简单流体模拟算法（魔改版）

TAs

2021 年 8 月 3 日

1 问题背景

本题目讨论的是**无外力作用的不可压缩流体**的模拟。在接下来对于二维平面的讨论中，认为纵向方向为 x 方向，横向方向为 y 方向。

模拟的区域为 $x \in (0, h), y \in (0, w)$

为了方便描述边界条件和计算梯度，约定：流体被约束在一个矩形盒子中，四边分别为：

- $x = 1$
- $x = h - 1$
- $y = 1$
- $y = w - 1$

这个盒子以外的数值都是未定义的，检查时对这些位置的值没有要求，因此可以随意存储任何数值。

2 算法概述

如果将流体的速度场表示为 $\mathbf{u}$，压强场表示为 p，以下是不可压缩流体的 Navier–Stokes 方程：

$$\frac{\partial \mathbf{u}}{\partial t} = \underbrace{-(\mathbf{u} \cdot \nabla)\mathbf{u}}_{1} \underbrace{-\frac{1}{\rho}\nabla p}_{2} \underbrace{-\nu \nabla^2 u}_{3} + \mathbf{g} \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0$$

其中 g 表示外力导致的加速度，由于无外力作用，$\mathbf{g}=\mathbf{0}$。其他部分的含义分别是：

1 Advection 表示随流体本身的流动，把某一处的速度带到相邻位置

2 Pressure 表示局部的压强不均导致的速度变化

3 Diffusion 表示因为流体粘度带来的局部的速度扩散

为了求以上方程的数值解，在这里需要用到一个数学结论：*Helmholtz-Hodge* 分解称任意向量场都可以被唯一分解为以下形式：

$$\mathbf{w}=\mathbf{w_n}+\nabla q \tag{2}$$

其中 q 是标量场，$\mathbf{w_n}$ 散度为 0：$\nabla\cdot\mathbf{w_n}=0$。据此可以定义算子 P 表示将一个向量场映射到它的散度为 0 的分量上：$\mathsf{P}\mathbf{w}=\mathbf{w_n}$。

将 P 应用在等式 1 两侧：

$$\mathsf{P}(\frac{\partial\mathbf{u}}{\partial t})=\mathsf{P}(-(\mathbf{u}\cdot\nabla)\mathbf{u}-\frac{1}{\rho}\nabla p-\nu\nabla^2 u) \tag{3}$$

其中，由于 $\nabla\cdot\mathbf{u}=0$，而 ∇p 本身就是一个梯度场，因此 $\mathsf{P}(\frac{\partial\mathbf{u}}{\partial t})=\frac{\partial\mathbf{u}}{\partial t}$，$\mathsf{P}\nabla p=0$ [1]：

$$\frac{\partial\mathbf{u}}{\partial t}=\mathsf{P}(-(\mathbf{u}\cdot\nabla)\mathbf{u}-\nu\nabla^2 u) \tag{4}$$

根据方程 4 可以构造数值模拟算法，将连续时间切分为时间片，在每个时间片内分为以下几步:

- 进行流动 (Advection) 的模拟，更新速度场
- 进行速度扩散 (Diffusion) 的模拟，更新速度场
- 将速度场投影 (Projection) 到其散度为 0 的分量上

[1] 可以认为这个投影过程对应的是：方程右侧除了压强项以外的非零散度造成了流体的局部挤压、拉伸，这会造成不均的压强场，它带来的加速度会让方程右侧的散度整体等于零。损失的速度耗散成了热。

2.1 P 算子的计算方法

对方程 2 两侧同时计算散度，由于 $\mathbf{w_n}$ 散度为 0：

$$\nabla \cdot \mathbf{w} = \nabla^2 q \tag{5}$$

$\mathbf{w}$ 是已知的，可以直接计算它的散度，因此这是一个关于 q 的泊松方程，在流体边界 ∂D 上的边界条件是 $\frac{\partial q}{\partial \mathbf{n}} = 0$

在解得 q 以后，$\mathbf{w_n} = \mathbf{w} - \nabla q$。

3 实现细节

3.1 场的离散表示和算符实现

计算中储存的所有场都会被离散化为边长为 1 的正方形的格子，存储在 Numpy 数组中。`a[x][y]` 被定义为第 x **行**，第 y **列**的格子正中央的场的值。

据此，我们可以定义梯度、散度和拉普拉斯算符的离散形式：

```
grad(x, y) = (
  (a[x+1][y] - a[x-1][y]) / 2,
  (a[x][y+1] - a[x][y-1]) / 2,
)
div(x, y) = (a[x+1][y] - a[x-1][y] + a[x][y+1] - a[x][y-1]) / 2
laplacian(x, y) = a[x+1][y] + a[x-1][y] + a[x][y+1] + a[x][y-1] - 4 * a[x][y]
```

你可以使用 `np.roll` 将整个场进行平移，这样可以并行计算整个场的梯度、散度或者拉普拉斯算符应用后的结果。这样会带来一个问题：`np.roll` 对溢出的行为是循环滚动，这样一侧最靠边的值会被移动到另一侧，这是没有意义的。

根据我们对于模拟域的约定（见第 1 节），边界正好分别位于最前两行、最前两列、最后两行、最后两列之间。由于在计算上述算子的时候，最多只需要将场平移一格，因此只需要在每次更新场时将边界外的值覆写，就会解决边界问题。

3.1.1 边界条件

选择边界外的值被覆写的内容需要和 Navier-Stokes 方程的边界条件一致。我们选用 No-slip 边界条件：在边界上流体速度为 0：$\mathbf{u} = 0$。这可以通过将模拟域内部的值取反复制到边界外得到。

```
-   - - - - - - - ...
    ----------------
_ | a b c d ? ? ? ...
_ | x ? ? ...
_ | y ? ? ...
...

To:

_   -a -b -c -d  ? ? ? ...
     ^--^--^--^--^-^-^----
-a < a  b  c  d  ? ? ? ...
-x < x  ?  ?  ...
-y < y  ?  ?  ...
...
```

在 2.1 提及了计算 P 算子的方法，其中会需要标量场 q 计算梯度。q 的边界条件是垂直于边界方向的变化率为 0，因此需要直接把模拟域内部的值复制到边界外。以上两个过程可以实现为同一个函数：

```
def set_boundary(field, factor):
  field 的第一行 <- field 的第二行 * factor
  field 的最后一行 <- field 的倒数第二行 * factor
  field 的第一列 <- field 的第二列 * factor
  field 的最后一列 <- field 的倒数第二列 * factor
```

在以下过程中中将会多次用到这一函数。

3.2 Advection

为了保证算法的稳定性，我们采用隐式方法：根据某一点的速度，找到上一个时间片这一点的流体所处的位置，然后把那里的速度复制过来。

$$\mathbf{u}_{t+\Delta t}(\mathbf{x}) = \mathbf{u}_t(\mathbf{x} - \mathbf{u}_t(x)\Delta t) \tag{6}$$

由于 $\mathbf{x} - \mathbf{u}_t(x)$ 不一定正好落在整点上，需要进行插值。我们约定采用**双线性插值** (Bilinear Interpolation)

3.2.1 双线性插值

TODO: 这里差张图，喵喵要学 TikZ

3.3 Diffusion

同样为了保证算法的稳定性，采用隐式方法：

$$(\mathbf{I} - \nu\Delta t\nabla^2)\mathbf{u}_{t+\Delta t}(x) = \mathbf{u}_t(x) \tag{7}$$

由于我们将 ∇^2 表示成了矩阵，因此上面这个方程其实是一个线性方程组。

助教在 `utils.py` 中提供了一个求解以下形式线性方程组的函数：

$$(r\mathbf{I} + s\nabla^2)\mathbf{x} = \mathbf{b} \tag{8}$$

其中 $\mathbf{x}$ 和 $\mathbf{b}$ 是二维标量场。

使用方法是调用 `utils.build_poisson_solver(height, width, r, s, factor)`, 其中 `factor` 的含义和 `set_boundary` 中相同。上述调用会返回一个函数 `f`，当你需要求解的时候，你需要将场 $\mathbf{b}$ 形状变为 **长 `width * height` 的一维向量**，传入 `f`，f 会返回一个 **长 `width * height` 的一维向量**，代表解得的 $\mathbf{x}$，你需要将其变回原来的形状。

构造函数 `f` 非常耗时。可以注意到，ν 是整个模拟中都不会改变的参数，因此上述线性方程组的等式左侧其实永远不变。因此可以在模拟开始前调用一次 `utils.build_poisson_solver`，把返回的函数保存起来，之后重复使用即可。

3.4 Projection

在 2.1 一节中提到了去除速度场的散度的方法，需要使用你之前实现好的梯度和散度算子。

注意! 在使用算子前，请一定要使用 `set_boundary` 设置好场的边界值。

此外，`utils.build_poisson_solver` 也可以用来求解 $\mathbf{q}$，返回的函数也同样可以重复使用。