**流形CFD开发文档**

**v 0.4**

软 件 声 明

本软件遵循GPL-3.0协议，详情请参考代码仓LICENSE声明。

本软件100%原创，100%国产，100%开源。

代码仓：https://gitee.com/DuanCP/lx-cfd

邮箱：[1368312974@qq.com](mailto:1368312974@qq.com)

电话：15610129897

微信：同电话号码

日 期： 2021.8.22

前 言

1. **关于本软件**

**任何的设计都是为了满足特定的设计需求。**由于是个人业余时间编写，因此首先要考虑的是提高开发效率，低代码量自然是第一要求，其次才考虑计算效率、并行方式等。目前代码量估计1万行左右，尽可能保持代码量在个人开发者的可控范围内。

**并行方案的选择：**多线程（个人开发，精力有限）。

**架构：**软件完全的按照面向对象的编程思路，划分出了互相独立的功能模块（作者是这么认为的）。软件的最底层是对偏微分方程的离散求解，在其上构建了CFD计算，因此本软件理论上可轻松实现多物理场耦合。本软件初始的C#版基于WPF平台构建，因此理论上C++版也可以很容易添加GUI。

1. **关于作者**

**毕业即转行：**动力工程及工程热物理这个专业就业率数据挺高，但我找工作的时候也没几个选择。去了一家制造空调的国企（血泪教训：参与市场竞争的国企不是“国企”）做供应商质量工程师，到处跟人扯皮甩锅，实在不能胜任（现在估计可以了）。后进入某大学教授开的小企业，盈利方式有：参与政府组织的比赛拿奖金，接“好友”的项目给回扣。这……？

**两年半后开始从事热设计：**后来不得不到大城市南京打个工，在一家制造业做热设计。来这边已经近两年了，怎么说呢，我们组十个人左右，我已经是组内排名第三的老员工了……。

**边自学边编程一年半：**做热设计后才开始了解CFD的理论，读了一本书后突然感觉这东西没想象中的难，新冠肺炎刚爆发的那段时间，大家都居家隔离，开始学习用C#编写WPF界面。2020年5月份才买到《The Finite Volume Method in Computational Fluid Dynamics》，算是正式开始算法部分。2021年4月份在知乎上发了个帖子，红皮书上的算法写到湍流了。6月份买了一本C++的书，开始更换语言改写。

**3）软件会不会收费**

软件不会收费，盈利方式也没有。不过我还是想提一下，如果软件盈利了，我会把钱拿来探索共产主义企业制度。现在我们提到的内卷，分明是经济危机的前兆。假如10亿人工作可以养活14亿人，结果会是人人安居乐业吗？不会，会有7亿人加班7亿人失业。

**目录**

[1 让代码跑起来 1](#_Toc20145)

[1.1 使用步骤 1](#_Toc15127)

[1.2 算例说明 1](#_Toc32216)

[1.2.1 算例0 瞬态固体导热 1](#_Toc12714)

[1.2.2 算例1 两区域耦合计算 2](#_Toc9770)

[1.2.3 算例2 自定义热源 3](#_Toc19511)

[1.2.4 算例3 方腔顶盖驱动流 3](#_Toc29975)

[1.2.5 算例4 自然对流 5](#_Toc30800)

[2 数据结构 1](#_Toc18501)

[2.1 软件数据结构 1](#_Toc4602)

[2.2 网格数据结构 1](#_Toc29903)

[2.3 矩阵数据结构 1](#_Toc1632)

[3 代码结构 3](#_Toc30130)

[3.1 计算流程 3](#_Toc1313)

[3.2 类 3](#_Toc8787)

[4 附录 5](#_Toc32044)

[4.1 用一个简单的例子学会代数多重网格（AMG）算法 5](#_Toc22149)

[4.1.1 引入一个例子 5](#_Toc18749)

[4.1.2 使用AMG算法 5](#_Toc32637)

[4.1.3 分析AMG算法过程 6](#_Toc20854)

[4.1.4 作者的说明 7](#_Toc23948)

# 让代码跑起来

## 使用步骤

1. 安装软件VSCode及插件，GCC等。具体可参考可参考视频https://www.bilibili.com/video/BV13K411M78v
2. 下载源码。LxCFD下载地址https://gitee.com/DuanCP/lx-cfd
3. 生成项目。
4. 运行，选择演示算例。
5. 计算完成后会在同目录下生成包含温度场信息的vtu文件，可使用paraview打开查看。

注1：目前软件只能计算1.2节中展示的算例，如有其他需求，需要自己改写代码。

注2：msh格式的文件需要放置在D:/LxCases目录下，或者在代码中更改目标文件夹。

注3：代码文件的编码格式是GB2312，如果出现乱码需要切换编码。

如仍有不明确之处，可直接联系我，或关注我的社交账号并留言。

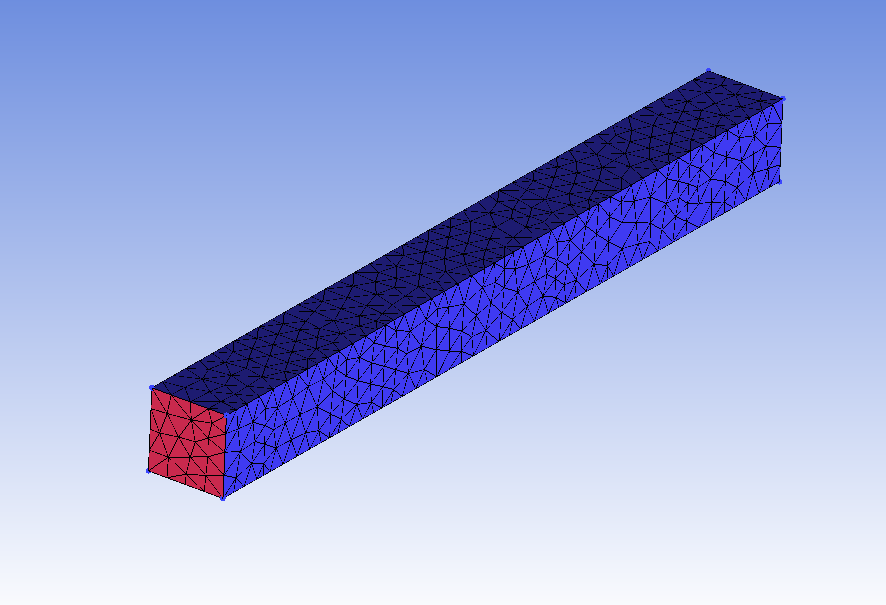
B站：<https://space.bilibili.com/447574913>

知乎：https://www.zhihu.com/people/duan-chong-peng

## 算例说明

### 算例0 瞬态固体导热

问题描述：如图，计算温度场随时间变化。



1m

1m

10m

固体：

导热系数200W/m·K

密度2700kg/m3

比热900J/kg·K

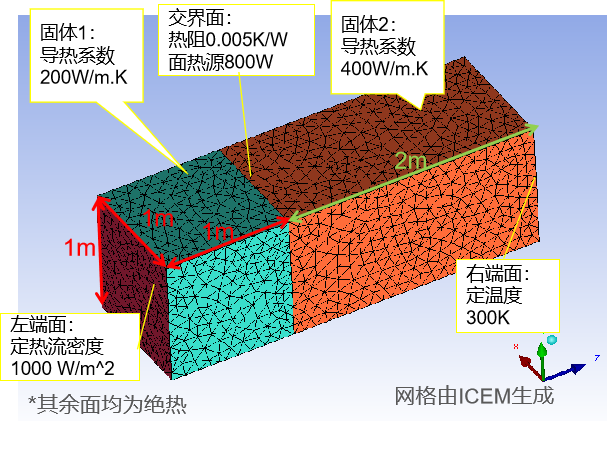
左端面：

定热流密度3000W/s

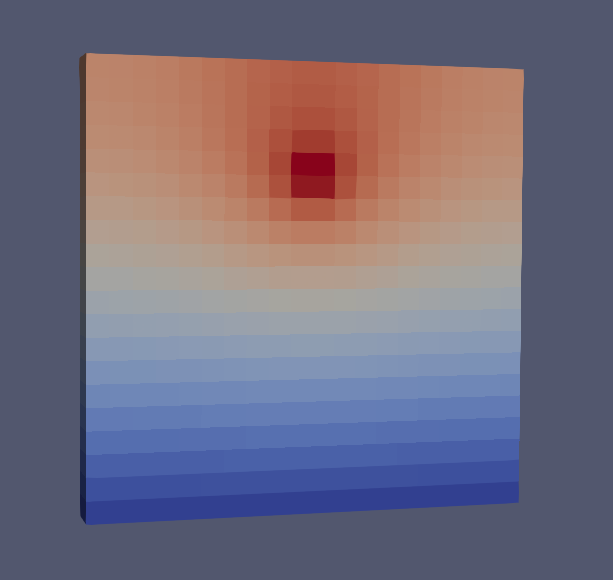
右端面：

定温度300K

### 算例1 两区域耦合计算



### 算例2 自定义热源

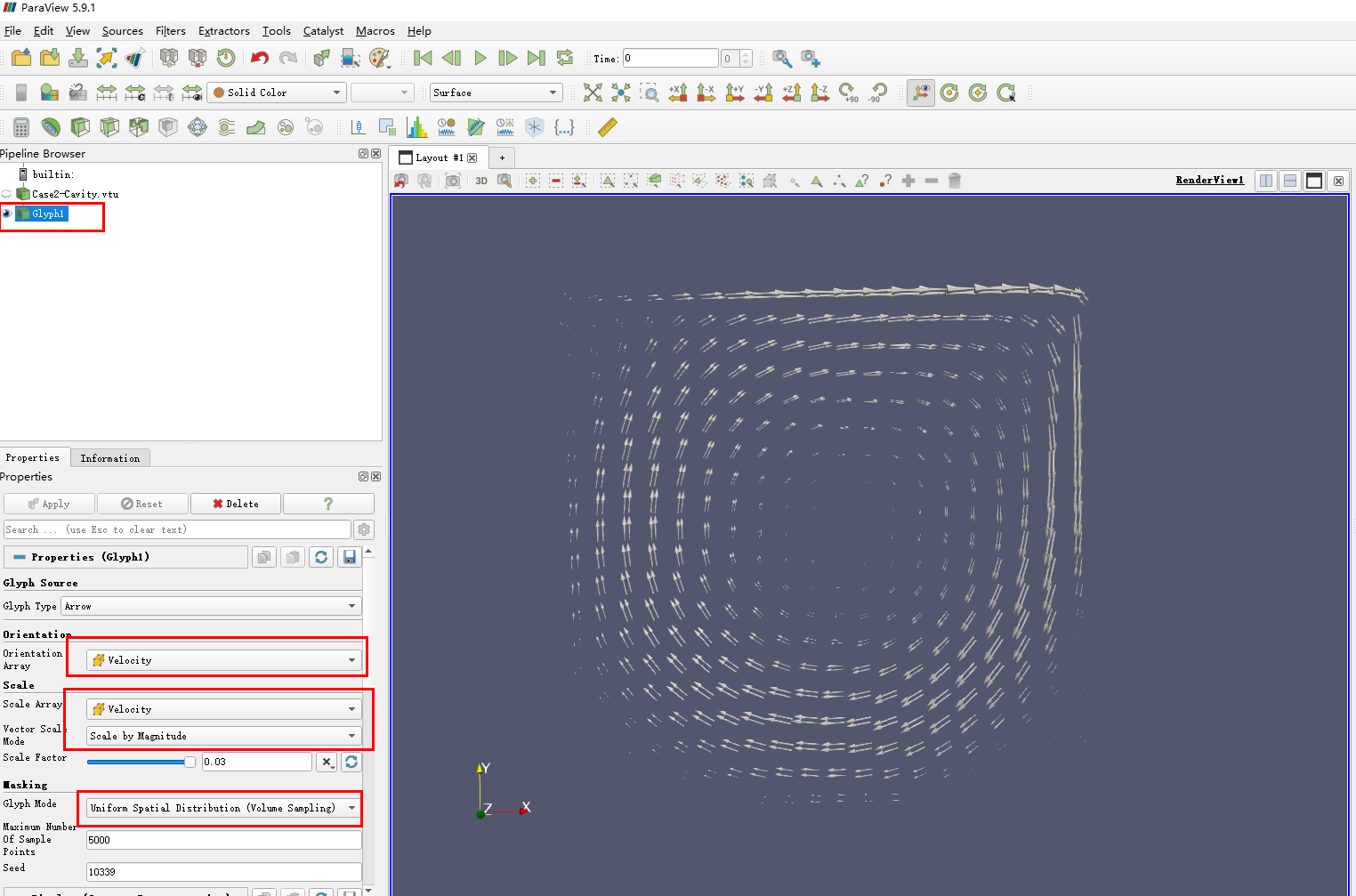


定温度边界

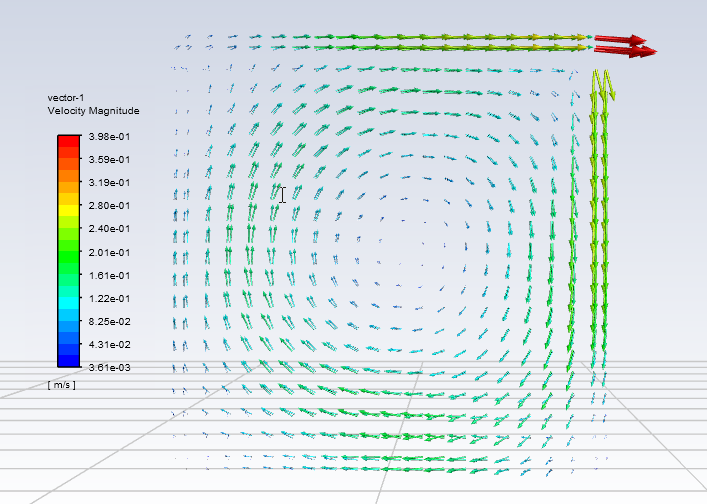
自定义范围热源

### 算例3 方腔顶盖驱动流

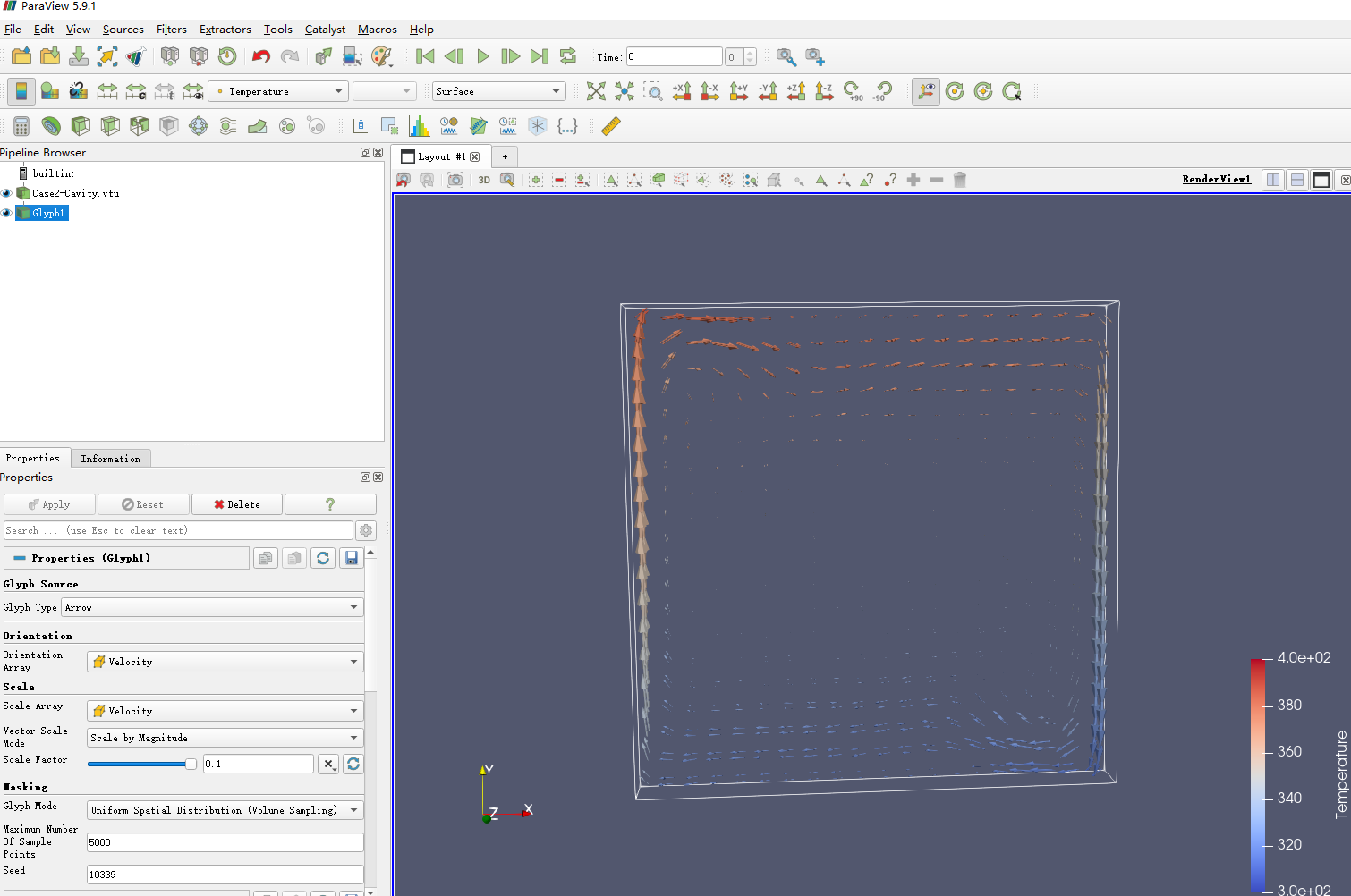
本算例是一个经典算例，查看结果时需要注意下图标红部分，由于本软件写出的vtu文件中，既包括Cell的信息，也包括边界Face的信息，边界处信息存在重叠。



另外，本软件对NS离散方法进行了一定的改进，下图为某商软的计算结果，右上角处网格的速度明显不合理。本软件修正了该错误，牺牲了部分收敛速度。



### 算例4 自然对流



400K

300K

# 数据结构

## 软件数据结构

见附件。

## 网格数据结构

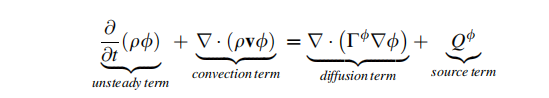
## 矩阵数据结构

# 代码结构

## 计算流程

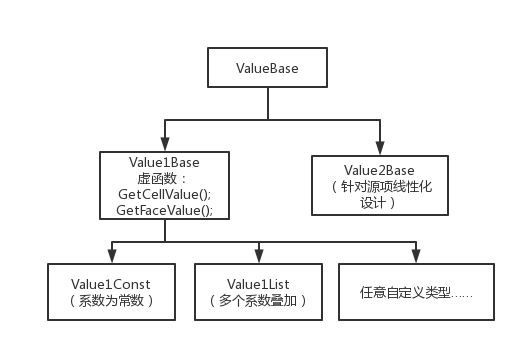
## 偏微分方程通用表达

下图为NS方程的通用表达形式，对于动量方程、能量方程、湍流方程，仅仅是红框内的系数发生变化，因此首先需要找到系数的抽象表达形式（代码中使用基类或接口实现）。

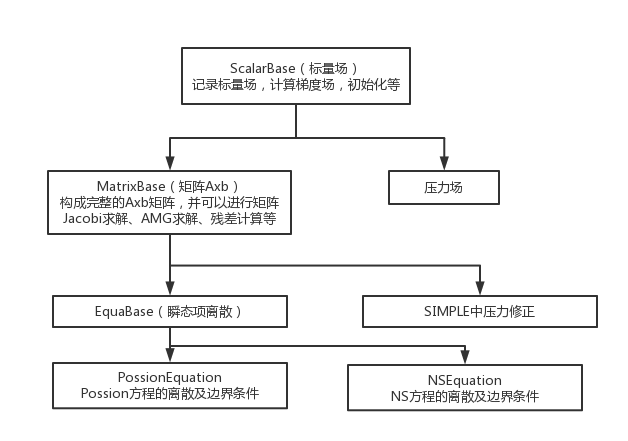


在离散过程中，非稳态项、源项的系数对应的是网格点的值，对流项、扩散项系数对应的是网格面的值。以源项为例，无论是常数，或者随位置发生变化，或者随某个物理量发生变化，在将偏微分方程转化为线性方程组时，都需要读取网格点对应的源项值，乘以网格体积后放到线性方程组等号右端。

因此在流形CFD这款软件中，偏微分方程系数的类设计及继承关系见下图：



有了系数，接下来就是为偏微分方程设计的类及其继承关系（与代码相比，有简化）：



最近看了下OpenFOAM的代码（仅限于浏览，不是很懂），大量使用的模板。流形CFD则大量使用了继承重写来实现复杂的逻辑关系。我还是觉得模板这个东西仅仅适用于处理数据类型的通用化设计，逻辑比较复杂的问题还是用继承重写来解决吧。

# 附录

## 用一个简单的例子学会代数多重网格（AMG）算法

介绍代数多重网格的资料不少，但多数是泛泛而谈，毕竟没有亲手实践的话，对这个问题的理解不可能很深。我根据自己的理解，对这一问题进行了总结，如有不严谨的地方，欢迎指出。

### 引入一个例子

考虑到用数学符号来描述过于抽象，因此特意构造了以下方程组（该方程组看上去简单，但收敛极慢，更进一步，可以认为该方程组包含了两类边界条件，处为第二类边界条件，为第一类边界条件）：

（1）

先给答案：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | x1 | x2 | x3 | x4 |
| 解析解 | 19 | 10 | 9 | 0 |

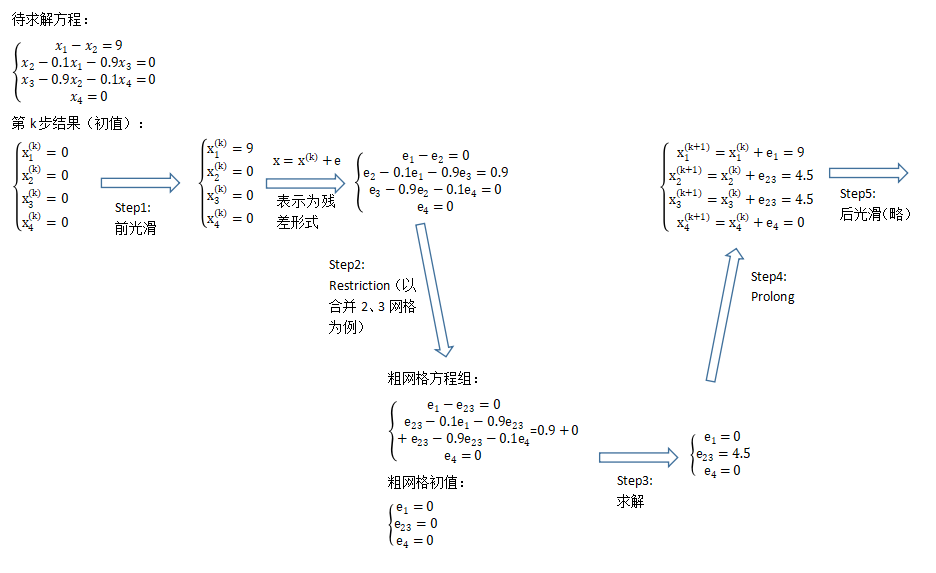
初始值均设置为0，我们用Jacobi迭代试一下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 迭代次数 | x1 | x2 | x3 | x4 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 9 | 0 | 0 | 0 |
| 2 | 9 | 0.9 | 0 | 0 |
| 3 | 9.9 | 0.9 | 0.81 | 0 |
| 4 | 9.9 | 1.719 | 0.81 | 0 |
| 5 | 10.719 | 1.719 | 1.5471 | 0 |

要注意，这个方程组只有3个未知量，这样的收敛速度显然不可以。我们发现，在计算时过分依赖，在计算时过分依赖，两者互相拖后腿，那么2、3位置只有一个网格会不会好一点？

### 使用AMG算法

先上用法：



粗网格计算完成后结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | x1 | x2 | x3 | x4 |
| 解析解 | 9 | 4.5 | 4.5 | 0 |

收敛速度是不是快了很多！

### 分析AMG算法过程

1）Step1：前光滑，图中仅使用了一次Jacobi迭代，实际使用时视情况而定。

2）残差形式：代入第k步迭代值，使Ax=b转化为Ae=r=b-Ax(k)的形式，一般资料都有介绍，不再赘述。

3）Step2：Restriction，接下来对该部分进行详细推导。

经过 上一步骤，细网格上的方程组已变为：

（2）

现在，我们准备将、合并，形成粗网格，合并后的数值为，即存在：

（3）

将式（3）代入式（2）：

（4）

将式（4）分成两部分，即式（4）=式（5）+式（6）：

（5）

（6）

将式（5）中第2、3个方程合并：

（7）

注意，以上步骤未引入任何假设。

接下来需要确定式（7）中r的值，以为例。由于式（5）（6）是由式（4）分解出来的，存在，我们不妨取，。粗网格节点数量少容易收敛，意味着在粗网格上可以很容易实现=0，即通过粗网格实现了=0，这样离收敛状态均为0更近了一步（剩余的误差是高频误差，通过细网格迭代可快速消除）。

以上过程称之为Restriction，一般书籍翻译为限制，我个人理解翻译为约束更容易理解，表示根据细网格上的残差对粗网格施加约束，使粗网格上的方程组具有唯一解。

4）Step3：求解粗网格对应的方程组。

粗网格上初始值均为0，本例中进行一次Jacobi迭代：

（8）

5）Step4：Prolong，即插值。

（9）

6）Step5：后光滑，根据需要选择合理方法。

### 作者的说明

其实，线性方程组的求解方法翻来覆去就这几个，在所有的求解方法里综合效果最佳的就是多重网格和共轭梯度。

商软中一般将这两种方法结合起来用：比如Star-CCM+采用AMG（默认V循环，可选）+共轭梯度法（具体哪种不清楚）；Fluent采用AMG（猜测F或W循环）+其他（作者没看出来）；Comsol采用BiCGSTAB+几何多重网格。

流形CFD采用了AMG和BiCGSTAB算法，和商软比起来差别不大。关于直接求解器和预处理，我觉得会有锦上添花的作用，但要考虑增加的计算量是否能带来足够的收益。

Step3:

求解

Step1:

前光滑

Step2:

Restriction（以合并2、3网格为例）

Step5:

后光滑（略）

表示为残差形式

Step4:

Prolong

粗网格方程组：

=

粗网格初值：

待求解方程：

第k步结果（初值）：