元的组合体. 由于单元能按不同方式组合, 且单元本身又可以有不同的

形状, 因此可以模型化几何形状复杂的求解域. 有限元法作为数值分析方法的另一个重要特点是在每个单元上用分片多项式做基函数来近似精确解. 这样, 一个问题的有限元分析中, 未知场函数或及其导数在某个自由度上的数值就成为新的未知量, 从而使一个连续的无限维问题变成离散的有限维问题. 已经求解出这些未知量, 就可以通过插值函数计算出各个单元内场函数的近似值, 从而得到整个求解域上的近似解. 由于大多数实际问题难以得到准确解, 而有限元不仅计算精度高, 而且能适应各种复杂形状, 因而成为行之有效的工程分析手段.

有限元求解问题的基本步骤通常为: 1.问题及求解区域定义:根据实际问题近似确定求解区域. 2.有限元网络剖分: 将求解区域剖分成小单元, 二维情况常见的单元有三角形单元和四边形单元, 三维情况常见的单元有四面体单元和六面体单元. 单元越小 (网络越细) 则离散域的近似程度越好, 计算结果也越精确, 但计算量将增大，因此求解区域的网格剖分是有限元法的核心技术之一. 3.形成刚度矩阵: 在每个小单元上形成单刚矩阵, 利用局部编号和全局编号的映射关系合成总刚矩阵. 4.线性系统求解和结果解释: 有限元法最终导致一个稀疏的线性系统. 线性系统的求解可用直接法或选代法. 求解结果是近似解在自由度上的取值. 对于计算结果的质量, 将通过误差估计来评价并确定是否需要重复计算. 简言之, 有限元分析可分成三个阶段, 前处理, 处理和后处理. 前处理是建立有限元模型，完成单元网格剖分; 处理是形成线性系统以及求解; 后处理则是采集处理分析结果, 使用户能简便提取信息, 了解计算结果.

本文的组织结构如下, 第二部分将介绍一下有限元方法的基本知识以及有限元软件包的基本搭建步骤, 第三部分介绍基于离散变分格式的刚度矩阵合成过程, 第四部分介绍边界条件以及误差估计的处理, 最后一节展示几个数值计算结果.

2 变分问题的有限元方法

这一节我们简要的介绍一下有限元方法的基本知识以及有限元软件包的基本搭建流程.

文章下面的部分我们将以如下的变分方程为例具体说明我们的软件包构建方式: 求 $u \in H_0^1(\Omega)$ 使得

$$(\nabla u, \nabla v) + (\mathcal{A} \cdot \nabla u, v) \quad = \quad (f, v), \quad \forall v \in H_0^1(\Omega), \tag{2.1}$$

其中 $\mathcal{A} = (a_1(x), \cdots, a_d(x))$, d 是空间维数.

现在介绍一下有限元方法 [2, 3]. 首先构造一个网格: 对计算区域 Ω 进行一个规则的剖分, 得到一系列小单元构成的网格 $\mathcal{T}_h$, 单元 $K \in \mathcal{T}_h$的直径表示为 h_K. 网格直径 h 描述所有单元 $K \in \mathcal{T}_h$ 的直径的最大值. 有限元软件包中网格的产生可以先产生一个初

始网格, 然后进行网格加密. 初始网格的产生的方法有波前法, 加密方法有二分法加密, 一致加密, 自适应加密等. 也可以直接调用一些成熟的网格剖分软件, 目前常用的网格剖分软件有Netgen, Tetgen, Libmesh, Ani2D, Ani3D 等. 依靠产生的网格 $\mathcal{T}_h$, 我们构造线性取值有限元空间 $V_h \subset V$ 和测试有限元空间 $W_h \subset V$. 对于软件包结构中的有限元空间结构, 则要整合网格, 基函数类型, 自由度局部编号和全局编号的映射关系等信息. 那么有限元方法解变分方程就是求解如下方程: 求 $u_h \in V_h$, 使得

$$(\nabla u_h, \nabla v_h) + (\mathcal{A} \cdot \nabla u_h, v_h) \quad = \quad (f, v_h), \quad \forall v_h \in W_h. \tag{2.2}$$

记取值有限元空间基函数为 $\varphi_1, \varphi_2, \cdots, \varphi_N$, 测试有限元空间基函数为 $\psi_1, \psi_2, \cdots, \psi_N$, 其中 N 是空间 V_h 的维数. 那么 u_h 表示为

$$u_h = u_1\varphi_1 + u_2\varphi_2 + \cdots + u_N\varphi_N.$$

记 $U = \{u_1, u_2, \cdots, u_N\}$ 则方程可化为

$$AU = F. \tag{2.3}$$

其中 $A_{i,j} = (\nabla\varphi_i, \nabla\psi_j) + \sum_{k=1}^{d} (a_k\partial_k\varphi_i, \psi_j)$, $F_i = (f, \psi_i)$, $U_i = u_i$.

有了线性系统 (3.3), 我们就可以调用相应的解法器进行求解. 对于特征值变分问题, 最后形成一个广义特征值问题.

随着科学技术的发展, 快速的交通工具、精密的航空航天设备、大规模的建筑物、大跨度的桥梁和大功率的发电机组等新事物不断取得进步. 这一切都要求工程师在设计阶段就能精确地预测出产品和工程的技术性能, 需要对结构的静、动力强度以及温度场、流场、电磁场和渗流等技术参数进行分析计算, 模拟出在实际环境中可能遇到的问题, 以便节省研发时间. 上世纪60年代在计算机技术和数值分析方法支持下发展起来的有限元方法 (FEM，Finite Element Method)则为解决这些复杂的工程分析计算问题提供了一种有效的途径. 有限元方法首先在结构力学领域提出, 最初应用于航空航天业, 随着近年计算机技术快速发展和普及, 这项技术已进入机械制造业、建筑业、汽车制造业、船舶制造工业、计算机及电子通信、材料工程等各行各业. 在工程界, 有限元方法已经成为应用最广泛的数值分析方法之一. 有限元模拟的意义具体表现在以下几个方面: 1．缩短产品设计和分析的循环周期; 2．增加产品和工程的可靠性; 3．采用优化设计, 降低材料的消耗或成本; 4．在产品制造或工程施工前预先发现潜在的问题; 5．模拟各种试验方案, 减少试验时间和经费.

有限元方法的基本思想是将连续的求解区域离散为一组有限个、且按一定方式相互联结在一起的单元的组合体. 由于单元能按不同方式组合, 且单元本身又可以有不同

的形状, 因此可以模型化几何形状复杂的求解域. 有限元法作为数值分析方法的另一个重要特点是在每个单元上用分片多项式做基函数来近似精确解. 这样, 一个问题的有限元分析中, 未知函数及其导数在某个自由度上的数值就成为新的未知量, 从而使一个连续的无限维问题变成离散的有限维问题. 求解出这些未知量以后, 就可以通过插值函数计算出各个单元内函数的近似值, 从而得到整个求解域上的近似解. 由于大多数实际问题难以得到准确解, 而有限元不仅计算精度高, 而且能适应各种复杂形状, 因而成为行之有效的工程分析手段.

有限元求解问题的基本步骤为: 1. 问题及求解区域定义: 根据实际问题近似确定求解区域. 2. 有限元网格剖分: 将求解区域剖分成小单元, 二维区域常见的单元有三角形单元和四边形单元, 三维区域常见的单元有四面体单元和六面体单元. 单元越小 (网格越细)则离散域的近似程度越好, 计算结果也越精确, 但计算量将增大. 因此求解区域的网格剖分是有限元方法的核心技术之一. 3. 形成刚度矩阵: 在每个小单元上形成单刚矩阵, 然后利用自由度的局部编号和全局编号的映射关系合成总刚矩阵. 4. 线性系统求解和结果解释: 有限元法最终导致一个稀疏的线性系统. 线性系统的求解可用直接法或迭代法. 求解结果是近似解在自由度上的取值. 对于计算结果的质量, 将通过误差估计来评价并确定是否需要重复计算. 简言之, 有限元分析可分成三个阶段, 前处理、处理和后处理. 前处理是建立有限元模型，完成单元网格剖分; 处理是形成线性系统以及求解; 后处理则是采集处理分析结果, 使用户能简便提取信息, 了解计算结果.

本章介绍为了实现本文之前的所有算法而构建有限元软件包的方法. 组织结构如下, 第一节将介绍一下有限元方法的基本知识, 第二节介绍有限元软件包的主要结构, 第三节介绍基于离散变分形式的刚度矩阵合成过程, 第四节介绍边界条件以及误差估计的处理, 最后一节展示几个数值计算结果.

3 变分问题的有限元方法

这一节我们简要介绍一下有限元方法的基础知识.

本章我们将以如下的变分方程为例具体说明该软件包的搭建方式: 求 $u \in V$, 使得

$$(\nabla u, \nabla v) + (\mathbf{b} \cdot \nabla u, v) \quad = \quad (f, v), \quad \forall v \in V, \tag{3.1}$$

其中 $\mathbf{b} = (b_1(x), \cdots, b_d(x))^\top$, d 是空间维数.

现在介绍一下有限元方法[2, 3]的求解过程. 首先构造一个网格, 对计算区域 Ω进行一个规则的剖分, 得到由一系列小单元构成的网格 $\mathcal{T}_h$, 单元$K \in \mathcal{T}_h$的直径表示为h_K. 网格直径h表示所有单元$K \in \mathcal{T}_h$直径的最大值. 有限元软件包中网格的产生可以先给定一个初始剖分, 然后进行网格加密. 初始网格产生的方法有波前法、Delaunay方法等,

加密方法有二分加密法、一致加密法、自适应加密法等. 也可以直接调用一些成熟的网格剖分软件, 目前常用的网格剖分软件有Netgen、Tetgen、Libmesh等. 依靠产生的网格$\mathcal{T}_h$, 我们构造取值有限元空间$V_h \subset V$和测试有限元空间$W_h \subset V$. 对于软件包中的有限元空间结构, 则要整合网格、基函数类型、自由度局部编号和全局编号的映射关系等信息.

那么有限元方法解变分问题(3.1)就是求解如下方程: 求 $u_h \in V_h$, 使得

$$(\nabla u_h, \nabla v_h) + (\mathbf{b} \cdot \nabla u_h, v_h) \quad = \quad (f, v_h), \quad \forall v_h \in W_h. \tag{3.2}$$

记取值有限元空间V_h的基函数为 $\varphi_1, \varphi_2, \cdots, \varphi_N$, 测试有限元空间$W_h$的基函数为 $\psi_1, \psi_2, \cdots, \psi_N$, 其中$N$是有限元空间的维数. 那么 u_h可以表示为

$$u_h = u_1\varphi_1 + u_2\varphi_2 + \cdots + u_N\varphi_N.$$

记$U = \{u_1, u_2, \cdots, u_N\}$, 则方程可化为

$$AU = F, \tag{3.3}$$

其中 $A = (a_{i,j})_{i,j=1}^N$, $a_{i,j} = (\nabla\varphi_j, \nabla\psi_i) + \sum_{k=1}^{d}(b_k\partial_k\varphi_j, \psi_i)$, $F = (f_i)_{i=1}^N$, $f_i = (f, \psi_i)$, $U = (u_i)_{i=1}^N$.

有了线性系统 (3.3), 我们就可以调用相应的解法器进行求解. 对于特征值变分问题, 最后离散成为一个广义特征值方程.

4 有限元软件包的主要结构

有限元软件包求解变分问题时的主要步骤包括: 网格产生 $\Rightarrow$ 有限元空间构造 $\Rightarrow$ 单刚合成总刚 $\Rightarrow$ 线性系统求解 $\Rightarrow$ 误差估计.

下面对每一个步骤里用到的数据结构做一下详细解释.

4.1 网格结构

在软件包里, 网格的结构如下, 具体的含义可以参考后面的注释语句.

```
/*网格结构定义*/
typedef struct MESHES_ {
  int Num_Verts_Global; //网格点的个数
  int Num_Lines_Global; //网格边的个数
```

```
  int Num_Faces_Global; //网格面的个数
  int Num_Volus_Global; //网格体的个数

  VERT **Verts; //网格包含点的信息
  LINE **Lines; //网格包含边的信息
  FACE **Faces; //网格包含面的信息
  VOLU **Volus; //网格包含体的信息
} MESH;
```

网格的数据结构定义为MESH: 内部包含点(VERT), 边(EDGE), 面(FACE), 体(VOLU)的信息. 另外为了实现多重网格有限元方法, 还需要不同网格层之间的映射关系, 所以单元结构(VOLU)包含Parents和Ancestor两个变量,其中Parents表示当前网格单元在上一层网格中对应的单元编号, Ancestor表示当前网格单元在初始网格中对应的单元编号.

软件包中网格信息通过数据结构ELEMENT3D来进行调用, 这样当网格的结构改变时, 只需要调整一下ELEMENT3D的结构，而不必整体修改. ELEMENT3D的结构如下:

```
typedef struct ELEMENT3D_ {
  int ID_Num; //对应的单元的编号
  int Num_Verts;  //点的个数
  double *Vert_X; //点的坐标x
  double *Vert_Y; //点的坐标y
  double *Vert_Z; //点的坐标z
  double Volu;    //单元的体积
  double **Jacobian;       //单元的雅克比矩阵
  double **Inv_Jacobian;  //单元的雅克比逆矩阵
} ELEMENT3D;
```

有了网格就可以定义不同类型的基函数, 给出自由度分布情况.

4.2 有限元空间结构

```
/*有限元空间结构定义*/
typedef struct Fespace3D_{
  int DOF_ALL;    //自由度个数
  int Num_Volus;  //网格单元个数
```

```
  MESH *Mesh;       //网格信息
  BASEFUNCTION3D *Base; //基函数信息
  BoundCondFunct3D *BoundCond; //边界条件
  int *BeginIndex;     //自由度信息
  int *GlobalNumbers;//每个单元上的自由度
} Fespace3D;
```

有限元空间是对网格以及基函数信息的一个汇总, 包含网格以及基函数类型. 根据基函数类型对自由度进行编号并且给出单元上的局部自由度以及全局自由度之间的映射关系. 在计算过程中用到这些数据时, 从有限元空间里直接调用即可. 同时有限元空间包含判断边界条件的函数BoundCond, 该函数可以判断边界属于什么类型.

4.3 有限元线性系统结构

这一小节我们介绍有限元线性系统结构. 这个结构的功能是将形成刚度矩阵需要用到的部件封装起来. 其内部的结构包括数值积分信息、取值及测试有限元空间信息、有限元基函数涉及到的导数类型以及变分方程的组合形式等.

```
/*有限元线性系统结构定义*/
typedef struct DISCRETEFORM3D_ {
 Fespace3D *Ansatz_Space;     // 取值函数空间
 Fespace3D *Test_Space;       // 测试函数空间
 int N_Test_MultiIndex;       // 测试函数空间需要求导的个数
 MultiIndex3D *Test_MultiIndex;     //测试函数空间需要求导的类型
 int N_Anasatz_MultiIndex;          //取值函数空间需要求导的个数
 MultiIndex3D *Anasatz_MultiIndex;//取值函数空间需要求导的类型
 int N_AuxFeFun;                //有限元函数个数
 Fefunction3D **AuxFeFun;     //有限元函数
 int *N_AuxFeFun_MultiIndex;//有限元函数需要求导个数
 MultiIndex3D *AuxFeFun_MultiIndex;//有限元函数需要求导的类型
 int N_AuxFeFun_Values;       //有限元函数需要求值的个数
 double *AuxFeFun_Values;     //有限元函数需要求值的存储位置

 //下面的结构存储刚度矩阵的形成方式
 DiscreteFormMatrix *DiscreteFormVolu;
 DiscreteFormMatrixFace *DiscreteFormFace;
 DiscreteFormMatrixLine *DiscreteFormLine;
```

```
 DiscreteFormMatrixVert *DiscreteFormVert;
 //数值积分信息
 Quadrature3D *Quad3D;
 Quadrature2D *Quad2D;
 Quadrature1D *Quad1D;
} DISCRETEFORM3D;
```

有限元计算过程中需要调用DISCRETEFORM3D中存储的矩阵合成方式来形成刚度矩阵. 其中 DiscreteFormMatrix *DiscreteFormVolu用来实现在单元上的变分形式, DiscreteFormMatrixFace *DiscreteFormFace用来实现在面上的变分形式, DiscreteFormMatrixLine *DiscreteFormLine用来实现在边上的变分形式, DiscreteFormMatrixVert *DiscreteFormVert用来实现在点上的变分形式, 需要用户根据方程的具体形式自己给出. 本章下面的部分我们会结合具体的方程来说明如何给出这些结构的形式.

5 基于离散变分形式合成刚度矩阵

这一节我们介绍如何利用有限元线性系统来有效的实现有限元求解的第三步, 即形成刚度矩阵.

合成刚度矩阵的过程我们可以详细地描述如下: 遍历网格单元, 在每一个网格单元上根据方程的形式求出单刚矩阵, 然后按照网格单元上自由度的局部编号与全局编号的映射关系合成总刚矩阵. 类似地, 对于方程右端项也有相同的处理流程.

这里关注在每一个网格单元上如何有效地形成单刚矩阵. 我们的想法是在每一个单刚矩阵形成时, 需要提供变分方程中取值空间和测试空间涉及到的各阶导数值, 以及这些导数值的组合形式. 这两部分我们分别称为离散变分值和离散变分公式. 有了这两部分就可以得到每个单元上的单刚矩阵. 下面我们详细地介绍一下如何给出这两部分数据.

5.1 离散变分值

根据变分方程的具体形式, 给出取值空间和测试空间涉及的导数类型, 然后根据这些导数类型求出两个空间中的基函数在每个积分节点上的取值, 这样可以避免不必要的基函数取值.

以方程(3.2)为例, 对二维情况, 我们需要计算取值函数空间基函数的梯度值$\partial_x\varphi$和$\partial_y\varphi$, 测试函数空间基函数的梯度值 $\partial_x\psi$、$\partial_y\psi$和ψ. 对三维情况, 我们需要计算取值函数空间

基函数的梯度值 $\partial_x\varphi$、$\partial_y\varphi$和$\partial_z\varphi$, 测试函数空间基函数的梯度值 $\partial_x\psi$、$\partial_y\psi$、$\partial_z\psi$和ψ. 那么方程(3.2)就可以表示为

$$\begin{aligned}(\nabla\varphi,\nabla\psi)+(\mathbf{b}\cdot\nabla\varphi,\psi) &= \sum_{K\in\mathcal{T}_h}\Big\{(\nabla\varphi,\nabla\psi)|_K+(\mathbf{b}\cdot\nabla\varphi,\psi)|_K\Big\}\\ &\approx \sum_{K\in\mathcal{T}_h}\sum_{j=1}^{M}|K|\omega_j\Big\{\nabla\varphi(x_j)\nabla\psi(x_j)+(\mathbf{b}\cdot\nabla\varphi)(x_j)\psi(x_j)\Big\},\end{aligned}$$

其中M是单元上的积分节点个数, x_j是积分节点, w_j是积分权重.

这个过程需要对网格单元进行遍历, 在每个网格单元K上的所有积分节点上得到各个基函数的相应取值.

5.2 离散变分公式

当有了上一节得到的离散变分值以后, 需要给出相应的组合形式, 即离散变分公式, 以供软件包调用.

为了方便实现上述过程, 对各个基函数各阶导数值的存储顺序做一定的规定, 最外层按积分节点来存, 每一个积分点按基函数的局部标号来存, 每一个基函数再按离散变分值中需要的不同的导数类型来存储相应的数值, 最后利用组合形式得到想要的值.

以(3.2)为例说明离散变分公式的具体形式. 若将取值函数空间的基函数的计算值存在指针Ansatz所指向的内存中, 将测试函数空间的基函数的计算值存在指针Test所指向的内存中. 在每一个小单元K上, 对二维情况, 因为离散变分值要求的取值有限元空间导数类型为$\partial_x\varphi$和$\partial_y\varphi$, 测试有限元空间导数类型为$\partial_x\psi$、$\partial_y\psi$和ψ, 因此对某一个积分节点x_0, Ansatz[0]中存的是$\partial_x\varphi(x_0)$, Ansatz[1]中存的是 $\partial_y\varphi(x_0)$, Test[0]中存的是$\partial_x\psi(x_0)$, Test[1]中存的是$\partial_y\psi(x_0)$, Test[2]中存的是 $\psi(x_0)$. 则相应的离散变分公式可表示为Ansatz[0] $*$ Test[0] + Ansatz[1] $*$ Test[1] + $b_1(x_0)$ $*$ Ansatz[0] $*$ Test[2] + $b_2(x_0)$ $*$ Ansatz[1] $*$ Test[2]. 对三维情况, 因为离散变分值要求的取值有限元空间导数类型为$\partial_x\varphi$、$\partial_y\varphi$和$\partial_z\varphi$，测试有限元空间导数类型为$\partial_x\psi$、$\partial_y\psi$、$\partial_z\psi$和ψ. 因此, Ansatz[0]中存的是$\partial_x\varphi(x_0)$, Ansatz[1]中存的是$\partial_y\varphi(x_0)$, Ansatz[2]中存的是$\partial_z\varphi(x_0)$, Test[0]中存的是$\partial_x\psi(x_0)$, Test[1]中存的是$\partial_y\psi(x_0)$, Test[2]中存的是$\partial_z\psi(x_0)$, Test[3]中存的是$\psi(x_0)$. 则相应的离散变分公式可表示为Ansatz[0] $*$ Test[0] + Ansatz[1] $*$ Test[1] + Ansatz[2] $*$ Test[2]+$b_1(x_0)$$*$Ansatz[0]$*$Test[3]+$b_2(x_0)$$*$Ansatz[1]$*$Test[3]+$b_3(x_0)$$*$Ansatz[2]$*$Test[3].

5.3 单刚矩阵合成总刚矩阵

设每一个网格单元上自由度的个数为 n, 利用离散变分值和离散变分公式, 我们

可以得到网格单元上的一个$n \times n$的单刚矩阵, 然后需要把它加到总刚矩阵上去, 这需要单元上基函数局部编号对应的全局编号, 这个映射关系一般在形成有限元空间的时候已经生成. 存储在BeginIndex和GlobalNumbers这两个指针指向的内存中, 其中GlobalNumbers存储所有单元局部自由度编号对应的全局自由度编号, BeginIndex存储每一个单元在GlobalNumbers中的起始位置. 遍历所有网格单元就可以得到总刚矩阵.

6 边界条件处理和误差计算

有了离散变分公式这种处理方式, 边界条件和误差估计也可以统一处理. 处理非Dirichlet类型的边界条件其实就是计算边界上的积分, 可以看做是在低维空间的变分形式, 类似的也只需给出在边界上变分形式需要的导数类型以及组合形式, 对应的积分节点坐标需要映射到高维空间来使用, 相应的权重保持不变. 所以这种软件包编写方式, 可以统一处理各种类型的边界条件.

如果变分形式里出现$(u,v)|_{\partial\Omega}$这种边界积分项, 首先这也是一个积分形式, 只是积分区域在$d-1$维空间, 也可以得到相应的积分节点和权重, 然后权重不变, 将积分节点投影到d维空间. 在积分节点x_0, 相应的离散变分值为$\varphi(x_0)$, $\psi(x_0)$, 相应的离散变分公式为Ansatz[0] $*$ Test[0].

对于误差估计, 我们也可以将之写成变分形式进行统一处理. 对1半模估计 $|u-u_h|^2_{1,\Omega} = (\nabla(u-u_h), \nabla(u-u_h))$, 二维情况下相应的离散变分值为$\partial_x\varphi(x_0)$、$\partial_y\varphi(x_0)$、$\partial_x\psi(x_0)$、$\partial_y\psi(x_0)$, 相应的离散变分公式为 Ansatz[0]$*$Test[0]+Ansatz[1]$*$Test[1]. 三维情况下相应的离散变分值为$\partial_x\varphi(x_0)$、$\partial_y\varphi(x_0)$、$\partial_z\varphi(x_0)$、$\partial_x\psi(x_0)$、 $\partial_y\psi(x_0)$、$\partial_z\psi(x_0)$, 相应的离散变分公式为 Ansatz[0] $*$ Test[0] +Ansatz[1] $*$ Test[1] + Ansatz[2] $*$ Test[2].

7 数值实验

本小节, 我们利用依照本章前面的方式所构建的软件包计算几个数值算例, 以验证我们软件包的正确性和有效性.

7.1 自适应有限元方法

第一个例子我们求解一个三维凹区域的特征值问题:

$$\begin{cases} -\Delta u &= \lambda u, \quad \text{在}\Omega\text{内}, \\ u &= 0, \quad \text{在 } \partial\Omega\text{内}, \\ \|u\|_{0,\Omega} &= 1, \end{cases} \tag{7.1}$$

计算区域$\Omega = (-1,1)^3 \backslash [0,1)^3$. 因为计算区域有凹角, 所以特征函数常伴有奇性. 图1是自适应加密网格以及该网格沿XY平面的截面. 图2是求解方程第一个特征对的误差. 同时我们也求解了该方程的前5个特征值. 相应的结果包含在图3中.

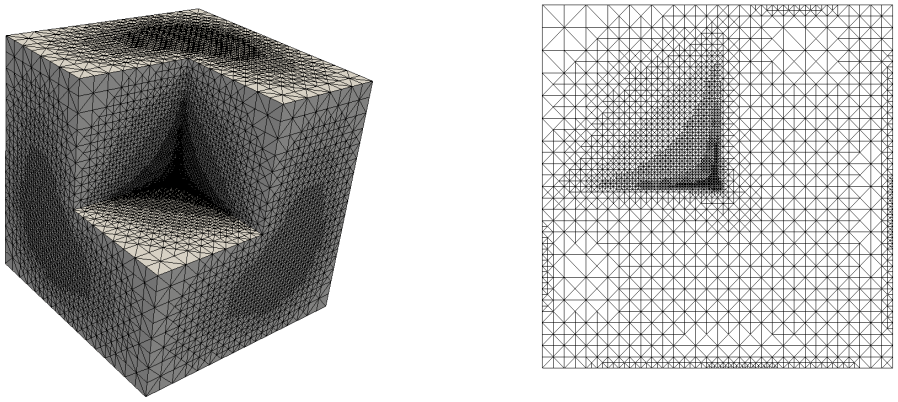

图 1: 例7.1的自适应加密网格以及截面

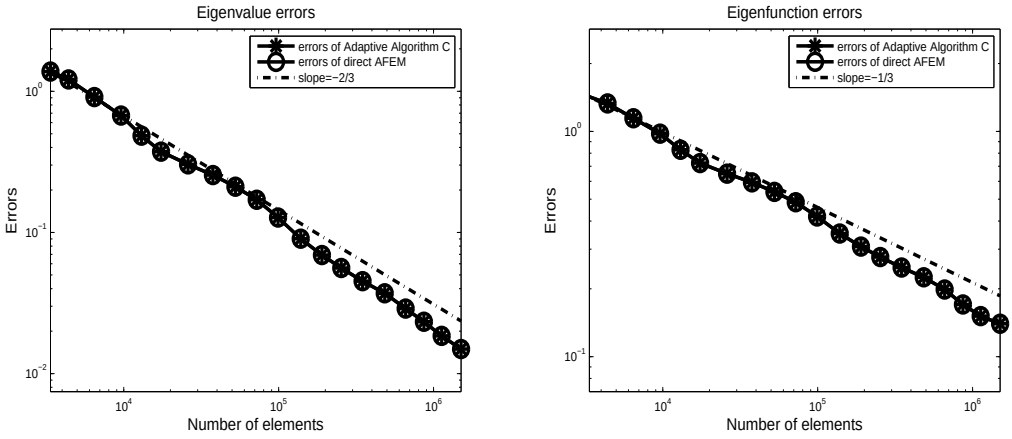

图 2: 例7.1的特征对误差

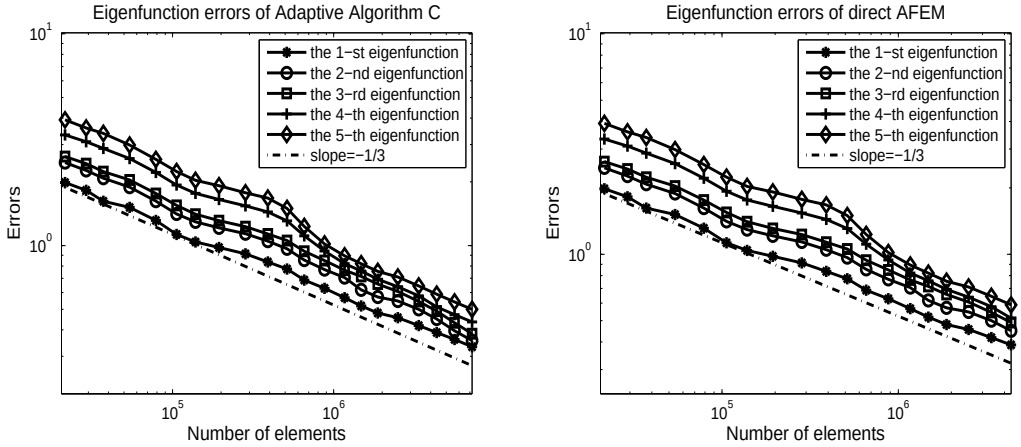

图 3: 例7.1的前4个特征函数的误差

7.2 多元波色-爱因斯坦凝聚

第二个例子求解描述多元波色-爱因斯坦凝聚(MBEC)的向量GPE[**?**, **?**]:

$$\begin{cases} -\triangle\phi_1 + V_1\phi_1 + \beta_{11}|\phi_1|^2\phi_1 + \beta_{12}|\phi_2|^2\phi_1 + \beta_{13}|\phi_3|^2\phi_1 = \lambda_1\phi_1, & \text{在}\Omega\text{内}, \\ -\triangle\phi_2 + V_2\phi_2 + \beta_{21}|\phi_1|^2\phi_2 + \beta_{22}|\phi_2|^2\phi_2 + \beta_{23}|\phi_3|^2\phi_2 = \lambda_2\phi_2, & \text{在}\Omega\text{内}, \\ -\triangle\phi_3 + V_3\phi_3 + \beta_{31}|\phi_1|^2\phi_3 + \beta_{32}|\phi_2|^2\phi_3 + \beta_{33}|\phi_3|^2\phi_3 = \lambda_3\phi_3, & \text{在}\Omega\text{内}, \\ \phi_1 = \phi_2 = \phi_3 = 0, & \text{在}\partial\Omega\text{上}, \\ \int_\Omega \phi_1^2 d\Omega = \int_\Omega \phi_2^2 d\Omega = \int_\Omega \phi_3^2 d\Omega = 1, & \end{cases} \tag{7.2}$$

计算区域$\Omega = (0,1)^3$, 势阱 $V_1 = (x_1^2 + x_2^2 + x_3^2)/2$、$V_2 = (x_1^2 + x_2^2 + x_3^2)/4$、$V_3 = (x_1^2 + x_2^2 + x_3^2)/5$, 不同内部态之间的相互作用系数 $\beta_{11} = 10$, $\beta_{12} = 2$, $\beta_{13} = 3$, $\beta_{21} = 2$, $\beta_{22} = 10$, $\beta_{23} = 4$, $\beta_{31} = 3$, $\beta_{32} = 4$, $\beta_{33} = 8$. 多元波色爱因斯坦凝聚是当系统的温度降低到某个特定值之下时, 有多个内部态可以同时存在于势阱中. 我们这里求解该方程的基态能量, 即方程的最小特征值.

我们这里利用多重校正算法来求解多元波色爱因斯坦凝聚模型, 图4和5给出了特征对的误差估计以及计算时间.

参考文献

[1] R. A. Adams, Sobolev spaces, Academic Press, New York, 1975.

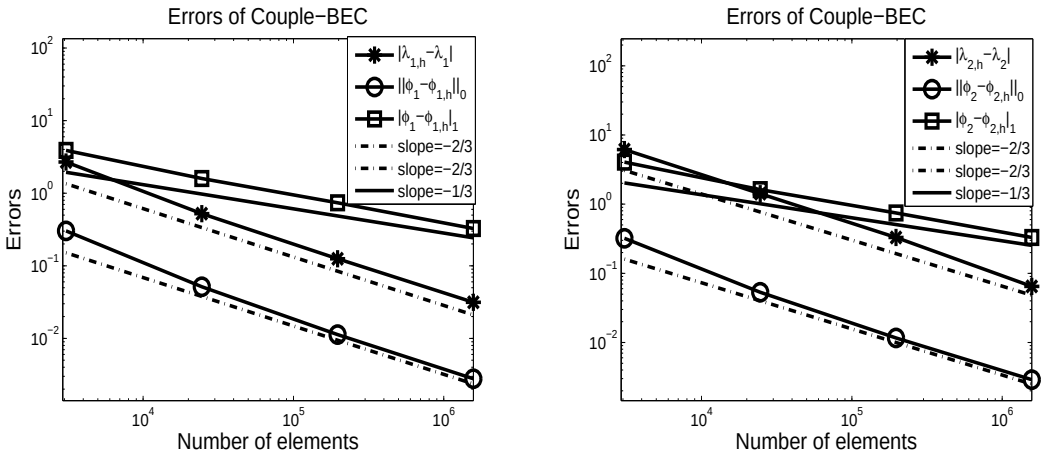

图 4: 例7.2的特征对误差

[2] S. Brenner and L. Scott, *The Mathematical Theory of Finite Element Methods*, New York: Springer-Verlag, 1994.

[3] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, Amsterdam: North-Holland, 1978.

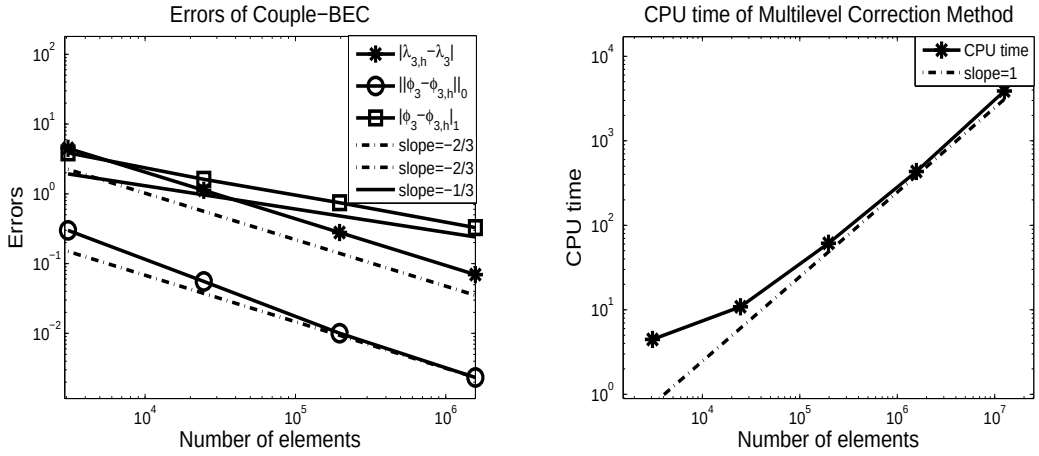

图 5: 例7.2的特征对误差及计算时间