目 录

[2 相关理论与技术基础 1](#_Toc129643741)

[2.1 流体模拟相关理论 1](#_Toc129643742)

[2.1.1 N-S方程简述 1](#_Toc129643743)

[2.1.2流体模拟的拉格朗日法与欧拉法 2](#_Toc129643744)

[2.1.3 物质导数 2](#_Toc129643745)

[2.1.4 不可压缩性 3](#_Toc129643746)

[2.1.5 边界情况 3](#_Toc129643747)

[2.1.6 计算机求解N-S方程 4](#_Toc129643748)

[2.1.7 切割流体模拟等式 4](#_Toc129643749)

[2.1.8 时间间隔 5](#_Toc129643750)

[2.1.9 流体模拟网格 5](#_Toc129643751)

[2.1.10 解对流方程 6](#_Toc129643752)

[2.1.11 时间步长大小 7](#_Toc129643753)

[2.1.12 使液体不可压缩 7](#_Toc129643754)

[2.1.13 压力等式 9](#_Toc129643755)

[2.1.15 水平集方法 10](#_Toc129643756)

[2.1.16 速度外推 11](#_Toc129643757)

[2.1.17 粘度 11](#_Toc129643758)

[2.2 VTK相关理论 17](#_Toc129643759)

[2.2.1 VTK中的坐标系与转换 17](#_Toc129643760)

[2.2.2 着色器和图形管线 17](#_Toc129643761)

[2.2.3 VTK渲染管线 18](#_Toc129643762)

[2.2.4 VTK可视化管线组成 18](#_Toc129643763)

[2.2.5 VTK渲染引擎组成 18](#_Toc129643764)

[2.2.6 VTK管道的数据控制 19](#_Toc129643765)

[2.2.7 VTK的引用计数与智能指针 19](#_Toc129643766)

[2.2.8 VTK透明物体渲染 20](#_Toc129643767)

[2.2.9 VTK数据结构 20](#_Toc129643768)

[2.2.10 VTK的属性数据 20](#_Toc129643769)

[2.2.11 VTK的材质模型 21](#_Toc129643770)

[2.2.12 VTK交互器 22](#_Toc129643771)

[2.2.13 VTK交互优化 23](#_Toc129643772)

[2.3 相关开发工具介绍 24](#_Toc129643773)

[2.3.1 VTK 24](#_Toc129643774)

[3 总体设计 25](#_Toc129643775)

[3.1 流体模拟与显示模型 25](#_Toc129643776)

[3.2 功能模块划分 25](#_Toc129643777)

[4 具体实现 27](#_Toc129643778)

[4.1 流体模拟相关算法 27](#_Toc129643779)

[4.1.1 求解对流项 27](#_Toc129643780)

[4.1.2 求解压力与不可压缩项 28](#_Toc129643781)

[4.2 VTK可视化 29](#_Toc129643782)

[4.2.1 可视化管线的构建 29](#_Toc129643783)

[4.2.2 VTK渲染引擎构建 29](#_Toc129643784)

# 2 相关理论与技术基础

## 2.1 流体模拟相关理论

### 2.1.1 N-S方程简述

N-S方程叫做动量方程，他是牛顿经典的方程F=ma的推导。F=ma告诉我们当有力量施加在物体上物体的运动情况，同样N-S方程也是借助这个思想，通过力施加在流体上来计算流体的运动情况。方程为：

方程的各个量的意义如下：

表示流体的速度。

p表示压强，即流体施加在某个单位面积上的力大小。

g一般情况下我们叫他体积力，因为他是作用在整个流体上的力而不仅仅在表面。

V是粘度，用来表示流体的粘着程度。像是蜂蜜就有较高的粘稠度，而汞则较低，他用来表示流体有多大的力量来抵抗形变。

N-S方程推导：

当用粒子系统来模拟流体。每一个粒子都用于表示流体的一小团粒子。每个粒子将会有m（质量），v（体积），u（速度）等属性。为了预测粒子的运动，就需要分析都有那些力用于每个粒子：

f=ma告诉说明当力作用在粒子上，它是如何加速。

首先。将其带入到F=ma中得到

之后考虑有哪些力作用到粒子上：

1. 压力：粒子从高压区流向低压区。而求解所关注的是施加在粒子的合力。比如说，如果压力在各个方向都是相等的，这些合力将会是0，没有由于压力导致的加速度。只当不平衡时看到流体的效果，一侧粒子比另一侧压力大导致力从高压指向低压。表示粒子压力不平衡的最简单办法是去压强的负梯度−∇p，因为是从高压指向低压所有要加负号。如果是不可压缩流体则可以得到密度为一个常数，因此我们可以积分液体的体积来获得压力。
2. 体积力：流体整体中的每个粒子都受到的力，通常为重力
3. 粘力：流体中的粒子相互制约，导致每个粒子的速度都收到周围粒子的影响，从而影响流体整体的运动情况。现在直观的把它作为一个力，尝试使我们的粒子以一个与周围粒子平均的速度移动，粘力的作用是来最小化附近粒子速度的不同。微分算子中测量当前的测量值与他周围的数值的平均差距通常是用Laplacian ∇·∇.之后再用一个动态粘度系数η结合粒子的体积来表示他所受的粘度。而如果是变粘度流体，系数η将会变得更加复杂。

最后将所有的力放在一起就可以构成以下方程,之后我们需要将质量与体积趋向于0。可以得到流体粒子的受力和速度情况。然后，再除体积v来进一步化简等式。

通过除以密度得到

最后由于运动粘度等于ν = η/ρ所以的得到最后化简为

。

这其中：

为非定常项：当定常流动和静止的时候为0

为对流项：静止流场时为0，蠕变流时约为0

为单位质量流体的压力差

是单位质量流体的体积力

是扩散项（粘性力项）：是对静止或是理想的流体为0，高速非边界层约为0

### 2.1.2流体模拟的拉格朗日法与欧拉法

当想要追踪流体粒子的运动情况，有两种方法，一种是拉格朗日法，一种是欧拉法。

拉格朗日法，他对待流体像是对待粒子系统。在流体或固体的每一个点都像是独立的粒子，拥有独立的位置x和速度u。可以把他想像成每个粒子都可作为流体的一个分子。

欧拉法，一般用于液体。与追踪每个粒子不同的是，它关注空间中的固定点。他考虑流体中质点通过空间固定点流动行为，从而获得整个流场的运动规律。

### 2.1.3 物质导数

由于使用欧拉法来观察流体的运动情况，而又因为欧拉法是根据一个位置点来观察通过该空间点的不同流体质点的运动，而拉格朗日法是根据追踪一个质点来得到流场信息。流体力学中运用的是欧拉法，但这就带来了一个问题，牛顿第二定律是针对一个质点而言的，只有质点才有加速度，对于欧拉法下的物理量，由于其是以固定空间为基础的，所以求加速度时对空间固定点求加速度是没有意义的。所以我们要求的是物体质点的加速度在欧拉法下的答描述，物质导数顾名思义就是求导数是针对流体质点而不是空间固定点。所有物质导数为运动流体微团参数的时间变化率。

其中

为本地导数：它是固定点处的时间变化率。

为迁移导数：代表的是物理上由于流体微团从流场的一个点运动到另一个点，流场空间的不均匀性引起的时间变化率。

### 2.1.4 不可压缩性

真正的流体，就算是水，也可以改变体积。然而，液体一般不改变体积。尽管用很强力的方法来进行压缩也几乎是不改变的，空气则一定情况下会改变体积。因此流体可以分为可压缩流体与不可压缩流体。

在模拟的大部分过程中，我们接触的流体一般都是不可压缩流体。而在数学层面上，假设存在一点液体，可以通过对其在边界处的法向分量的速度积分来测量体积的改变速度: ,其中表示围出的区域。

对于不可压缩流体，他的体积是保持不变的，即：，可以根据散度定理将其化成定积分，因此可以得到。因为等于0是对任意闭区域都成立，所有可以得到。因此一个满足不可压缩条件的向量场被称为发散场。

所以，模拟不可压缩流体最关键的部分是确定速度场保持发散。

所以可以继续化简：原式

可以将第一项整理成然后化简成

### 2.1.5 边界情况

大部分，在流体模拟中主要的问题是得到正确的边界条件。对于边界情况可以分为两种：一种是固体边界，一种是自由表面。

对于固体表面：用速度来描述它是最简单的，流体不会流入或流出固体，因此表面的法向分量必须为零: ，而如果固体移动的话就会复杂一些。 这里流体在固体法向的速度要配合固体的法向分量，这个称谓不粘条件，此时只是限制速度的法向速度而没有限制在切线上的自由运动，而现实情况下，流体的切向速度可能与固体的切向速度没有关系。而关于固体表面的压强则有新的问题。关于压强可以认为是“抵抗液体的压缩”。在流体的N-S方程中，也是在边界上可以成立的，因此在边界上，压强可以用来控制。

以上就是无粘度流体在边界的情况，而事实是很多的流体都需要计算粘度的问题。如果有粘度的话计算就会变得更复杂了。在这种情况下，粘性可能就对流体速度的切向分量有影响，因此就可以分为固体运动和不运动的情况。

对于自由表面：流体不与固体接触，比如雨滴落下的时候周围全部都是空气。这情况下一般接触的是另一种流体，一般常见的就是空气，又由于空气的密度远远的小于水的密度，所以它对很多液体的影响不会那么大。我们通过简化，将它变成气压稳定的一个区域。事实上，因为只有压强不同才需要讨论，我们可已将大气压设为任意常数，0最常见。自由表面的其他情况是流体在一个更大的区域：比如，模拟烟在空气中，我们不能模拟整个大气。因此模拟我们关心的区域，越过模拟区域的边界，流体继续流动，但那些就不追踪了；我们允许他进入离开区域。这时候需要注意的情况是：越小的液体，表面张力越重要。在分子水平上，表面张力因为不同分子间的不同吸引强度引起的。比如水分子对水分子吸引力比对空气分子吸引力强：因此，水分子尽力想分开，空气想环绕着水分子。在几何的角度被建模成一种试图最小化表面积的力，或者等效地，试图减少表面的平均曲率，因此通常叫表面张力。简而言之，这个模型实际上是两种流体之间的压力跃变。可以表示为。其中[p]表示压力的变化, 水侧和空气侧测得的压力差，γ是表面张力κ是平均曲率[7]。

### 2.1.6 计算机求解N-S方程

直接求解N-S方程是个很难的事情，因此将使用分割的方法，将复杂的等式分割成很多个小部分，不断地解决每个部分从而最后求得结果。

方程切割正确性：

假设要想求解方程，这里的f（q）与g（q）全部都是独立的黑盒函数，有着输入数据，输出结果。

通过向前欧拉法计算过程为：

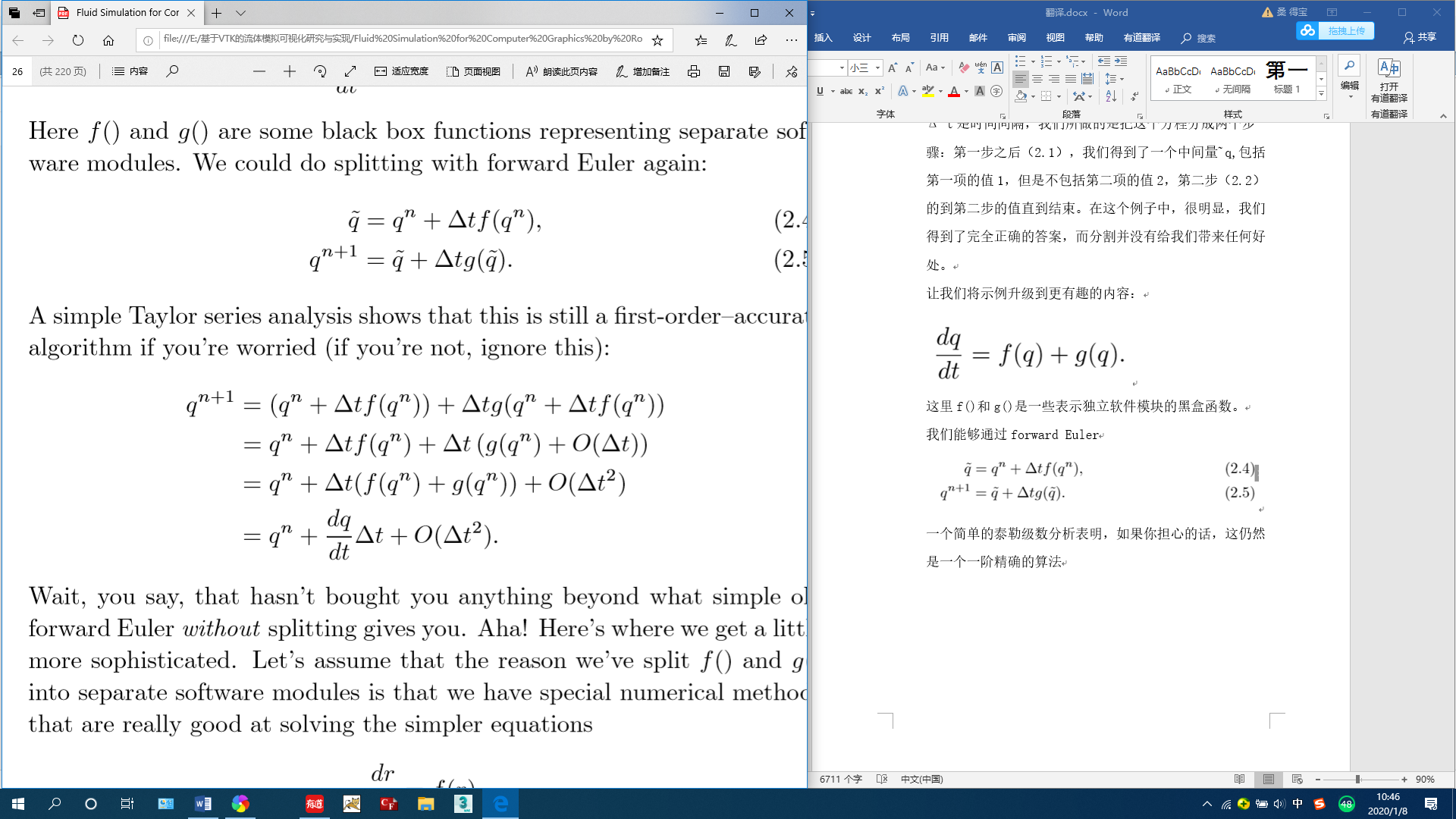
，对其推导如下图

图2.1向前欧拉法展开

可以得出这仍然是一个一阶精确的算法。

### 2.1.7 切割流体模拟等式

将原来的方程切割为：

对流项Dq/Dt=0函数为,表示量q在速度场上的速度场上运动，并且要保证速度场无散度。

体积力项使用向前欧拉法

压力和不可压缩项用project（，）来生成，并且保持速度的发散。

因此流体模拟的计算过程为：首先从定义一个初速度场开始。然后再决定一个时间间隔。之后就是计算，之后计算。计算project（，）

### 2.1.8 时间间隔

流体模拟首先要考虑到的是时间上的离散化，不可以选择超过当前动画帧的持续时间:当发现tn+Δt>tframe,，那样的话我们就需要Δt=tframe−tn，并且说明到达了帧的结束时间。但是Δt=tframe−tn也会存在问题。因为由于计算机浮点运算不精确，所以最终可能不会是你想要的帧结束时间，而且当一帧结束时我们可能会做一些其他操作，比如将动画的状态保存到磁盘，或者渲染到屏幕上。

### 2.1.9 流体模拟网格

由于如果使用欧拉法来进行求解，则需要网格来对流体流动到空间进行离散化。

通常是用MAC网格来进行空间离散化。Mac网格是交错网格，不同变量存储在不同的位置。比如二维的网格如图

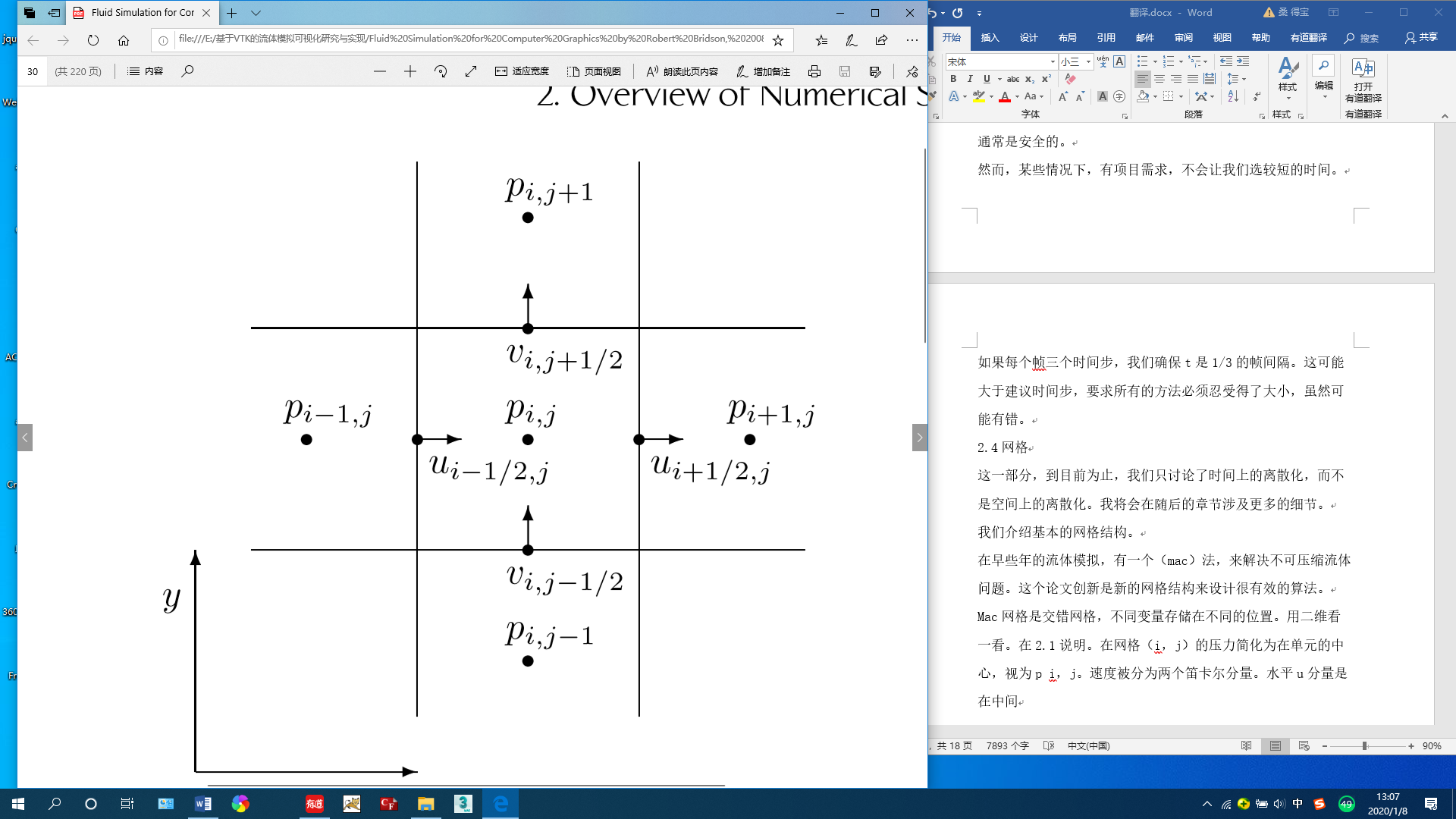


图2.2 二维MAC网格

其中网格单元（i，j）定义在单元的中心，计为Pij。而速度则是被分为两个笛卡尔分量，对边的中点进行采样。

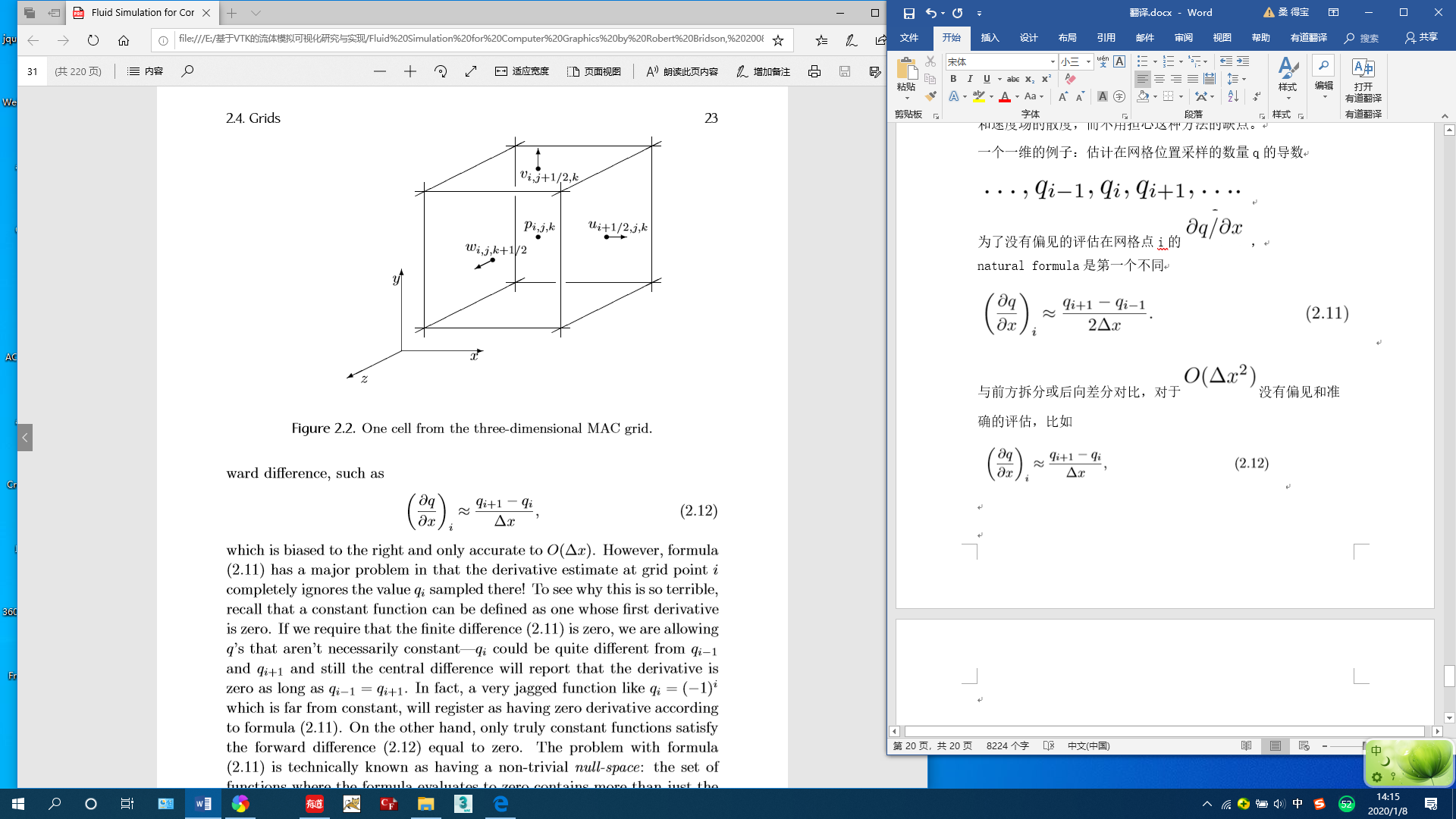
在三维空间，mac网格建立相似的方法，在网格单元中心的压强和三个不同的速度分量被分开，这样我们就有了在每个单元中心采样的速度的法向分量。而对于三维空间则如图2.2

图2.3 三维MAC网格

使用这种交错排列，就可以精确表示压力梯度和速度场的散度。

假设在一维条件下，无偏的估计i处的，当用的时候,虽然可以精确到O（）, 但是会出现完全忽略qi的情况，因此将会出现一些极端的错误情况，比如当出现锯齿状的函数时，即使qi的数值很高，依然会出现计算的结果为0的情况。因此在计算的时候则是采取通过的方程来计算，可以得到同样的无偏的，精确到O（），但不会出现上述情况的方法。

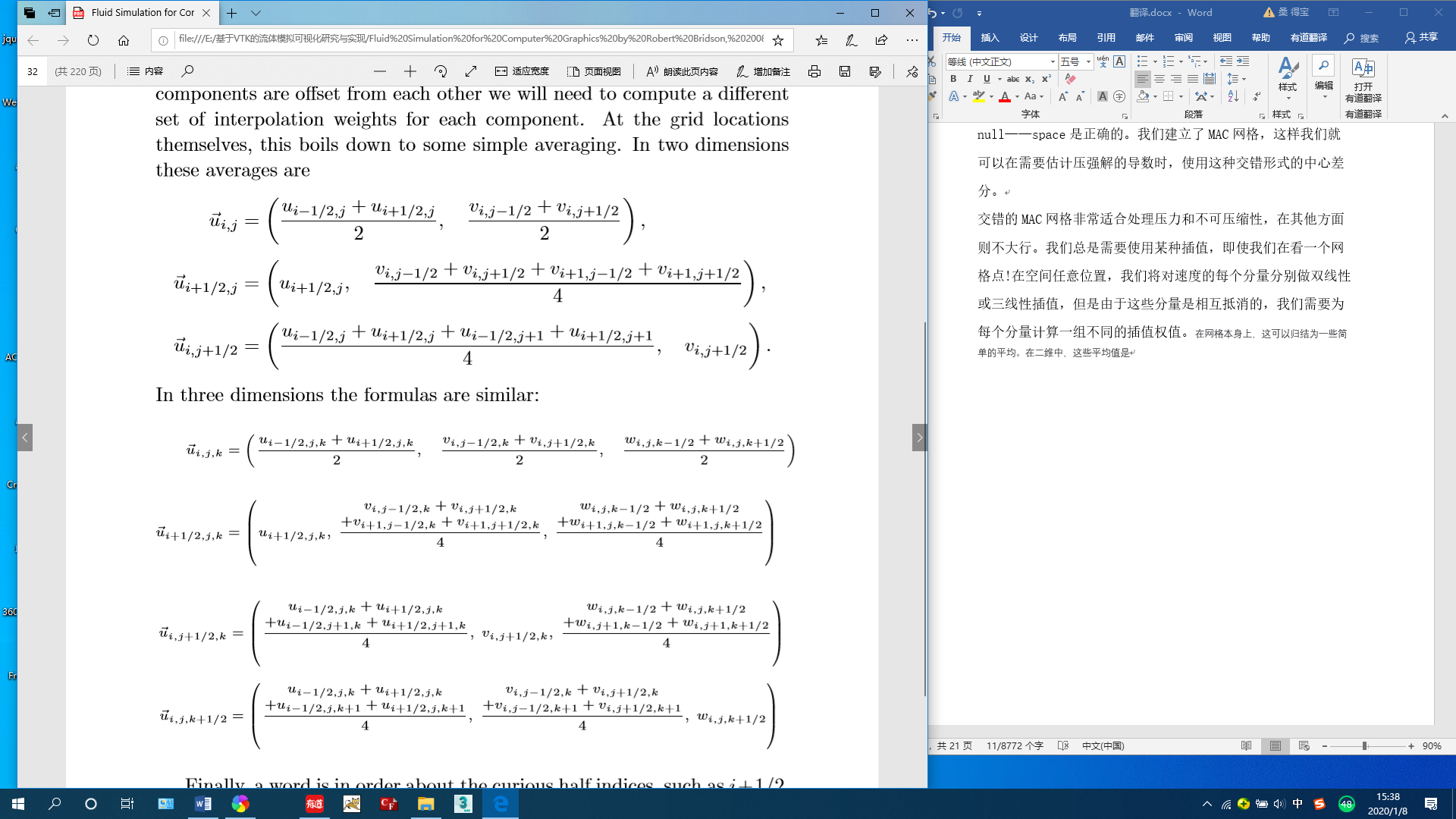
交错的MAC网格非常适合处理压力和不可压缩性，但是也会存在一些问题，比如要计算某一点的速度的时候，由于每个点只保存了一维数据 因此需要进行差值来实现,如图

图2.4 二维插值

当使用MAC网格的时候，就必须要定义：

P(i,j,k)=pi,j,k

U(i,j,k)=ui-1/2,j,k

V(i,j,k)=vi,j-1/2,k

W(i,j,k)=wi,j,k-1/2

MAC网格的数组大小为对于nx×ny×nz网格，压强被存在nx×ny×nz的数组位置上。U分量存在(nx+1)×ny×nz数组。v分量存在nx×(ny+1)×nz数组。w分量存在nx×ny×(nz+1)数组。

### 2.1.10 解对流方程

对流是指由于流体宏观运动，从而使流体内部各部分发生相对位移，人们研究对流扩散方程，主要的研究对象是流体在流动过程中，流体所携带的某种物质的物理量的变化规律，例如传热过程中温度的变化规律或者溶解于流体中溶质的物质浓度等物理量的变化规律。主要的求解方法是使用半拉格朗日的方法。

### 2.1.11 时间步长大小

当使用半拉格朗日法来计算，必须要考虑对于Δt不同大小的情况下，他是不是稳定的，数值会不会激增爆炸。但是根据原理可以知道无论粒子起点在那里,我们是找到旧的q值来更新新值q，即使是线性/双线性/三线性插值也总是产生位于我们插值的值之间的值，所以这是有界的。所以在步长上就可以比较自由的实际情况选择长度，如果希望运行效率则可以选择步长为每一帧的长度。但是如果设置太过于大的话则会产生奇怪的结果。所以有种策略是流体中的最快速度穿过5倍的Δx的时间，即：。

而如果使用则可以使粒子在每次计算的时候大约行进一个网格的距离，从而有效的降低错误。当然这种设置可以只在局部进行，比如一些流动速度比较快的区域。

### 2.1.12 使液体不可压缩

project（，）主要的工作为。他计算的结果为内部流体的速度满足不可压缩性，即，然后对于边界的情况必须要满足。

因此, 第一件事是将压力的更新给离散化,如何估计MAC网格的压力状况，之后, 我们将在MAC网格上定义离散散度，把它们联立在一起，得到一个线性方程组来求解压强。

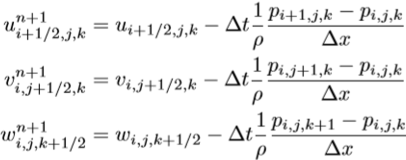
首先是要将压力梯度给离散化，如图2.4

图2.5 压力梯度离散化

其中，这些压力更新适用于包含流体的网格单元周围的每个速度分量。使用压力解决方案的最棘手的部分之一是跟踪哪些速度和压力样本是“活动的”。但是，在流体区域的边界面上，它们涉及到流体区域外的网格单元的压力。因此，我们需要确定压力边界条件。现在，假设网格单元可以分为流体单元(包含流体的单元)、固体单元(完全被固体占据的单元)或空气单元(不包含任何内容的单元——如自由地表水模拟)。而填充部分固体或液体的单元在之后考虑。然后假设固体与网格单元对齐，并且忽略流体单元中空气与水的分布情况，统一标为流体单元。

当常见的水模拟中，压力边界条件常出现在自由表面，可以假设流体外部压强为0，这种叫做狄利克雷法。更复杂的则是在固体边界。由于我们假设固体边界与网格单元面平行，则可以得出在网格单元存储的速度是，在这种情况下，当要减去压力梯度的时候要加上条件通过代入压强的更新，这就变成了一个简单的线性方程来求解固体壁面内的压强.

比如当知道方程：，又知道，的值是。我们将更新。

当我们实际更新代码中的速度时,我们可以简单地将固体边界的速度设置为固体的正常速度，上述方程在推导内部压力方程时起到作用。而对于边界条件来说，通常独立计算每个边界面上的固体网格单元的压力。如果一个固体网格单元有两个或两个以上的面与流体单元相邻，我们将计算单元的两个或两个以上完全独立的压力值。在连续介质条件下，流体的压力在固体内部没有定义，我们只有固体边界上的压力梯度。因为重要的只是边界的压力差。

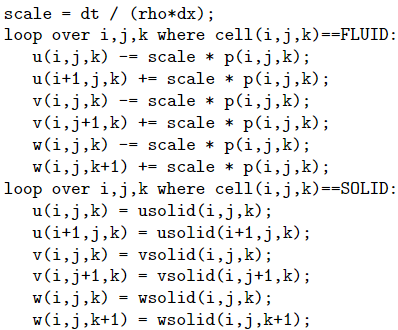
通过压力离散可以得出图2.5的伪代码

图2.6压力散度求解

在网格中，我们要求流体是不可压缩的，这就需要将流体的散度进行离散化，二维散度离散化后的结果为图2.6

图2.7 二维散度离散化

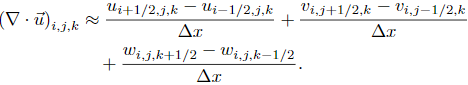
三维散度离散化的结果为图2.7

图2.8 三维散度离散化

而散度为0也只是对于流体的网格进行计算。而mac网格的作用也得到了体现，当用传统的将所有变量全部存储在网格中心的情况下，散度表示是下图

图2.9 中心存储的散度离散化

将会导致在一些高度发散的数据场内比如（（-1）i，（-1）j），计算结果产生很大误差。

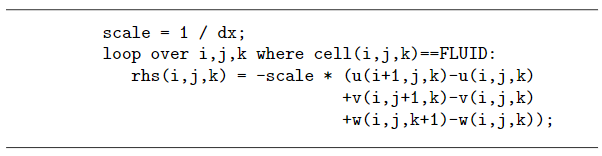
而负散度的计算代码为下图

图2.10 求解负散度

### 2.1.13 压力等式

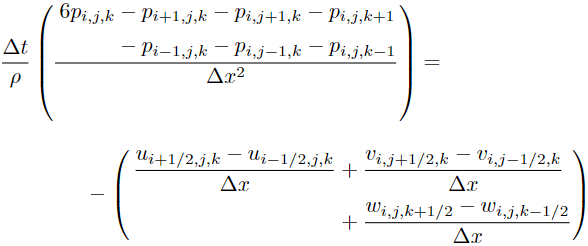
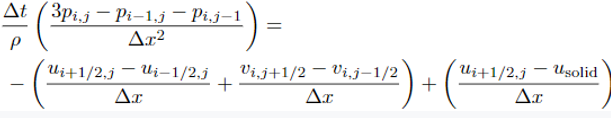
计算的最终结果是要求计算的速度是不发散的。又将之前的速度计算公式化简可以得到：

图2.11 速度化简结果

可以近似为。

如果一个流体网格单元在边界上,在边界面上的新速度涉及到流体之外的压力，比如网格（i，j+1）是气体单元，则需要将等式的此处的压力给替换掉，替换为数值0.但是如果网格（i+1，j）是一个固体单元，你那我们就需要替换为我们计算的固体边界的压强的值。比如在二维的空间中，我们假设（i-1，j）和（i，j-1）是流体单元，这样可以使得方程简化为：

图2.12 简化方程结果

因此可以总结出一般规律：

对于空气边界的情况只需要从方程中删除掉。而pij前面的系数则是等于非固体网格单元的领域数目，上述的二维空间的规律，经检验在三维同样也适用。因此我们需要对于rhs进行一个额外的循环来修正，伪代码为：

scale - 1 / dx;

loop over i,j,k where cell(i,j,k)==FLUID:

if cell(i-l,j,k)==SOLID:

rhs(i,j,k) -=scale \* (u(i,j,k) - usolid(i,j,k));

if cell(i+i,j(k)==SOLID:

rhs(i,j,k) +=scale \* (u(i+l,j,k) - usolid(i+l,j,k));

if cell(i,j-l,k)==SOLID:

rhs(i,j，k) -= scale \* (v(i,j,k) – vsolid(i,j,k));

if cell(i,j+l,k)==SOLID:

rhs(i，j，k) +- scale \* (v(i，j+l，k) – vsolid(i,j+1,k));

if cell(i,j,k-l)==SOLID:

rhs(i,j，k) -= scale \* (w(i,j,k) - wsolid(i,j,k));

if cell(i,j,k+l)—SOLID:

rhs(i,j,k) += scale \* (w(i,j,k+l) – wsolid(i,j,k+l));

### 2.1.15 水平集方法

来描述物体时不需要围绕粒子构建一个隐式函数，然后将其采样到网格中进行渲染，我们可以省去粒子并直接使用网格。这是水平集方法的核心思想。水平集方法相对于标记粒子的主要优势是消除了斑点:水平集可以很容易的产生光滑的曲面。

在MAC网格的中心定义的隐式曲面函数φ(i ,j, k)。三线或双线性插值可以在网格之间用来估计φ(x)，而函数的表面则是定义为φ（x）为0的位置，一般情况下，定义那些φ(x)<0是内部，φ（x）>0的地方则被认为是外部。而符号距离则被定义为图2.13

图2.13符号距离函数

对于如何计算符号距离，Sethian等提出利用快速步进法计算符号距离，分为两个步骤，其时间复杂度为O(NlogN)。李俊等提出基于原点映射的改进扫描算法快速构造符号距离函数，时间复杂度为O(n)[5]。

（1）二维初始化。发封闭曲线c上所有点距离为0而且标记访问，再将其他点距离无限大，且未访问。

（2）分别以四个扫描方向扫描网格。每次扫描数中，进行如下更新。

假设从（x+，y+）方向进行，则对每个网格点（i，j）比较，（i-1，j）与（j，i-1）的最小值min，并且将min加1，更新（i，j）的值为min与本身的最小值。然后（x-，y-），（x+，y-），（x-，y+）都要进行一遍。符号表可以通过一次搜索得来，从而判断在外部还是内部。当用圆形或球形的时候则可以简单的根据点到圆心的距离与半径进行比较二得出。

### 2.1.16 速度外推

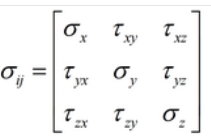
之前的计算暗示着只有拥有流体的网格的速度才将会更新。其他速度不受压力影响，因此可能不可靠。因为advection方法可能依靠那些未触及的值进行插值速度。对于无粘度流体是不能简单的使用固体的速度：只有固体速度的法向分量与流体有关系。因此，通过已经计算好的流体数据在网格其他部分来外推速度值。

外推最直接的方法是直接使用最近点的值，当我们构造好了符号距离，我们实际上在每个网格点都存储了最近邻元素的索引。所以从这些点里的到数值比较容易。当外推速度的时候，是在交错网格上进行工作的，所以可以从相邻的网格单元中心求平均值，找到存储在相邻网格单元中心的两个值中更近的一个。

### 2.1.17 粘度

自然界中，存在实际的流体都有粘性，但是虽然只是多了一项粘度力，但是给分析带来很多困难。

应力介绍：物体由于外因(受力、湿度、温度场变化等)而变形时，在物体内各部分之间产生相互作用的内力，以抵抗这种外因的作用，并试图使物体从变形后的位置恢复到变形前的位置。在所考察的截面某一点单位面积上的内力称为应力。同截面垂直的称为正应力或法向应力，同截面相切的称为剪应力或切应力。

应力张量：设p为变形体的一点。在附近取一个平面六边形。此六面体上三个互相垂直的三个平面上的应力分量即可表示该点的应力状态 。应力分量的第一个下标为作用在平面的法线，第二个是应力方向。如下图2.14，图2.15。

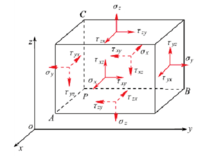
图2.14 应力

图2.15 应力在坐标系表示

所以x面的合应力为

Y面的合应力为

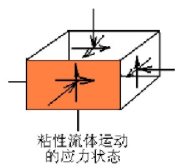
Z面的合应力为

切应力互等定律：切应力互等定理表达了一对切应力之间的关系，在相互垂直平面上，切应力成对存在且数值相等，两者都垂直于两个平面的交线，方向则共同指向或共同背离这一交线[8]。

由于切应力互等定理可以得到，矩阵为对角矩阵。则有六个应力分量相互独立。

运动的流体中，取出微分六面体，同一点任意三个正交面法向应力值和不变。

在粘性流体流动的时候，流体质点之间可以存在相对运动，流体具有抵抗变形能力，因此作用流体内部面上力有正向力，也有切向力

图2.16 应力状态

粘性流体应力状态：因为存在切向力使得任一点单位面积上表面力不一定垂直于面。

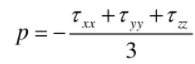
在理想流体中，因为不存在切应力，只有法向应力，他们的值也为压强负值。可以表示为：，其中δ来表示单位张量。而在粘性流体时，任一点的三个互不垂直面法向应力之和为不变值，并且值为该点的负压力值。但是切应力一般不为0，但是在静止时他是只有正应力而无切向应力。

图2.17 法向应力之和

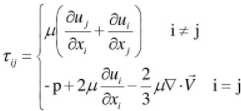
根据广义牛顿内摩擦定理可以得出：

图2.18 牛顿内摩擦定理

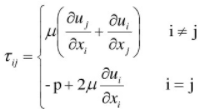
又因为当探讨不可压缩流体的时候有，则代入继续化简为：

图2.19 不可压缩流体牛顿内摩擦定理

则可以得出切应力和法向应力的值为图

图2.20 应力值

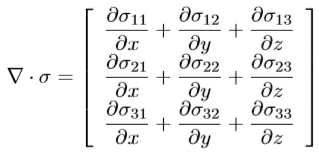
作用应力：要想求得液体微团的压力可用曲面积分再通过微分定理转成定积分其中为下图

图2.21

因为F=ma，则可以将上述两个式子划等号再加上体积力，又由于对于任意体积上式都成立，可以得出下图：

图2.22 连续体一般动量方程

对与粘度流体，压力是非常重要的，于是可以单独提取出来得到，其中是个对称张量，被用来使用压力来处理不可压缩约束和模拟其他行为。

基于粘度流体的两类边界状况：

一种是自由表面，一种是固体边界。在自由表面的情况下，也就是旁边是真空或是密度小的多的流体，就可以假设影响忽略不计。因此，自由表面并没有牵引力，所有边界条件为：。

当出现固体边界的情况下：在无滑动边界条件下可以得到 ，所以在静止边界可以得出u为0。

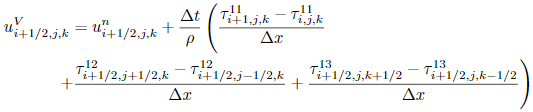
粘度部分要做的是利用粘性边界条件，通过输入的速度场更新获得新的速度场。由公式离散化会变成下图结果

图2.23离散化结果

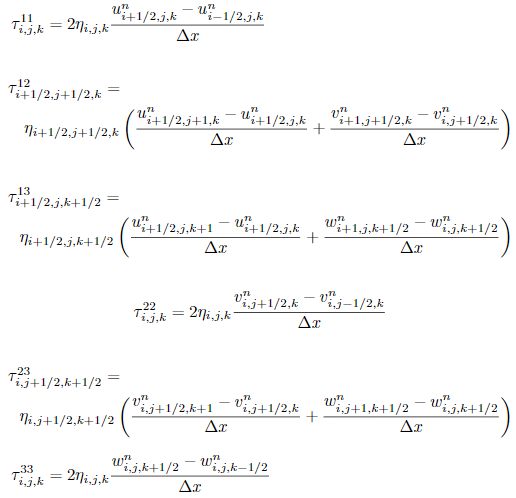
又因为τ为粘滞力张量,他的定义为下图：

图2.24 粘滞力张量定义

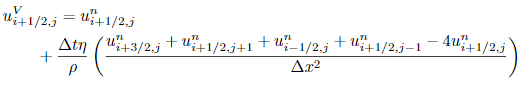
在η为恒定的情况下可以将τ直接代入化简。在二维情况下的u速度分量为下图

图2.25 二维速度分量

可以用求解压力的时候类似的方法来进行求解。但是在处理粘度流体与固体接触的时候，可以直接用固体速度，但是对于自由表面则要更细致一些，首先做的应该是将速度场外推空气中。但是上述的做法会出现一些问题。因此则需要使用别的方法来进行处理。一种叫做向后欧拉法的计算方法同样也可以用来处理。他大体意思是通过新的速度来评估应力张量，而不是通过旧的速度来评估。向后欧拉法的计算可以在一维的时候表示为下图

图2.26 向后欧拉法

然后将它进一步转换将q^(n+1)移动到左边可以得到下图。

图2.27 进一步转化结果

然后就可以将它看做是一个稀疏矩阵乘未知向量等于已知向量的形式。

即得到形式，其中I为单位矩阵，A是对角线上的元素为2，而对角线上下一单元的值则为-1，其他的部分的值则为0，而等式的右边则是我们已经知道的速度值。

对于多维的计算过程，总体是差不多的，最后的结果也差不多，A的对角线上的元素值为6，而对角线上下一单元的值则为-1，其他的部分的值则为0。这样就可以像压力一样求解，将两边的参数带入到PCG求解器中求解，其中左边的矩阵为稀疏矩阵，右边的为一个多维向量，并且不论是对多少维的流体进行模拟，一定要将多维的数据转换成一维的，然后再进行计算。

## 2.2 VTK相关理论

### 2.2.1 VTK中的坐标系与转换

VTK中的坐标系分为模型坐标系，世界坐标系，View坐标系，Display坐标系。

模型坐标系：定义模型的时候用到的坐标系，每个模型都需要一个自己的坐标系，通常是局部笛卡尔坐标系。

世界坐标系：当需要在渲染引擎中渲染的时候则需要将Actor放置在世界坐标系下，这是一个公共的坐标系比如渲染时的灯光，模型都是共同放在此坐标系下，actor的一个主要的功能也是将模型坐标系转换为世界坐标系。

View坐标系：这个是相机所接受到的坐标系xyz分别表示，平面上的位置，z表示物体与相机的距离。相机的主要功能就是将世界坐标转换为View坐标。

Display坐标系：各坐标轴的取值是像素值，是在具体屏幕上显示的坐标。

VTK常见的坐标变换为平移，缩放，旋转。

平移

缩放

旋转包括了x，y，z三个方向，绕x旋转，绕y旋转，绕z旋转。他的证明都是比较类似的，首先假设一个点要绕着z坐标轴旋转，则z的坐标不会改变，但是xy坐标都要变换，因此可以转换成二维的结果，此时可以将x和y的值换一个写方法，x为rcost，y为rsint。当旋转一个 角度之后可以得到x为rcos（t+u），y为rsin（t+u）。再通过三角函数展开，然后写成矩阵的形式就可以，推出上边的绕x旋转的矩阵，同理，绕y和z的结果也是差不多的。

### 2.2.2 着色器和图形管线

Vtk的渲染引擎，主要的作用就是将一系列的三维或二维的数据全部映射到屏幕的2D窗口上去。图形管道分为两个部分：3D与2D坐标的变换，2D的坐标转换为实际的像素颜色。

图形管线可以分为多个部分，每个部分都需要前面数据,有着较大的 并行性。着色器则是在每个计算步骤上运行在GPU上的程序，可以快速处理图形管线中的数据 。

管道部分解析：

顶点着色器：将单个顶点作为输入，在这里可以进行一些坐标的变换和顶点属性的处理，允许着色器修改。

形态装配:输入顶点着色器的顶点，形成图元并且组合形状的点，该过程不允许修改。

几何着色器：将基本图元的顶点作为输入，并且能够在这个过程里形成新的图元，允许着色器修改。

光栅化：将图元映射到相应的屏幕像素，而且会裁剪掉视图之外的片段，不允许修改。

片段着色器：包含着像素值的许多信息，用于计算像素的最终颜色。

深度测试与混合阶段：将已经绘制的图形与在帧缓存中的图形做混合，因此片段作色器的颜色并不一定是最终颜色。

### 2.2.3 VTK渲染管线

包括VTK可视化管线和VTK渲染引擎两个部分。VTK可视化管线主要是创建或者是获取数据，然后再将数据进行处理，转换成渲染引擎能够解析的数据，然后传输给渲染引擎。渲染引擎主要负责将传输过来的数据进行可视化表达。

### 2.2.4 VTK可视化管线组成

Source：作为VTK的数据源，用于创建对象或者是读取对象他的输出对象可以用于下一级来处理。

Filter：主要作用是用来加工数据，可以有多个之间相互连接，他的输入和输出的个数大于等于1。

Mapper：连接VTK的可视化管线和渲染管线的是Mapper，他的主要作用是将VTK的可视化管线的数据进行转换，转换成几何图元（点，线，多边形），使得渲染引擎可以识别，他作为是连接两者之间的一个组件。

### 2.2.5 VTK渲染引擎组成

VtkProp：是任何可以存在于渲染窗口中的对象的父类。包括了二维和三维的对象，而vtkProp3D则是他的派生，主要是渲染三维对象。他主要依赖于两个对象，Mapper对象，负责存放数据和渲染的信息，vtkProperty对象存储外观（属性）参数，如颜色、不透明度、材质的环境光（Ambient）系数、散射光（Diffuse）系数和反射光（Specular）系数等，这样设计主要的优势 就在于可以方便将外观的参数多个共享。

VtkCamera：存储场景中的摄像机的数据。

主要参数是摄像机的位置、焦点、和场景中的上方向向量。

相机位置：设置的坐标是世界坐标。

相机焦点：默认为世界坐标的原点位置。

朝上方向：他可以确定相机的朝上方向。

投影方向：相机位置到相机焦点的 向量方向。

投影方法：有两种，正交投影和透视投影，正交投影进入相机的光线与投影的方向是平行的。

透视投影则类似于人的眼睛可以产生近大远小的效果，而且是有视角的。

视角。

裁剪平面：只有当物体距离摄像机的位置位于前后裁剪平面之间，他才会显示数据。

VtkLight：可以分为位置灯光（Positional Light，也叫聚光灯）和方向灯光（Direction Light）[6]。

聚光灯：光源位于渲染场景的某个位置，可设置颜色，灯光位置，灯光焦点，灯光强度，灯光锥角。

平行光：光源位置来自无限远，光线可认为是平行光可设置位置，焦点。

VtkRenderer：负责管理场景中的渲染过程。

VtkRenderWindow：将操作系统与VTK的渲染引擎连接起来，具有平台无关性，所以当我们想要渲染的时候可以调用它再由VTK根据相应的平台来更换相应的子类。

VtkRenderWindowInteractor：负责监听鼠标、键盘和时钟消息，并通过VTK中的Command/Observer设计模式进行相应的处理，并且可以选择不同的渲染交互风格对场景实时查看。并且使用start（）方法后将进入事件的响应循环，使得交互器等待交互事件。

### 2.2.6 VTK管道的数据控制

VTK采用一种惰性赋值的方法。他的执行过程是直到需要数据（或被强制）时，可视化管道才会产生数据，产生的数据是最新的。对数据的需求由Update()方法提出，这一方法可以显示调用，也可以由图形流水线中演员 Actor的Render()方法自动调用。对数据的需求会逆着流水线的方向传递，如果某一部分的数据过时（通过检查时间标签），可视化流水线中的过滤器会重新执行，最新生成的数据沿着流水线传递给Actor。

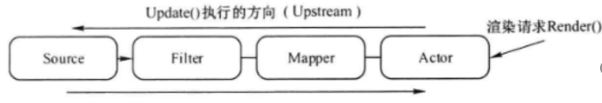
因为有惰性赋值，所以将会出现，数据执行更新请求方向和数据流向相反的现象

图2.13 可视化管线数据流方向

### 2.2.7 VTK的引用计数与智能指针

引用计数为当一个对象被引用的时候，该对象的引用计数加一，当一个对象取消该对象的引用。而且只要还其他对象引用该对象，那对象就一直存在，否则程序则进行销毁。但是当出现循环引用的时候则会出现问题。VTK中的vtkGarbageCollector用来解决这个问题。vtkObjectBase用于实现引用计数，意味着绝大多数的VTK类都支持引用计数。VTK采用智能指针管理内存，它会自动管理引用计数器的增加和减少。若检测某对象的数值减少为0，则会释放该对象的资源，用于达到自动管理内存的目的。

### 2.2.8 VTK透明物体渲染

VTK对透明物体渲染效果最好的方法为“深度剥离”算法，首先常规渲染深度离相机最近的一层，然后把这个深度设置给下次渲染采样。下次渲染的时候 遇到比这个depth更接近或一样近的像素就会丢弃。这样就渲染出 第二层接近相机的深度和颜色，那么再渲染一次 又会得到第三层 以此类推。最后再按照一定比例进行混合。

### 2.2.9 VTK数据结构

VtkDataSet是由VtkDataObject派生的，主要用于储存VTK中的各种数据。他的组织结构为拓扑结构和几何结构和属性数据：

几何结构：点数据所定义的一系列坐标点构成，描述对象的空间位置关系。

拓扑结构：点数据连接形成的单元数据由单元数据形成数据集的拓扑结构，描述对象的构成形式，它具有几何不变性。

属性数据：是对几何结构和拓扑结构的补充。

数据集由一个或多个的单元构成。其中单元是一系列有序的点按指定类型连接所定义的结构。分为线性单元与非线性单元，类型的区分主要是通过插值函数为依据：

线性单元：主要采用线性或者是常量插值函数。

非线性单元：为了更精确 的表达数据，采用非线性单元作为数据的基本表达结构。非线性单元可以提供更准确插值函数，可以获得更加理想的可视化效果。

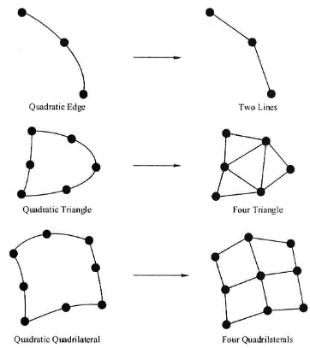
所有的这些单元都将要转换成图元才能被渲染引擎处理，线性单元可以方便的转化成线性图元。而非线性单元不能直接处理，只能先转化成线形图一来处理，才能被图形库支持。然而VTK的 主要方法是将非线性单元细分为线性单元，再把细分结果作为线性单元处理。转化过程部分如图2.14

图2.14 VTK用该固定细分的方法将非线性单元细分为线性单元

### 2.2.10 VTK的属性数据

属性数据与数据集的组织结构相关联的信息。可以将他分为：标量数据，矢量数据，张量数据，纹理坐标。

标量数据：数据集里的每个位置具有的单值数据，表示数据大小。

矢量数据：可以被认为是既有大小又有方向的量，如速度，应力，位移等。

纹理坐标：通过纹理映射，将纹理坐标从笛卡尔坐标空间映射到多维纹理空间。

张量数据：一个量是几维张量是由变量的坐标维数所决定的。

### 2.2.11 VTK的材质模型

（1）VTK的Phong光照模型:

Phong光照模型主要分成3个分量组成：环境光，漫反射，镜面反射。

环境光：现实世界的光可以向很多的反方向发散，对每一个事物都有着或多或少的影响，但是全局照明所需到的计算力和资源要求都非常的大，因此才需要抽象和简化成一个全局照明的模型，而环境光就是模拟全局光照下照明的结果

漫反射：光线投射到粗糙物体上再向各个方向反射的结果，强度与光线方向和法线方向的夹角有关，与相机的视角无关，比如光线直射物体时最亮。

高光：和观察点有关，光线照射向物体，沿着物体的法线方向反射，当摄像机接近反射的线时，高光就会越强。

（2）基于物理的渲染：

VTK的基于物理的渲染被加入到vtkPolyDataMapper中。这个着色模型是基于物理的渲染（PBR），可以使得表面的渲染效果更加真实。所有的PBR技术都是基于微平面理论，他认为，当进入到微观尺度之后任何的平面都可以被视为微平面。且他同时遵循能量守恒的原则。

漫反射：光线接触到一个表面的时候折射光是不会立即就被吸收的。光线实际上可以被认为是一束没有耗尽就不停向前运动的能量，而光束是通过碰撞的方式来消耗能量。每一种材料都是由无数微小的粒子所组成，这些粒子都能与光线发生碰撞。粒子的每次的碰撞，都可以吸收光线所携带的一部分或者是全部的能量而后转变成为热量。并非所有能量都会被全部吸收，而光线也会继续沿着（基本上）随机的方向发散，然后再和其他的粒子碰撞直至能量完全耗尽或者再次离开这个表面。而光线脱离物体表面后将会协同构成该漫反射颜色。不过在基于物理的渲染之中我们进行了简化，假设对平面上的每一点所有的折射光不会散开。但是当把散射考虑进去，也就会产生新的效果，比如许多类似的物质，如，皮肤，蜡烛，玉石等，它们都是光线进入物体内部散射然后从其他的位置传递出去。

粗糙度的影响：它影响着维表面的排列取向，当光线照射到表面上的时候，当一个平面的粗糙度越大，光线则更趋向与不同的方向发散开来，从而产生分布范围很广的镜面反射。光线照在一个粗糙度很小的平面上，光线则会趋向与同一个方向进行反射。造成更锐利的反射。由于微平面无法逐像素的运算，因此可以假设粗糙度来计算出某个相量方向。而且，由于微平面近似法使用能量守恒为：出射光线的能量不超过入射光线的能量，所有尽管随着粗糙度上升，镜面反射的区域增加但是亮度反而减小。

金属性的影响：金属表面对光的反映与非金属是不一样的，当光线照射过来的时候，除去镜面反射光的返回外，折射光都会被粒子所吸收，所以他并不会有漫反射的颜色，根据能量守恒可以得出进入材质内部的折射光是入射光减去反射光后余下的能力.

基于图像的照明：VTK中可以使用Cubmap纹理来制定当前的环境，Cubmap就是一个有六个面对立方体，在内部贴上图像从而也营造出空间场景感，可以通过引擎的处理使得Cubmap应用到反射和高光上

参数映射：ORM纹理的材质属性纹理的RGB通道分别是遮挡，粗糙度，金属度

法线贴图：在渲染的面上，每个面都有自己的法线，可以通过三角形两个边求得。而顶点的法线就是将公用此顶点的面的法线进行相加然后除以数量，得到的平均法线。当进行渲染平滑的时候，使用面法线进行光照计算，整个面公用一个法线，导致整个面光照强度相同，从而渲染结果不够光滑，但是使用点法线计算光线后，由于计算机自动插值的结果可以让整个平面变得更光滑。而法线贴图的作用就是使得低模可以具有高模的效果，当需要渲染一个很好的效果时，需要 一个精度很高的模型，但是相应的渲染的时间也会变得很长，但是换上精度不高的模型由会丢失很多细节，因此法线贴图可以让低模也能够获得较好的渲染效果。法线贴图就是使用贴图来存储一个面的法线方向，这样的话即使是一个平面，经过纹理的映射都可以计算出自己的法线朝向，然后再经过光照之后便可以更好的表现模型的细节，但是当摄像机近距离观察时，效果将会打折扣，因为只是改变了像素计算的结果，而并没有真正改变面的凹凸程度。

### 2.2.12 VTK交互器

可视化系统不仅需要强大的数据处理能力，拥有强大而且易用的交互功能同样十分重要。所以VTK同样具有强大的交互功能，他不仅能够像传统的交互一样监听鼠标或者是键盘触发的各种各样的事件，而且和可以在场景中渲染出提供各种实时交互的组件，UI的功能可以方便的设计各种交互。

观察者/命令模式：它是指Object可以有多可Observer，他定义了一个一对多的关系，当一个Object发生变化，所有和他有关的observer都被通知然后更新。

而在VTK中，实现两种方式来实现观察者/命令模式。对一种就是事件回调函数。第二种则为VTKCommand派生子类。

事件回调函数：通过AddObserver（）函数添加观察者到某个VTK对象中，当观察者观察到自己监视的事件发生的时候，他就会自动的调用回调函数，执行相关操作。

VtkCommand子类：设置的过程为vtkCommand类派生出子类，并实现vtkCommand：：Execute（）虚函数。然后再在主程序中实例化一个vtkCommand子类的对象已经调用相关的方法。最后再调用AddObserver（）函数监听目标事件。如果监听的事件发生，就会调用vtkCommand的子类中的Execute（）函数。而函数Execute（）也是执行具体功能的函数。

而另一种交互样式，则类似于在场景中放置了许多的UI界面，从而更容易观察场景的状态和交互的关系。这种交互方式为Widget交互。他的特点为每一种Widget都有提供不同的功能以及不同的API。但是，创建Widget的过程都是类似的。首先就需要实例化一个Widget，然后再指定VTK渲染窗口的交互器，Widget可以通过它来监听用户事件，做出响应。有的Widget需要使用观察者/命令模式来创建会回调函数，在和VTK的Widget进行交互的时候，他会调用一些通用的事件。再创建合适的几何对象，并用SetPrpresentation（）函数与Widget关联起来，或者使用默认的表达实体。

最后再激活Widget，这样就可以将它渲染到场景中了。

Widget的类型主要有，测量类Widget，标注类Widget，分割/配标类Widget，和其他的一些Widget。

### 2.2.13 VTK交互优化

图形系统在交互的时候会显的非常的慢，VTK的Level-of-detail技术对交互功能进行一个，优化，从而方便用户更流畅的进行实时交互。主要的原理就是在进行交互的时候可以用更低的分辨率来表示Actor从而达到足够的渲染速度。vtkLODActor是允许改变actor的几何表示的actor，而他的最低分辨率的模型是一个包围盒。

## 2.3 相关开发工具介绍

### 2.3.1 VTK

VTK作为一个开源、跨平台、可自由获取、支持并行处理的图形应用函数库，它的主要应用方向是三维计算机图形学，图形处理及其应用。VTK能够将实验数据以及模型逼真渲染，通过可视化来帮助理解科学结果。

主要特点包括：

1. 支持体绘制和面绘制的强大图性功能。
2. 具有非常好的流水线和高速缓存能力。
3. 支持比如JAVA和VRML等网络工具。
4. 支持多种着色器。
5. 具有设备无关性。
6. VTK中定义了许多宏，这些宏极大的简化了编程工作并且加强了一致的对象行为。
7. VTK具有丰富的数据类型，支持对多种数据类型进行处理。

# 3 总体设计

## 3.1 流体模拟与显示模型

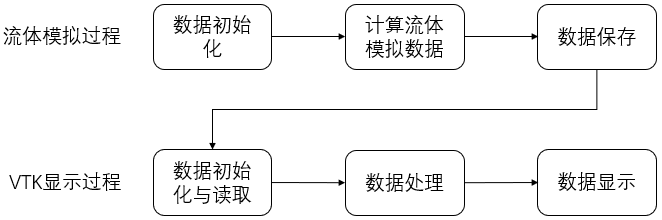
流体模拟与显示模型如图3.1所示。

图3.1 流体模拟与显示模型

该模型主要由流体模拟过程和VTK显示过程组成。

（1）数据初始化：主要内容为初始化MAC网格，流体粒子的生成和显示，和符号距离函数初始化等操作。

（2）计算流体模拟数据:他主要的功能是通过上一步定义的数据来循环计算流体模拟的相关数据，它的主要计算步骤为：计算时间间隔，对粒子位置依据速度场进行更新，通过数入无散度的数据场来计算对流项，通过输入速度计算受到体积力的速度更新，计算粘度流体对速度影响，根据压力对速度场的影响更新出速度，基于已有速度进行速度外推，约束固体边界外部的速度。

（3）数据保存：将计算结果的每一帧的速度场，流体中粒子位置等信息保存到特定的位置。

（4）数据初始化与读取：将流体模拟的计算结果读取到VTK中，并根据输入数据确定初始化的内容。

（5）数据处理：可视化管线通过输入的数据准备和处理好相应的图形可视化信息然后再传输给渲染引擎。

（6）数据显示：调整渲染引擎相关的参数，并且准备好Actor，摄像机，交互器等元素交由VTK渲染引擎渲染出结果。

## 3.2 功能模块划分

通过VTK流体模拟与显示模型分析，功能划分如图3.2所示。

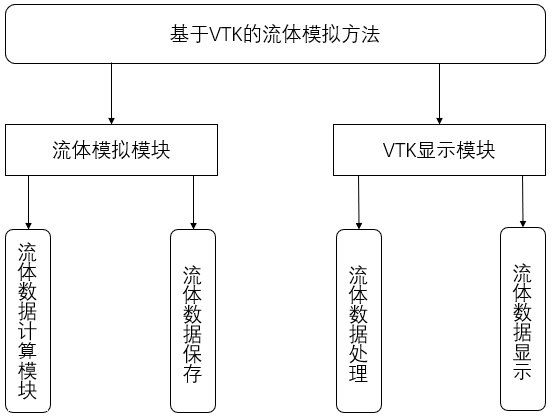


图3.2 功能划分图

（1）流体模拟模块：根据流体模拟的相关参数，计算出流体的每一帧的数据，再将程序计算出来的相关粒子数据保存在文件里面。

（2）VTK显示模块：通过读取流体模拟模块输出的每一帧的数据，通过渲染的流水线，将数据可视化显示出来，并且配有相应的交互，并且将查看的每一帧都能够保存成图片进行存储。

# 4 具体实现

## 4.1 流体模拟相关算法

### 4.1.1 求解对流项

对流是指由于流体宏观运动，从而使流体内部各部分发生相对位移，人们研究对流扩散方程，主要的研究对象是流体在流动过程中，流体所携带的某种物质的物理量的变化规律，例如传热过程中温度的变化规律或者溶解于流体中溶质的物质浓度等物理量的变化规律。

Dq/Dt=0。对应的函数是qn+1=，变量的解释为：表示速度场，表示时间间隔。qn表示传输的常量。通常情况下需要使用半拉格朗日法对速度场等一些数据进行更新。

当要搞清楚在网格某点处新的q值,要用半拉格朗日方法做的话，就需要算出粒子在格点处的旧值q. 粒子以u的速度在速度场中运动,因为知道它在哪里结束，所以为了找到它从哪里开始，我们简单地从网格点往回穿过速度场, 我们可以从这个起始点获取q的旧值，它将是网格点上q的新值。根据对平流的理解，假设一个粒子具有旧值qn，在处停止，当他通过速度场时，时间步长为Δt。可以得到，如果想要得到必须要知道。找出这个假想粒子的起点,起点就叫做，粒子的移动基于一个等式如果从时间上倒退,我们可以从XG倒退. 找到一个粒子在反向速度场下的终点，即

如果经过推算粒子的起点是在流体边界内部，那经过插值，不会出现问题，而如果经测算是来自流体边界外部，则可能是因为：

（1）液体从外部流入到内部：假设当液体以特定的速度v和温度t从流体区域一边的光栅流入,那么任何开始点超过这条边的粒子都应该得到速度v，温度也为t。

（2）数据错误：当是因为数据问题从而偏离流体边界的粒子轨道,合适的方法是在计算的目标位置边界最近的点推测值。

遇到这种情况只需要找到流体区域边界上的最近的点，然后从附近网格上存储的流体值中插入数值。如果我们找到的起始点在固体中的情况：在固体边界处的流体速度通常与固体速度不同，而流体速度的法向分量最好等于固体速度的法向分量。粘性流体和非粘性流体的处理也有不同。但是除了粘性流体，切线分量可以完全不同，因此，对于非粘性流体，通常在边界处插值流体速度，而不是简单地取固体速度。然而，对于粘性流动的特殊情况，确实可以采用固体速度来替代。

### 4.1.2 求解压力与不可压缩项

主要的流程就是先是通过求解压力场，然后通过求解出的压力场更新到速度场。压力，并且压力的处理也需要根据边界的情况进行修正。修正完成之后就会获得一系列的等式，其中每一个压力数值都是未知量，前面都有计算出来的系数，等式的右边则是输入函数的一系列速度参数。方程组的形式可以转换变成线性方程组并且应该压力从原来的二维或者是三维变量转化成一位的向量，同时将所有的系数全部都提取出来，组成一个大的系数矩阵，然后对于方程组的右侧，也因为全部都是已知的数，同样转化成为向量。也就是，需要将压力值定义了一个大的线性方程组为Ap=b。并且可以从概念上把它看作一个大的系数矩阵A，乘以一个由所有未知的压力组成的矢量p，等于一个负的速度的梯度b。

又因为A的每一行都是等于一个等式，例如，如果网格单元格(i, j, k)是流体的，那么就会有矩阵的一行用来表示一个方程。那一行的参数是方程中所有压强未知量的系数，但是又因为，假设在三维空间中 ，A的一行中除了（i，j，k）网格和周围的六个网格所带的参数不是另外, 几乎所有的参数都是0。因此A可以被认为是稀疏矩阵，我们只需要存储那些不是0的点

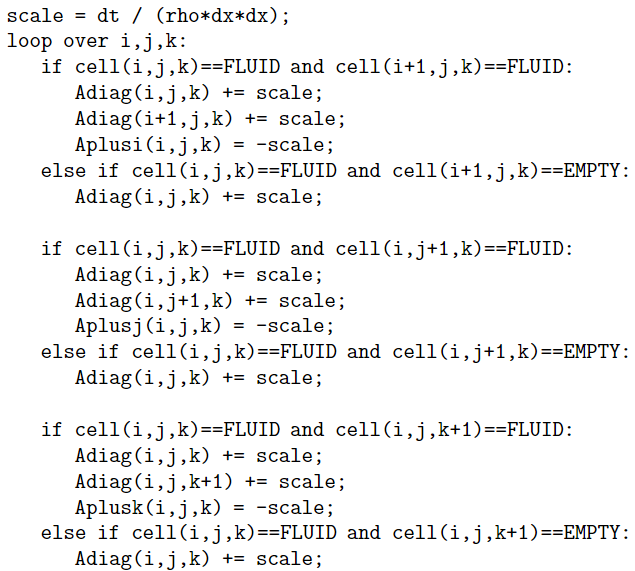
矩阵A的一个很好的性质就是它是对称的,因此只需要保存一半的值就可以。建立稀疏矩阵的伪代码如图2.12

图2.12稀疏矩阵伪代码

当求解出压力之后，就可以将数据带回到中就可以求出速度场了。

## 4.2 VTK可视化

### 4.2.1 可视化管线的构建

通过可视化管线可以准备好需要或者是生成所需要的数据，然后输入到渲染引擎中，所以它主要工作就是基本数据信息生成和整理。而在此程序的可视化管线阶段需要读取流体模型传输过来的流体点数据，生成相应的source对象，再通过Mapper使他能够映射成为相应的图元。为了展示流体流动的可视化表达，在VTK显示流体分子的时候可以使用球形来表现，一方面由于表面张力的缘故，使得液体单元的形状趋向于面积最小的形式，而球体是表现这一现象的合适对象。而且为了流体的表面光滑，也需要使用球体之类的几何对象，而对于外围的固体边界，最初的设置就是一个方形的空间的内部，挖空一个球体的区域，整个外部相当于固体，只有内部允许流体在其中自由运动，并且流体的设计也是生成一个球形小液滴。为了让内部的液体流动和固体的边界能够直观的显示，采用球形来表现固体边界，内部为自由空间，而外部全部为固体，并将其显示为半透明从而方便观察液体内部流动。

### 4.2.2 VTK渲染引擎构建

渲染引擎通过输入相应的图元，在经过图形渲染管线最后将图像显示在屏幕上。此过程需要创建相应的Actor对象，再将Mapper的数据输出和Actor的数据输入相连，然后需要定义相应的交互功能，VTK中的交互有渲染窗口交互，他可以对摄像机的位置角度进行变换，同时也可以与场景中对象进行交互，改变场景的位置与方向等。同时也可以设置相应交互风格。之后定义vtkRenderer，他的功能是相当于渲染场景，用于将所有的结果给显示出来，并且Actor等组件都是要添加到vtkRenderer中才可以渲染出来的，当VTK想要渲染出相应结果要显示在屏幕上的时候，则需要vtkRenderWindow，它作为VTK与操作系统之间交互的通道，可以将渲染的数据显示在屏幕结果上，并且由于具有设备无关性，所以并不需要担心编写平台的问题. vtkRenderWindowInteractor则可以监听鼠标、键盘和时钟消息，并通过VTK中的Command/Observer设计模式进行相应的处理.