

硕 士 研 究 生 读 书 报 告



题目 关于PBF流体模拟算法的理解

作者姓名 梁霄

作者学号 21851139

指导教师 高艺

学科专业 软件工程

所在学院 软件学院

提交日期 2018 年 12月

The Understanding of Position Based Fluids

A Dissertation Submitted to

Zhejiang University

in partial fulfillment of the requirements for

the degree of

Master of Engineering

Major Subject: Software Engineering

Advisor: Gao Yi

By

Liang Xiao

Zhejiang University, P.R. China

2018

摘要

本文主要是探讨Position Based Fluids（PBF）流体模拟算法。PBF算法是一种基于位置的流体模拟算法。PBF主要通过求解一组关于位置的恒定密度约束来近似保证流体的不可压缩性，这使得PBF能够在收敛到现代平滑粒子流体动力学（SPH）求解器的效果的同时，继承基于位置动力学（PBD）的稳定性，允许在大的时间步进上运行，从而满足实时应用程序的要求。PBF还使用了人工压力项来改善粒子分布，模拟表面张力。为了解决能量损失问题，PBF在速度的后处理过程中添加了涡量限制。从最终的结果来看，PBF与当时主流的PCISPH算法相比，能够在更大时间步进的情况下保证几乎相同不可压缩性的效果，大大提升了液体模拟的效率，达到实时液体模拟。

**关键词**：PBF，PBD，流体模拟, SPH

Abstract

This paper mainly discusses the Position Based Fluids (PBF) fluid simulation algorithm. The PBF algorithm is a position-based fluid simulation algorithm. PBF primarily guarantees fluid similar incompressibility by solving a set of positional constraints that enforce constant density, which allows convergence to the modern smoothed particle hydrodynamic (SPH) solvers, and inherits the stability of the position-based dynamics method. The stability of PBD allows operation over large time steps to meet the requirements of real-time applications. PBF also uses artificial pressure terms to improve particle distribution and simulate surface tension. In order to solve the energy loss problem, PBF adds a vorticity confinement during the post-processing of velocity. From the final result, compared with the PCISPH algorithm, PBF can guarantee almost the same incompressibility in a larger time step, greatly improving the efficiency of liquid simulation and realizing liquid real-time simulation.

**Keywords：**position based fluids, position based dynamics, fluid simulation, SPH

1引言

流体，特别是液体（水），是许多丰富视觉现象的原因。在图形学中，流体的模拟一直以来都是一个十分有趣而且很有挑战性的领域。但是基于力（Force-based）的模拟方法存在着大时间步进会导致不稳定的问题，这就意味着对实时性应用不能有很好的效果。Position Based Fluids（PBD）[1]是一种基于粒子方法的流体模拟算法。PBF的文章在SIGGRAPH 2013大会上发表，虽然文章很短，但是其展现出了十分惊艳的实时流体模拟的效果。本文主要探讨基于位置流体（Position Based Fluids PBF）算法的相关工作。

**2 流体模拟的相关工作**

首先介绍一下一些流体模拟的相关概念的定义。

不可压缩性（incompressibility）指物体密度变化小到可以忽略。在流体模拟中，模拟流体的不可压缩性对于其真实性来说是十分重要的，但是保证不可压缩性通常也需要大量的计算。

时间步进（time step）指将一个时间的连续函数离散成一系列点，每个相邻点之间的时间间隔。具体在流体模拟中指两次相邻的粒子状态的计算操作对应时间的间隔。时间步进越大，代表每一帧状态所需的计算次数越少。

模拟流体有许多种技术可以选择，其中比较主流的是粒子方法。粒子方法有简单性和灵活性的特点。其主要思想是通过大量的离散粒子的运动及相互作用来重构连续物体的运动规律。用粒子方法模拟流体主要需要解决的问题是如何定义粒子之间的相互用及其对位置和速度的影响。

平滑粒子流体动力学(Smoothed Particle Hydrodynamics SPH)[2]是一种众所周知的基于粒子的流体模拟方法。SPH方法主要是基于对单个粒子进行受力分析，通过粒子受到的力算出加速度，从而推导出粒子的运动，所以SPH方法是一种Force-based的方法。文中指出SPH具有许多好的特点：质量守恒，拉格朗日离散化和概念简单。然而，SPH对邻居粒子的缺少造成的密度波动很敏感，并且由于模型的非结构化特性，保证不可压缩性的成本很高。预测-纠正的不可压缩SPH（[Predictive-corrective incompressible SPH](https://dl.acm.org/citation.cfm?id=1531346) - PCISPH）[3]将SPH的效率提高了一个数量级，但是其小的时间步进仍然限制了实时应用程序。

基于位置的动力学(Position Based Dynamics PBD)[4]提供了一种基于Verlet积分的在游戏中模拟动力学的方法。PBD使用高斯迭代求解一系列非线性约束，从而直接更新粒子的位置，再通过粒子位置地变化求出粒子运动的变化，这就是PBD中position based的由来。PBD也因此能够避开显示地求解力，通过位置更新隐含地推导出动量变化，从而避免了显式方法带来的不稳定性。

**3 Position Based Dynamics**

PBF是基于PBD方法的，通过使粒子的位置满足密度约束来修正粒子的位置。因此为了更好地理解PBF算法，了解一下关于PBD的知识是十分有必要的。

首先探讨的是PBD算法的来源。这就不得不提到Position Based和Force Based，上面已经提到过两种方法的区别，总结来说就是，FB通过力求位置和速度，PB通过位置求速度。PBD最开始应用于模拟柔性物体形变，而模拟柔性物体最开始是使用Force Based方法。

FB方法存在一个难已解决的问题，就是上面提到的时间步长问题。FB方法从给定外力求解位置的过程中有两种方法，一种是显式方法，一种是隐式方法。显式方法计算量少，但该方法是条件稳定的，要求时间步进小于某个阈值h0才能保证稳定。在计算力的时候需要制定一个类似弹性系数的参数，该参数与h0成反比，小的弹性系数会导致模拟物体的不真实，而增大弹性系数就使的h0变小，进一步限制时间步进。而隐式方法是无条件稳定的，但是计算量较大[5]。于是就有人提出了PBD方法。

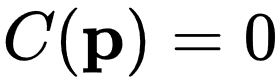
PBD中有一个十分重要的概念，约束（Constraint）。约束实际意义就是你要模拟的物体中的一些约束条件，比如我们希望布料不要被拉长，则可以定义一个距离约束C(p1,p2) = |p1 – p2| - d，保证该约束就能保证两个粒子之间的距离保持不变。约束C有5种特性：

1. 约束的基数n，代表约束影响的顶点数目
2. 约束是一个值为实数的函数，
3. 约束中顶点的索引值为{i1，…，in}，ik∈[1,…,N]
4. 约束强度k∈[0,1]
5. 一个约束要么是等式约束C = 0，要么是不等式约束C ≥ 0

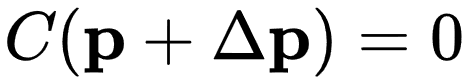
有了约束的定义，就可以理解PBD的主要思想了。PBD核心步骤有两步：

1. 将不能转化为位置约束的力（重力）直接通过显示形式积分求出预测位置。
2. 应用每一个约束（在论文里将其称为约束投影），修正上一步预测的位置称为最终的位置。

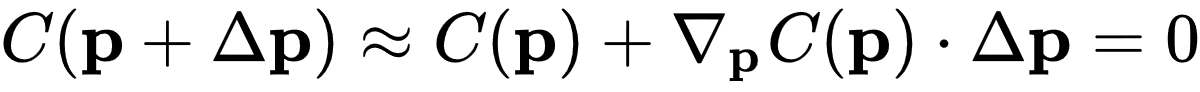
了解了PBD的步骤，那么如何去求解约束，进行约束投影。首先定义一个基数为n的约束，约束影响的粒子编号为1,…n，粒子的位置设为p1，…pn，设p=[p1T,…, pnT]T,则约束可以表示为：

 （1）

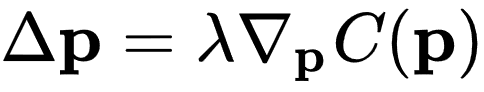
PBD的目标是找到位移向量，使处满足约束条件：

 （2）

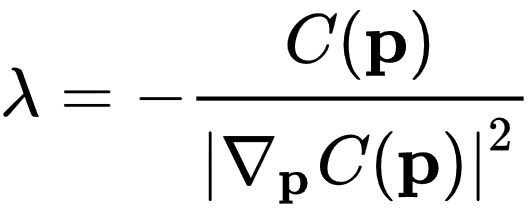
对（2）式在p处进行一阶泰勒展开：

 （3）

在PBD中，的方向被限制约束梯度的方向上，即：

 （4）

这样做的理由是，如果求解出的存在垂直于的分量，则表示约束投影粒子的位移做出了调整，但是约束的误差却没有产生变化，即垂直于方向的位移对于约束的投影没有任何作用，还会引入一种幽灵力，使粒子做出无法理解的运动[6]。于是，（3）式中求解就被简化了为了求解λ，λ的公式：

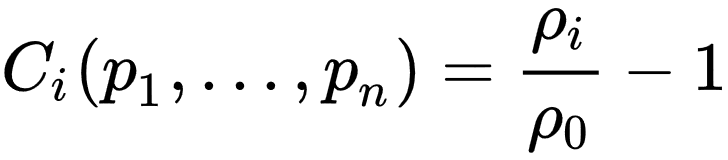


（5）

然后将解出的λ带入（4）即可解出。以上就是PBD的核心思想和方法。

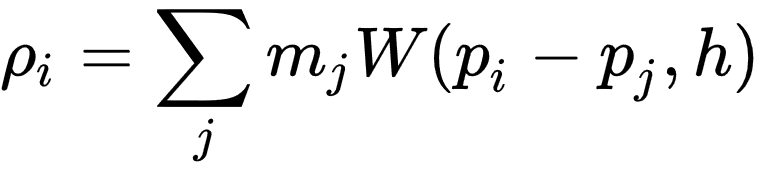
**4 保证不可压缩性**

首先，PBF通过约束来保证液体的不可压缩性的。PBF设计了一个拥有非线性约束的系统，系统中的每个粒子i都存在一个密度约束，表示为Ci，其中每一个约束都是一个关于粒子i及其邻居粒子位置的函数，设粒子的位置为p1, …,pn，则Ci由下面的表达式定义：

（6）

其中ρ0代表液体的静态密度，ρi代表粒子i当前的密度。

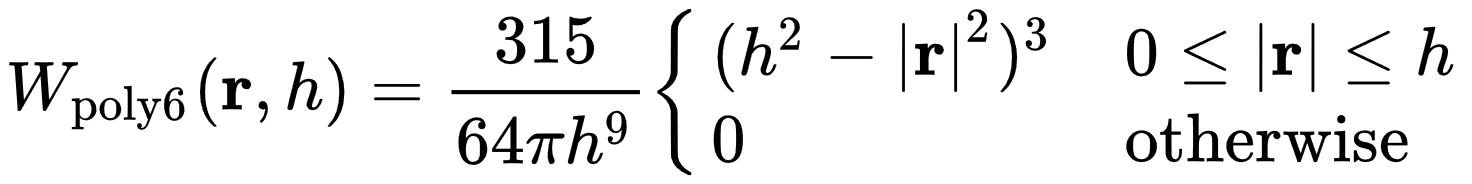
这个公式来源于Bodin等人在2012年工作[7]中使用的状态方程。可以这样来理解，为了满足不可压缩性，粒子i的密度应该尽量与静态密度ρ0相同。这是一个等式约束，即在Ci = 0的时候，约束被满足。

其中粒子i的密度由标准的SPH密度估计器给出：

（7）

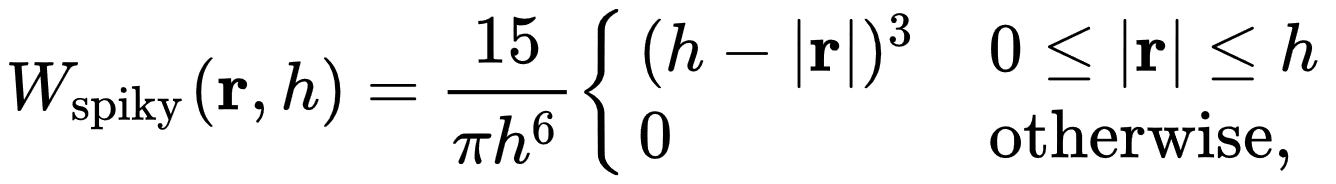
因为在系统中的粒子的质量都是相同的，所以后续的方程中删除m项。W（p，h）是光滑核函数，p是距离，h是光滑核的半径。光滑核函数有两个重要的属性，是偶函数和规整函数（即）。使用该函数能够得到邻居粒子对中心点的影响大小。

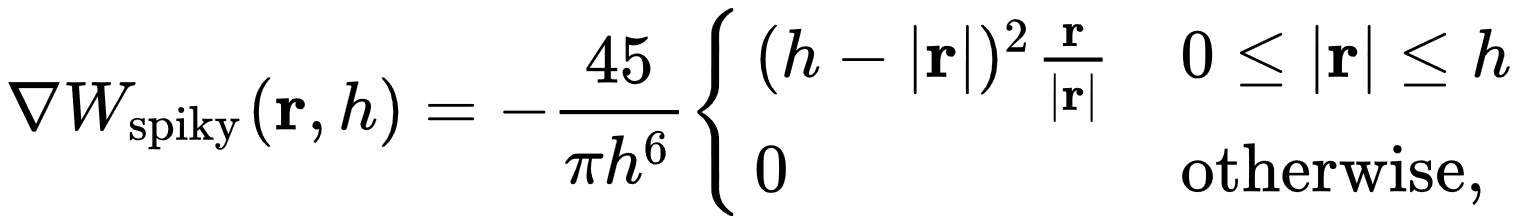
PBD在实现中使用Poly6内核进行密度估计.

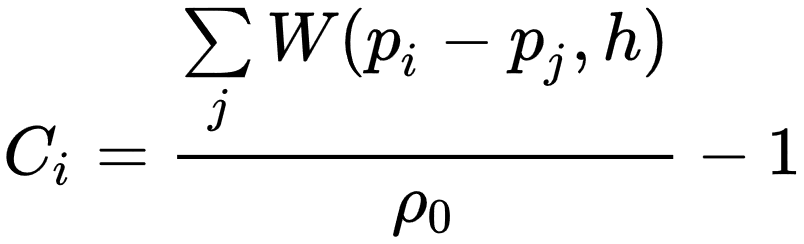


（8）

使用Spiky内核进行梯度计算。

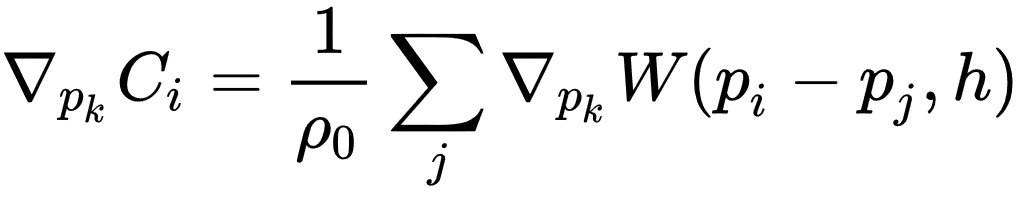
 （9）

（10）

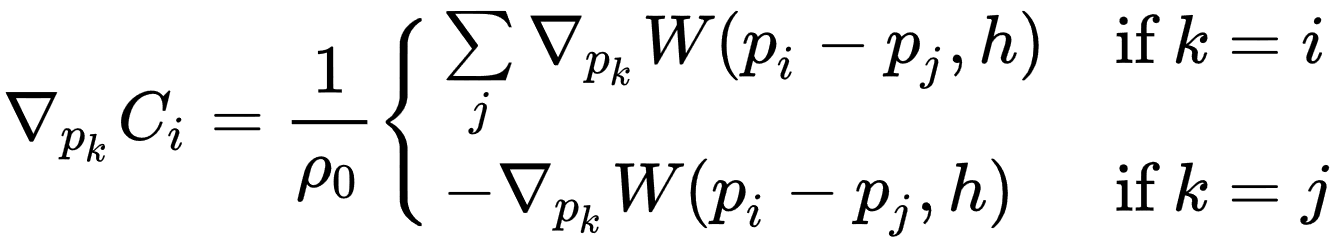
将（7）带入（6），粒子i的约束函数Ci是一个关于其位置pi及其邻居粒子位置pj的非线性方程函数，Ci(p1,…pj) = 0 则成为了一个非线性方程，所有的粒子的约束函数就组成了一个非线性方程组。Ci可以表示为：

（11）

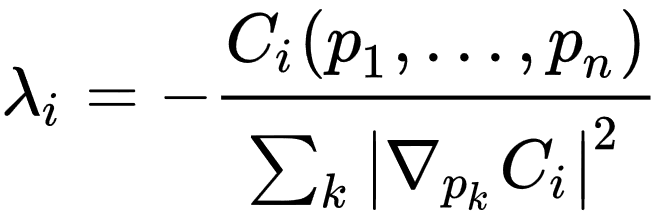
计算其梯度:



(12)

这里有两种情况，k=i，即约束关于粒子i的梯度，和k≠i，即k=j，j是i的邻居粒子：

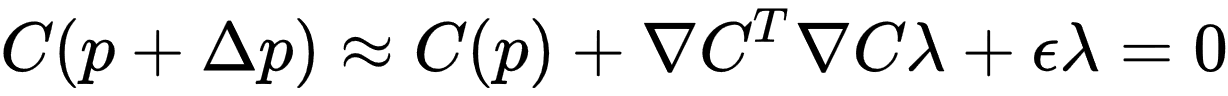
（13）

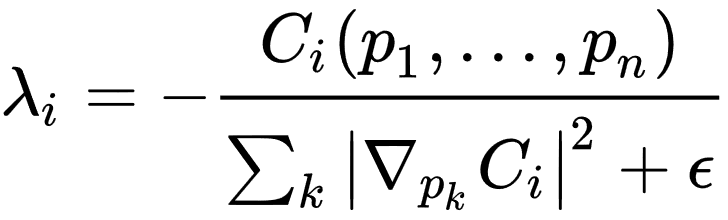
结合（13）式和PBD方法求解λ的式子（5），得到求解公式：

（14）

这个λ的值对于Ci中的所有粒子都是一样的。

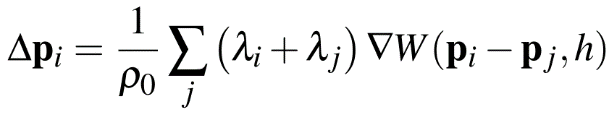
这里出现了一个问题，非线性的约束函数导致在pi – pj接近h的时候会变成零，有可能会导致计算除零，引起不稳定，所以PBF方法结合约束力混合（constraint force mixing CFM）方法，将求解出的约束力混合回原约束函数来软化约束，具体公式将会变成：

 （15）

这里ε是松弛参数，可以由用户指定。（14）式则变成了：

（16）

在算法中要首先求出所有的λ，对于每个粒子i，它不仅要满足自己的约束Ci，还要满足其邻居的约束Cj。因此最后的公式变成了：

 （17）

5 **拉伸不稳定性**

对于采用SPH插值技术计算密度的流体模拟方法，通常需要30∼40个邻居粒子才能使密度求值的结果趋于静态密度。在邻居粒子不足的情况下，会导致通过公式(1)求出的流体密度低于静态密度，由此造成压强为负数，原本粒子间的压力变为吸引力，使粒子产生不符合实际情况的凝聚(Tensile Instability)，这导致的结果是流体表面的模拟给人感觉不真实。

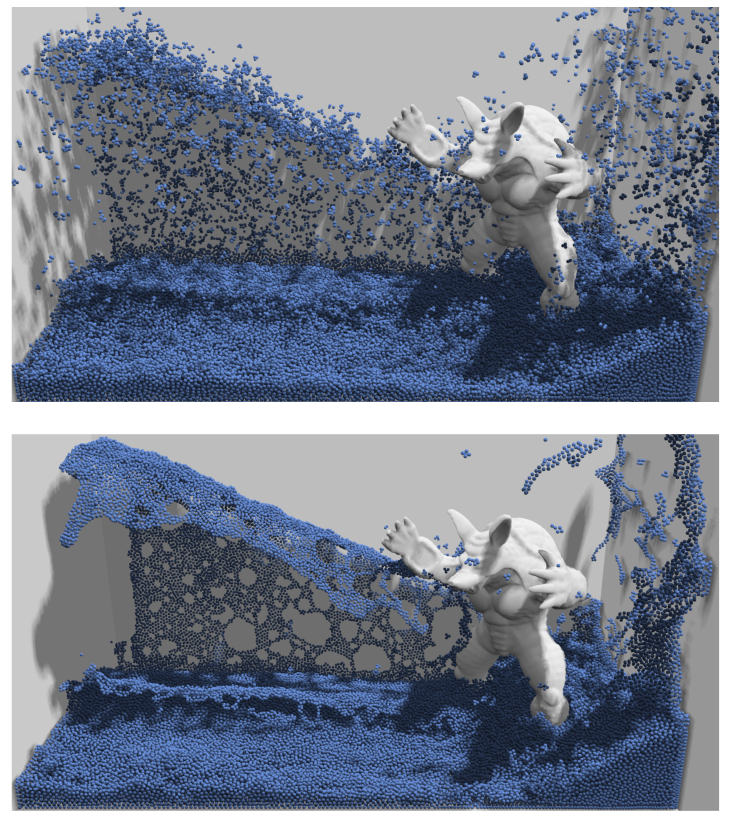
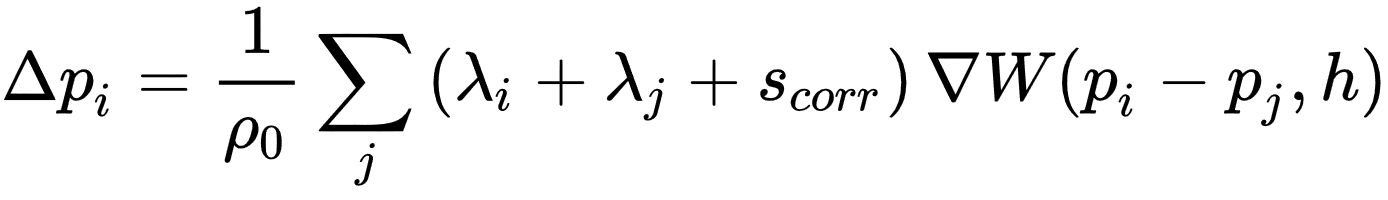
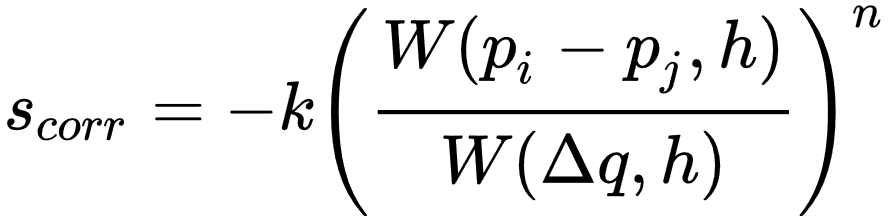


图1上图是负压引起的粒子聚集，下图是添加人造粒子项后的效果

为了解决这个问题，PBF采用了一种类似人工排斥力的方法公式(17)的基础上，加入排斥项εcorr：

 （18）

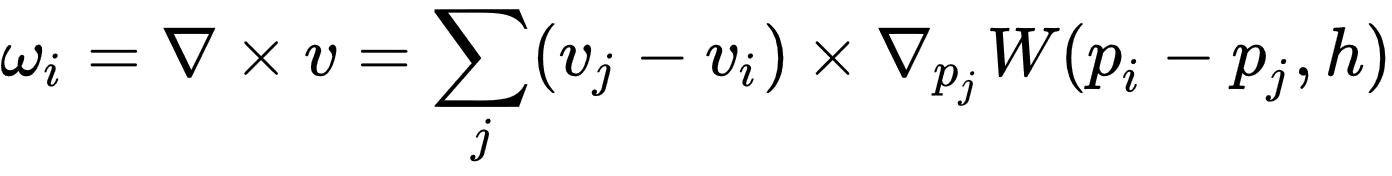
其中εcorr的表达式为：

（19）

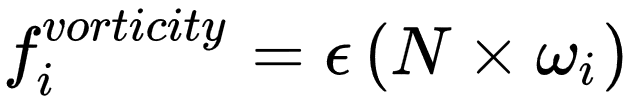
其中表示距离粒子i固定距离的零点，通常取。文中提到，这个排斥项使得粒子密度略低于静止密度，使粒子向内拉动它们的邻居，引起类似表面张力效应，因此k可以看做表面张力参数。

**6 涡度约束和粘度**

PBD方法通常会引入额外的阻尼，导致整个系统的能量损耗，由此会导致本来该有的一些涡流(vortices)快速消失。PBF选择使用涡量限制（vorticity confinement）来补充损失的能量，向系统重新注入能量。首先计算粒子所处位置的涡度：

 （20）

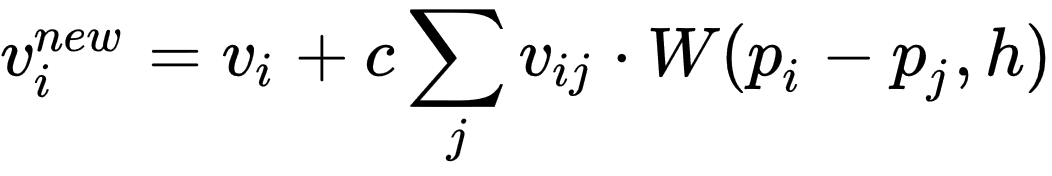
然后计算矫正力：

（21）

其中N = ，。

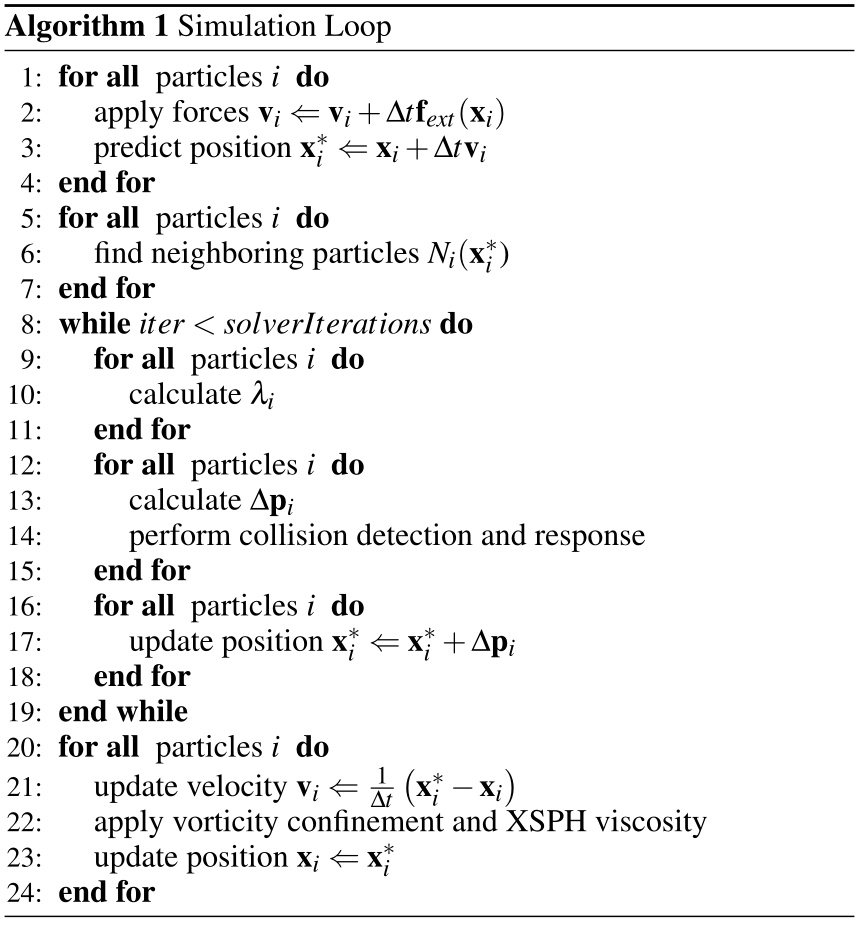
PBF没有使用归一化的ω，因为这会不加选择地增加涡度。相反，使用非标准化值能够只在已经存在的涡度的地方增加涡度。

此外，PBF使用了XSPH粘度，这对于相干运动很重要。在PBF的模拟中，参数c通常选择为0.01：

 （22）

**7 算法**

具体的算法在算法1中描述。它类似于原始的PBD的更新，但是其每个约束以雅可比方式独立解决，而不是通过顺序Gauss-Seidel迭代。同时固体的碰撞检测也被作为约束求解循环的一部分：



文中提到因为算法每一个时间步进要重新计算一次粒子邻域，在每个求解器迭代中重新计算距离和约束值。这种优化方式可能在某些情况下导致密度低估，例如，粒子与初始邻居集合分离时。在PCISPH中，这会导致严重的问题，一旦粒子变得孤立，每次迭代都会使其压力越来越大。如果它随后在后续迭代中重新接触，则施加错误的压力。然而PBF算法仅考虑当前粒子位置（不是累积压力），因此不会发生这种情况。

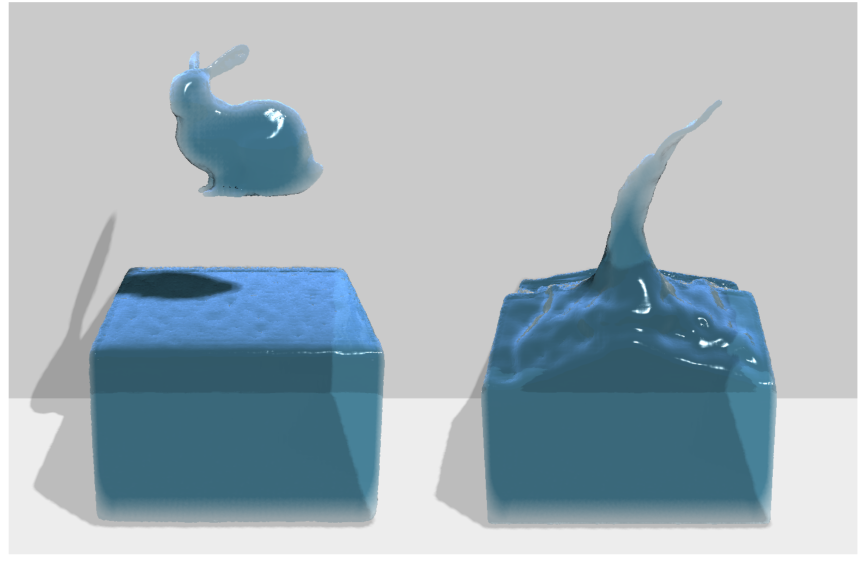


图2 兔子掉进水中

**8 结果**

文中通过将液体兔子放入水池中来测试PBF的算法（图2），并将结果与PCISPH的结果进行比较。PCISPH在每帧少于10个子步骤（Δt=0.0016s）时表现不稳定。相比之下，PBF算法只需一个子步骤即可稳定（Δt=0.016s）。

为了比较对不可压缩性的模拟，对比用的PCISPH每一秒有10个子步骤，每个子步骤4个迭代，PBF算法采用4个子步骤，每个子步骤10次迭代，因此每帧总共执行40次迭代。这种比较的目的是表明PBF方法可以通过更大的时间步长实现相同的效果，从而允许PBF算法通过更多的密度迭代来分摊每一步进中网格构建和邻居粒子发现的成本。

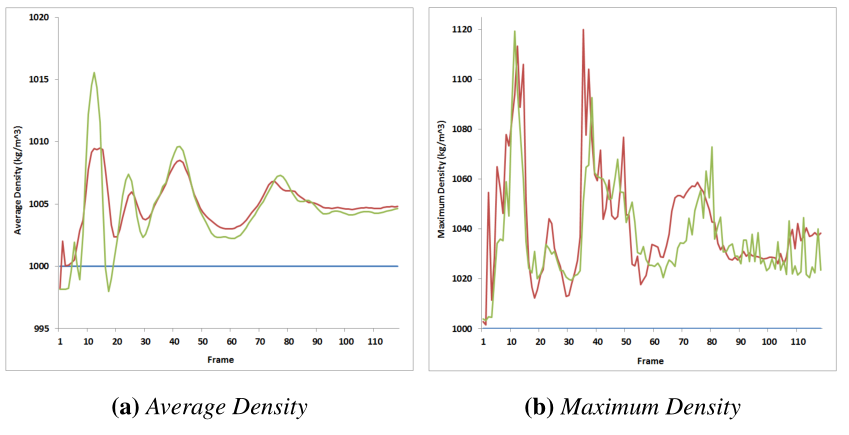


图3 蓝线静止密度，红线PCISPH方法，绿线PBF

实验的密度图结果非常相似，证明了压缩性水平相似，但是PBF方法有更大的时间步长（图3）。

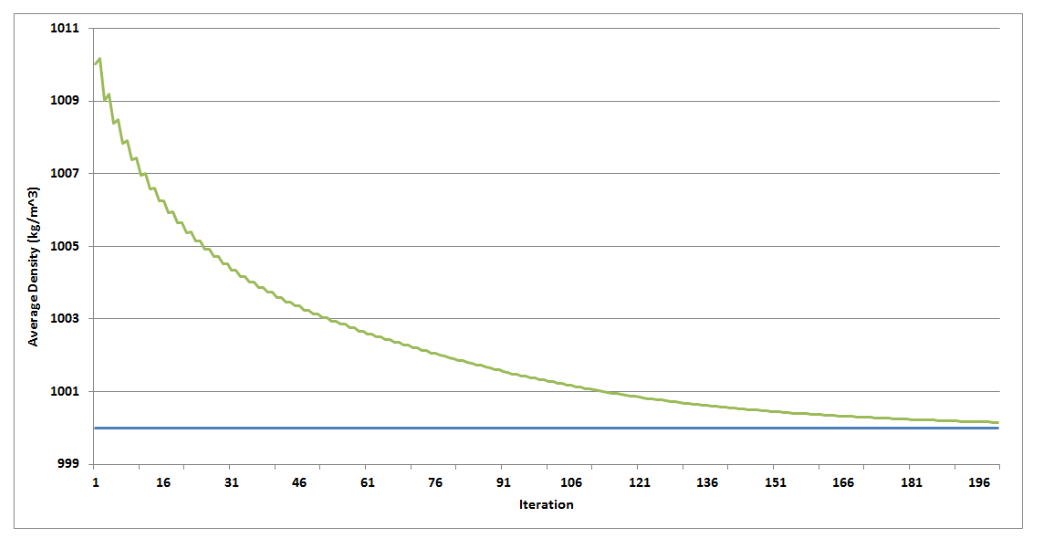


图4 t=1.0时多次迭代的收敛情况

因为PBF对具有可预测性能的实时应用程序感兴趣，所以将迭代次数设置为固定值（通常为2-4）。同时，PBF在图4中还展示了的多次迭代中的收敛性，这代表如果抛开实时性，PBF能够表现出更好的效果。

参考文献

[1] Macklin M, Müller M. Position based fluids[J]. ACM Transactions on Graphics (TOG). 2013 Jul 21;32(4):104.．

[2] Monaghan JJ. Smoothed particle hydrodynamics[J]. Annual review of astronomy and astrophysics. 1992 Sep;30(1):543-74.

[3] Solenthaler B, Pajarola R. Predictive-corrective incompressible[J] SPH[C]. ACM transactions on graphics (TOG). ACM, 2009, 28(3): 40.

[4] Müller M, Heidelberger B, Hennix M, Ratcliff J. Position based dynamics[J]. Journal of Visual Communication and Image Representation. 2007 Apr 1;18(2):109-18.

[5] [peterntu](http://www.opengpu.org/home.php?mod=space&uid=46629). 关于Position based dynamics [N/OL] OpenGPU，2016-3-14

[6] [Gfans](https://www.zhihu.com/people/gfans) 深入浅出 Nvidia FleX [N/OL] [Constraints & Solvers](https://zhuanlan.zhihu.com/GfansPhysics) 2018-11-17

[7] Bodin K, Lacoursiere C, Servin M. Constraint fluids[J]. IEEE Transactions on Visualization and Computer Graphics. 2012 Mar 1;18(3):516-26.