# 1程序概述

主要在WCSPH基础上进行增改，采用PCISPH

**仅说明增改部分。**

## 理论说明

PCISPH的全称为Predictor-Corrector Implicit SPH。

其实质为**通过修正压力梯度力来最小化密度误差**。

其与WCSPH区别仅在于计算压力部分（即计算压力梯度力之前，计算非压力梯度力之后的部分）。

其大致流程为：

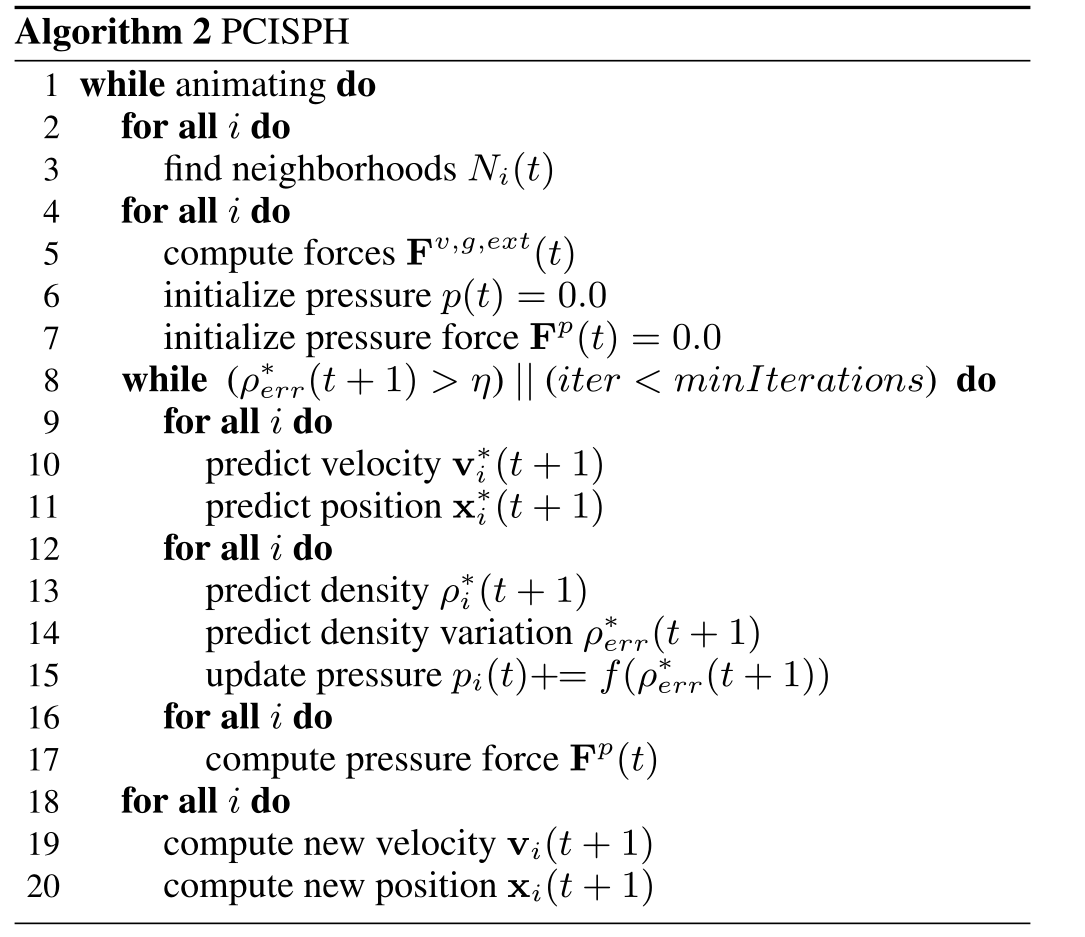
1. 预测速度和位置（即时间积分，是与预测的压力梯度力有关的）
2. 解析碰撞
3. 通过压力梯度力计算密度误差（预测的位置处使用核函数公式）
4. 计算修正后的压力梯度力（只要一个系数乘以密度差）
5. 计算的密度误差比是否小于给定误差，如果是，跳出循环，否则回到第1步

上述循环输出为符合条件的压力梯度力。跳出条件为密度误差。

上述循环可以认为是一个**功能模块**，目的就是通过**最小化密度差求解压力梯度力。**

最小化的目标函数是密度差，自变量是压力梯度力。两者联系的方式为：预测的压力梯度力施加到粒子上，粒子前进后得到预测的位置，在预测的位置处使用核函数公式就得到了预测密度，与静止密度作差或者做商就得到了密度误差（本文中做商）。

## 算法流程



算法参考自

B. Solenthaler and R. Pajarola. 2009. Predictive-corrective incompressible SPH. ACM Trans. Graph. 28, 3, Article 40 (August 2009), 6 pages. DOI:https://doi.org/10.1145/1531326.1531346

|  |  |
| --- | --- |
| **Algorithm 1: PCISPH** | |
| **Input:** 粒子旧位置速度 | |
| **Output:** 粒子新位置速度 | |
| **1.** | **while** 运行 **do** |
| **2.** | 邻域搜索 |
| **3.** | 计算非压力梯度力 |
| **4.** | 为压力和压力梯度力置零 |
| **5.** | 计算系数delta |
| **6.** | **while** 最大密度误差> 0.1% **do** |
| **7.** | 预测位置x\*和速度v\* |
| **8.** | 边界处理 |
| **9.** | 预测密度rho\*(根据预测位置对核函数求加权和) |
| **10.** | 计算密度误差rho\_err和最大密度误差 |
| **11.** | 更新压力（p+= delta \* rho\_err） |
| **12.** | 计算压力梯度力 |
| **13.** | 时间积分求新位置速度 |

# 2程序主要模块说明

4-12为核心循环, 放到一个功能模块里面，命名为PciPressureSolver

**与WCSPH一致的部分**

其中前处理的邻域搜索，计算粘性力和重力2个模块，后处理的时间积分模块都是与WCSPH完全一致的。

**与WCSPH比稍微改动的部分**

其次，预测位置速度，边界处理，预测密度和计算压力梯度力这4个模块只需要略微改动。改动如下：

**预测位置速度**：将加速度换为预测的加速度，包括已知的非压力梯度力和预测的压力梯度力这两部分。其中压力梯度力是不断迭代变化的。

**边界处理**: 由于预测后的位置可能穿透边界，所以需要穿插一个边界处理。与WCSPH不一样的部分仅仅在于输入的参数变为了预测的位置和速度。

**预测密度**：只需要把位置换为预测的位置

**更新压力梯度力**：只需要把密度和压力换为预测的密度和压力

**与WCSPH比增加的部分**

增加2个模块，即更新压力（p~ = delta \* rho\_err）和计算delta

## 2.1 预测位置速度

**函数/模块名称**：predictVelocityPosition()

**输入**：预测的压力梯度力（初始为0），给定的粘性力和重力(非压力梯度力)，给定的原位置速度（上一时刻的）

**作用**： 根据预测的压力梯度力计算预测的位置和速度

**输出**：预测的位置和速度

**注意**：其中，

a\_nonP就是

a\_P就是

可以直接带入公式，不必单独存储a\_P和a\_nonP了。

## 2.2 预测密度

**函数/模块名称**：computeDensityStar ()

**输入**： 预测位置x\*

**作用**：计算密度场，计算公式如下：

**输出**：预测密度场density\*。

**注意**：预测的粒子的邻域本来需要额外进行一次邻域搜索，但是为了节约计算量，就直接使用上一个时间步的邻域链表了。这样会造成误差。假如高估了密度，那么会导致最终计算密度在阈值（0.1%）以下震荡；假如低估，会导致计算失败。

## 2.3 计算密度误差

**函数/模块名称**：computeDensityErr()

**输入**： 预测密度rho\*和静止密度rho0

**作用**：

计算密度误差

计算最大密度误差（收敛条件）

**输出**： 和，返回平均密度误差

**注意**：也可用最大密度误差。当密度大于静止密度，即粒子过密，压力则为正数，即推开粒子。反之亦然。

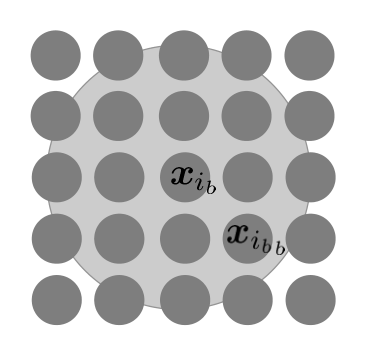
## 2.4计算PCI系数delta

**函数/模块名称**：computePciDelta()

**输入**：无

**作用**：

采用perfect sampling 来生成初始的完美采样。其采样示意图如下：



**输出**：

## 2.5 更新压力

**函数/模块名称**：updatePressure()

**输入**：系数, 密度误差

**作用**：

**输出**：压力

**注意**: 这个预测的压力可以直接存到压力里面去，当PCI的迭代结束，最后出来的压力就是符合条件的压力。

## 2.5 更新压力梯度力

**函数/模块名称**：updatePressureForce()

**输入**：压力 ，密度

**作用**： 更新压力梯度力，用于迭代

**输出**：压力梯度力

**注意**: 这里的算法与WCSPH完全一致，只是密度和压力换为了预测值。

# 3 变量说明

同样，我们只说明增改部分

## 3.1 常量

密度误差限度 1e-3

## 3.2全局场

场大小均为 numPar

预测位置场 positionStar

预测速度场 velocityStar

预测密度场 densityStar

密度误差场 densityErr

**3.3 其他新增变量**

有些可以直接通过返回值传递，不必单独开辟空间。

PCI系数delta

最大密度误差maxRhoErr

# 4 DEBUG记录

编号33的粒子的坐标[30.632456 20.632456]

33的邻域：

第0个: 编号0,坐标[30. 20.]

第1个: 编号1,坐标[30.632456 20. ]

第2个: 编号2,坐标[31.264912 20. ]

第3个: 编号32,坐标[30. 20.632456]

第4个: 编号34,坐标[31.264912 20.632456]

第5个: 编号64,坐标[30. 21.264912]

第6个: 编号65,坐标[30.632456 21.264912]

第7个: 编号66,坐标[31.264912 21.264912]