第十七章　物理模拟

我们生活在一个物理世界当中，万物的运动都遵循着相应的物理原理．物理现象在我们身边随处可见，它们多种多样，包括刚体现象、软体现象、流体现象、薄壳现象、磁现象等等．在物理学科漫长的发展史上，人类已经探索出大多数现象的物理方程，这些方程精确描述了物体的运动，为我们解释了自然界中各种现象的规律．

然而，仅仅拥有物理方程不足以让我们完全理解这个世界，因为这些方程往往过于复杂而难以求解，甚至不存在一种数学表达形式来描述它的解；1929 年，英国著名量子物理学家保罗・狄拉克曾说过，“大部分物理学和整个化学的数学理论所需的基本物理定律是完全已知的，困难只是这些定律的确切应用导致方程太复杂而无法解决”．[1]

正是因为绝大多数物理方程难以精确求解，人们开始退而求其次，选择求解一个数值近似解来大致地理解某一种现象的运动规律．并且，我们往往还要求这个数值近似解能够随着计算量的增大尽可能逼近真实解．这样，我们就可以将不可解的问题转化为计算量巨大的近似问题，并交给计算机去做了．于是，物理模拟就这样诞生了．

除了辅助研究物体的运动规律，物理模拟还有着丰富的应用场景．在工程实验上，我们可以借助计算机模拟飞行器的风洞试验、材料的抗压测试等；在机器人学与具身智能领域，人们往往会搭建一个物理模拟环境用来测试一个机器人控制算法，从而避免实机测试所造成的损坏；我们还可以根据气象数据对天气或气候进行建模并模拟，以做出较准确的天气预报……在图形学中，物理模拟能够帮助我们生成电影、游戏甚至元宇宙中更加真实的动画效果．为追求更真实的现象、更高的计算效率以及更复杂现象的模拟，已有大量的研究成果被发表．图17.1展示了一些工作对各种现象的模拟．

接下来,我们将带领读者模拟两种物理现象:弹簧质点 (第17.1节) 和流体现象 (第17.2节)．对这两种现象我们会分别介绍一种最基础的模拟方法，以便读者了解物理模拟算法的基本组成与思想．

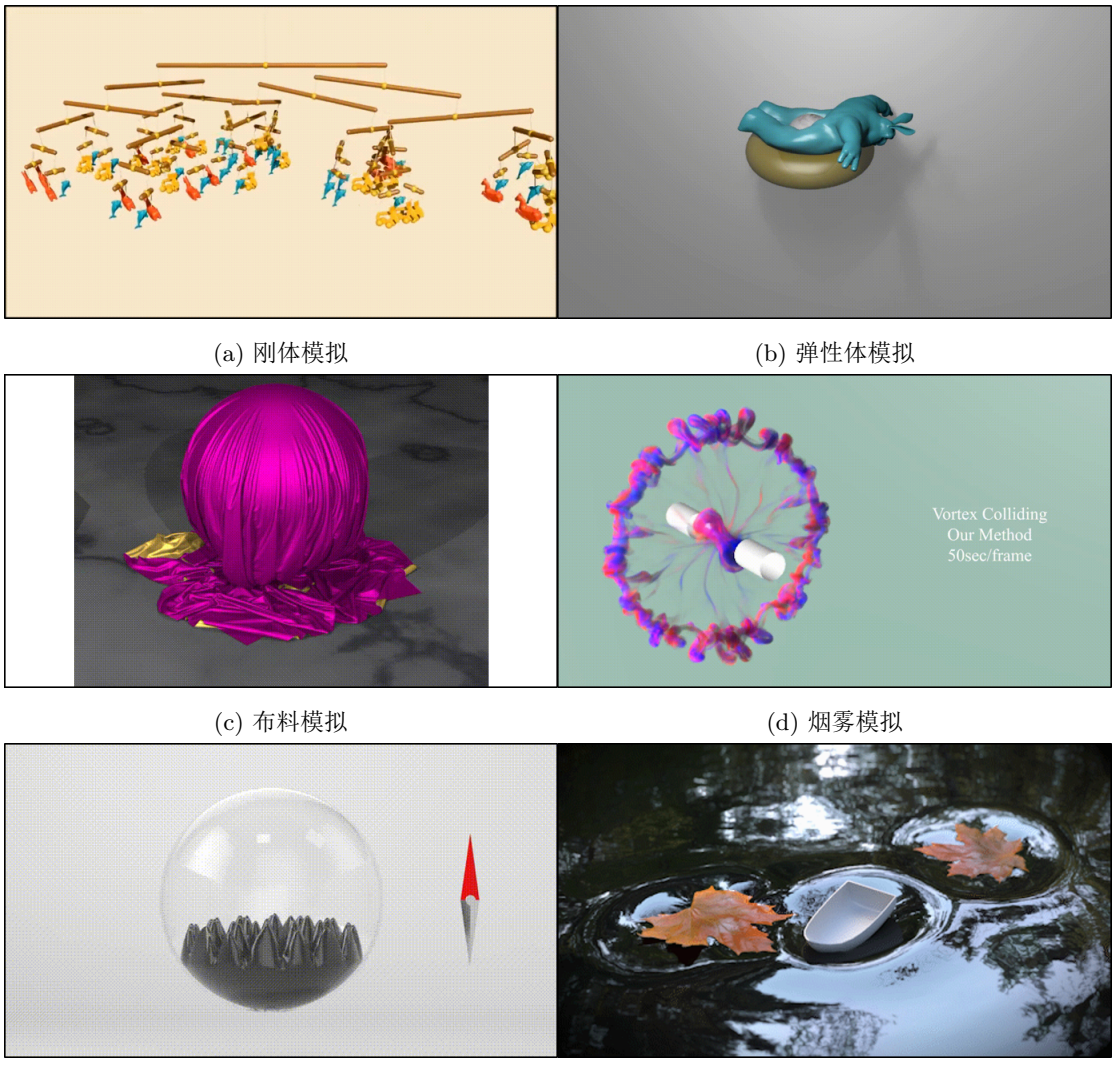
17.1　弹簧质点系统

17.1.1　物理模型

在模拟一个物理世界之前，我们首先需要理解它的物理原理，而在仿真当中，我们最关心的就是物体的运动情况．对于日常生活中宏观低速的物体，其运动规律就是我们所熟知的牛顿第二定律——力是物体改变运动状态的原因．对于一个质量为 m 的质点，假设其位置向量 $\mathbf{x}$，受力 $\mathbf{f}$，则它的运动学方程为：

$$\mathbf{f} = m\ddot{\mathbf{x}} \tag{17.1}$$

[1] https://zhuanlan.zhihu.com/p/407366845



(a) 刚体模拟　(b) 弹性体模拟

(c) 布料模拟　(d) 烟雾模拟

(e) 磁流体模拟　(f) 流体表面现象模拟

图 17.1: 各种物理现象的模拟

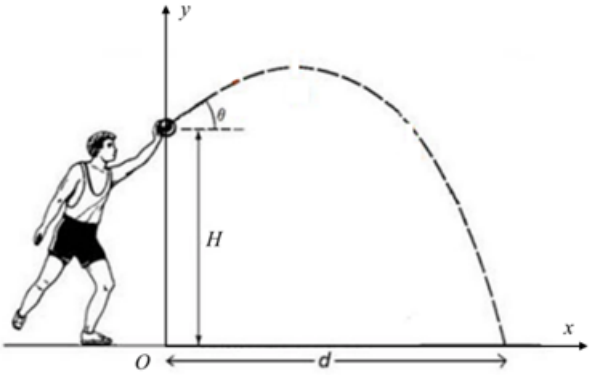


图 17.2: 单一粒子运动的例子

这里，我们常用 $\dot{\mathbf{x}}$ 和 $\ddot{\mathbf{x}}$ 表示位置对时间求一阶导和二阶导得到的梯度向量 (有时也分别记为 $\mathbf{v}$ 和 $\mathbf{a}$)，即速度和加速度．我们关心的物体的受力 $\mathbf{f}$ 可以是这个世界上具有的任何力，包括重力、弹性力、浮力、表面张力、磁力等等．

我们考虑一个最简单的例子：粒子以一定初速度被抛出，在重力作用下运动直至落地．由于粒子的运动轨迹一定在一个平面内，我们设 x 轴正方向为粒子的水平运动方向，y 轴正方向为竖直朝上 (如图17.2所示)，原点在粒子初始位置的正下方；另外设粒子初始高度为 H，初始速度大小为 v_0，方向与 x 轴夹角为 θ．这个粒子的运动十分简单，它只受到一个向下重力，即

$$\mathbf{f} = \begin{bmatrix} 0 \\ -mg \end{bmatrix} \tag{17.2}$$

其中 m 为粒子的质量．根据式17.1可得粒子的加速度

$$\ddot{\mathbf{x}} = \begin{bmatrix} 0 \\ -g \end{bmatrix} \tag{17.3}$$

粒子的速度可由初速度加上加速度的时间积分得到：

$$\dot{\mathbf{x}} = \begin{bmatrix} v_0 \cos\theta \\ v_0 \sin\theta \end{bmatrix} + \int_0^t \ddot{\mathbf{x}} \mathrm{d}\tau = \begin{bmatrix} v_0 \cos\theta \\ v_0 \sin\theta - gt \end{bmatrix} \tag{17.4}$$

粒子的位置同样可以由初始位置加上速度的时间积分得到：

$$\mathbf{x} = \begin{bmatrix} 0 \\ H \end{bmatrix} + \int_0^t \dot{\mathbf{x}} \mathrm{d}\tau = \begin{bmatrix} (v_0 \cos\theta)t \\ H + (v_0 \sin\theta)t - \frac{1}{2}gt^2 \end{bmatrix} \tag{17.5}$$

17.1.2　物理模拟

对上一节的例子，我们已经完全了解了粒子的全部运动，但是这个例子实在过于简单——整个场景中只有**一个粒子**，并且只受**一个力**——以至于它难以代表一般性的情况．在一般的物理模拟场景中，一块连续的材料 (如一个弹性体、一单位体积的流体或是一片布料) 包含**无穷多粒子**，并且往往会存在**多个外力**以及粒子之间的**十分复杂的内力**．这个时候，系统往往不再存在解析解，所以我们就退而求其次，寻求一个尽可能接近物理真实情况的数值解．**物理模拟 (或物理仿真)** 就是寻求数值解的过程．

为了进行物理模拟，我们首先要做的是将一个在空间上连续的物体进行空间离散化，设计适当的数据结构以存储它的拓扑结构及运动状态．然后根据它所遵循的运动规律 (运动学方程)，在给定初始状态的前提下，对它的运动进行步进求解．这个过程大致如图17.3所示，我们将依次介绍它的三个组成，即空间离散化、时间离散化、数值求解．

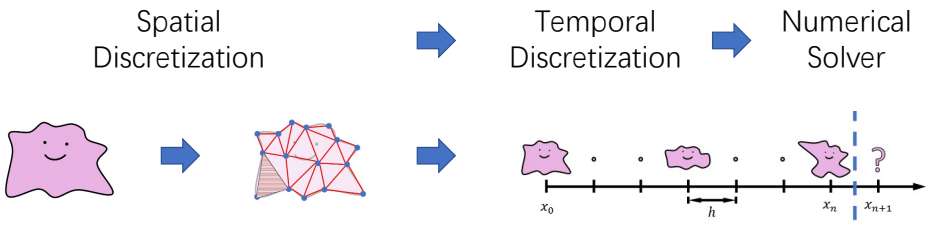


图 17.3: 物理仿真的过程

17.1.3　空间离散化

质点具有质量，会因力的作用改变运动状态，但没有空间体积和形状，是物理学中的一种理想化模型．单个质点的状态是由它的位置和速度确定的．而为了在已知状态的基础上模拟出质点的运动过程，我们还需要通过受力和质量计算出它的加速度．进一步地，一个由多个质点组成的质点系统，其数据结构可以简单地设计成一个质点的列表，如图17.4所示．

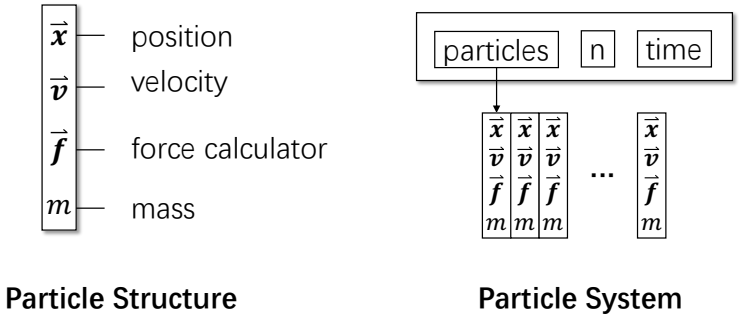


图 17.4: 质点系统的组成[2]

另外，弹簧质点系统的能量由忽略质量的弹簧提供，为了简便，我们可以额外假设弹簧无阻尼，即弹簧的弹力和弹性势能只是两端质点位置的函数，与质点速度和时间无关．一个三维弹簧的弹性势能，具有以下的特点：

1. 撤去所有外力后弹簧回到原长，即原长对应单根弹簧弹性势能的最小值点；
2. 刚性运动 (平移、旋转) 不改变弹性势能；
3. 弹性势能只依赖于所连接质点的位置 (对于无阻尼弹簧)．

对于一根连接质点 i, j 的弹簧，设其原长为 l_{ij}，劲度系数为 k_{ij}，前面已经提到两质点的位置分别为 $\mathbf{x}_i, \mathbf{x}_j$，取其弹性势能为：

$$E_{ij} = \frac{1}{2} k_{ij} \left(\|\mathbf{x}_j - \mathbf{x}_i\| - l_{ij} \right)^2 \tag{17.6}$$

[2]Online Siggraph '97 Course notes

对 E_{ij} 关于 $\mathbf{x}_i$ 求负梯度可得 i 受该弹簧的弹力 $\mathbf{f}_{ij}$，同理关于 $\mathbf{x}_j$ 求负梯度可得 $\mathbf{f}_{ji}$，$\mathbf{f}_{ij}, \mathbf{f}_{ji}$ 的表达式如下：

$$\begin{aligned}\mathbf{f}_{ij} &= -\nabla_i E_{ij} = k_{ij}\left(\|\mathbf{x}_j - \mathbf{x}_i\| - l_{ij}\right)\frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} = k_{ij}\left(\|\mathbf{x}_j - \mathbf{x}_i\| - l_{ij}\right)\mathbf{n}_{ij} \\ \mathbf{f}_{ji} &= -\nabla_j E_{ij} = -\mathbf{f}_{ij}\end{aligned} \tag{17.7}$$

其中 $\mathbf{n}_{ij}$ 为从质点 i 指向 j 方向的单位向量.

由此我们可以得到一个质点 i 所受的合力：

$$\mathbf{f}_i = \sum_{j \in N(i)} \mathbf{f}_{ij} + \mathbf{f}_i^{\text{ext}} \tag{17.8}$$

其中 $N(i)$ 表示所有和 i 之间有弹簧连接的质点的集合；$\mathbf{f}_i^{\text{ext}}$ 表示质点 i 所受的外力，如重力，外力与位置无关，一般可以提前计算出来，我们在后面就会看到为什么要将外力写成额外的一项.

17.1.4　时间离散化

物体运动状态随时间连续变化，因此在时间维度上同样需要进行离散化．这意味着对式17.1的微分方程进行离散化．一般来说，我们在时间区间上均匀采样，物理动画的帧率即由采样的时间间隔 h (通常也称为时间步长，单位 s，帧率即为时间步长的倒数) 决定．当已知当前物体的状态 (如质点系统中每个质点的位置和速度)，可以计算出相应的受力情况，因此可以通过时间积分求出下一采样时刻的物体状态．我们记 t 时刻的位置和速度向量为 $\mathbf{x}(t), \mathbf{v}(t)$，并简记第 k 次采样时刻 t_k 的位置与速度向量为 $\mathbf{x}^k = \mathbf{x}(t_k), \mathbf{v}^k = \mathbf{v}(t_k)$，对于一个 n 个质点组成的质点系统而言，分别为 n 个质点的位置和速度的堆叠向量，$\mathbf{x}^k, \mathbf{v}^k \in \mathbb{R}^{3n}$，即：

$$\begin{aligned}\mathbf{x}^k &= \begin{pmatrix}\mathbf{x}_1(t_k) \\ \mathbf{x}_2(t_k) \\ \vdots \\ \mathbf{x}_n(t_k)\end{pmatrix} \\ \dot{\mathbf{x}}^k = \mathbf{v}^k &= \begin{pmatrix}\mathbf{v}_1(t_k) \\ \mathbf{v}_2(t_k) \\ \vdots \\ \mathbf{v}_n(t_k)\end{pmatrix}\end{aligned} \tag{17.9}$$

递进一个时间步意味着物体状态从 t_k 时刻到 t_{k+1} 时刻的更新，具体来说，就是基于牛顿第二定律的离散化表达，计算出下一时刻物体的状态：

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{x}^k + \int_{t_k}^{t_{k+1}} \mathbf{v}(t)\,\mathrm{d}t \\ \mathbf{v}^{k+1} &= \mathbf{v}^k + \mathbf{M}^{-1}\int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{x}(t), \mathbf{v}(t))\,\mathrm{d}t\end{aligned} \tag{17.10}$$

其中，式17.1中的标量质量 m 需要处理成质量矩阵 $\mathbf{M}$，对于质点系统来说，可以取为对角矩阵，这也便于并行求解.

因此，给定系统的初始位置 $\mathbf{x}_0$ 和初始速度 $\mathbf{v}_0$，通过上式循环迭代，理论上就可以依次求出后续每个采样时刻的系统状态，从而得到一段物理动画．故剩下的任务是计算式17.10中的时间积分，接下来介绍两种典型的计算方法：**显式欧拉积分 (Explicit/Forward Euler)** 和**隐式欧拉积分 (Implicit/Backward Euler)**.

显式欧拉积分

在较为复杂的场景中，式17.10中的积分项可能不存在解析表达，好在积分区间 $[t_k, t_{k+1}]$ 比较小，所以我们可以用简单的形式近似这个积分的值．最简单的方式就是用每个时间步刚开始的值去近似这个时间步内任意时刻的值，于是式17.10转化成：

$$\mathbf{x}^{k+1} = \mathbf{x}^k + h\mathbf{v}^k \tag{17.11}$$

$$\mathbf{v}^{k+1} = \mathbf{v}^k + h\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^k) \tag{17.12}$$

这就是显示欧拉积分，只需要根据已知的位置和速度状态就可以直接计算出下一时刻的状态，计算过程十分简便．每个时间步内的计算步骤如下所示：

- 利用当前时刻的位置 $\mathbf{x}^k$ 使用式17.7计算每个弹簧的力，然后用式17.8计算每个质点的受力；
- 利用当前时刻速度 $\mathbf{v}^k$ 使用式17.11更新位置；
- 利用前面算好的每个质点的受力使用式17.12更新速度．

但是，显示时间积分有稳定性差的缺点．我们不妨想象一个质点被一根弹簧吊在天花板上的情形，如图17.5最左一列所示，假设质点初始时有一个向下的速度，质点在重力和弹簧拉力的作用下在水平虚线处达到平衡．若时间步很大，那么由于质点具有一定初速度，在第一个时间步过后质点就会越过平衡位置并且超出很多；那么在下一个时间步开始时弹簧的拉力会大于重力，导致质点的速度变成向上，且速度大小比初始速度更快，于是经过第二个时间步之后粒子会冲得更高，以此类推．我们可以很明显注意到这个过程中整个系统的能量大大地增加了，所以会导致这个粒子运动速度越来越快、运动幅度越来越大，直至冲出屏幕，或是超出浮点运算范围．

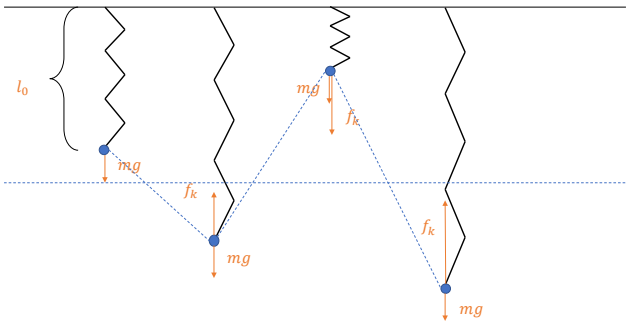


图 17.5: 在大时间步下显示时间积分炸掉的例子

当然，解决不稳定的方法是存在的，一个最简单的方法就是减小时间步长．图17.6展示了在利用显式欧拉积分解微分方程 $\dot{x} = -kx$ 时，不同时间步长对结果的影响．可以看到在左侧时间步长很小的情况下能够稳定模拟，但是随着时间步长的加大，结果会变得不准确，甚至炸掉．减小时间步是一个很有效的解决方法，但是这会大大增加需要模拟的时间步的数量；并且如果模拟的场景中物体速度越快，时间步长就需要越小才能够稳定，这会极大增加运算量．想要真正解决稳定性问题，我们还需借助另外一种时间积分格式．

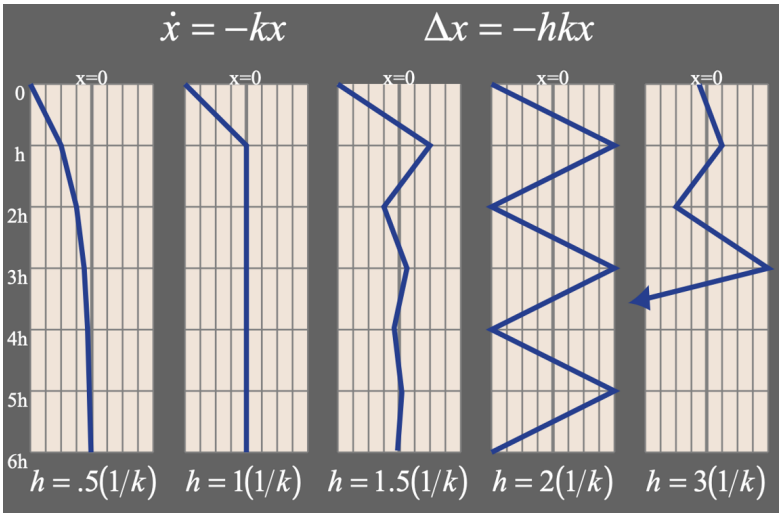


图 17.6: 显式欧拉积分在不同时间步大小情况下的表现．微分方程 $\dot{x} = -kx$ 存在解析解 $x(t) = ce^{-kt}$，其中 c 为任意常数，带入初始条件 $x(0) = -1$ 可得 $c = -1$．在时间步长足够小的时候 (最左侧)，可以模拟出 x 随 t 指数衰减至 0 的趋势，但随着时间步长的增大，模拟结果依次变得不准确、出现震荡、不收敛、指数级发散．

隐式欧拉积分

在隐式欧拉积分中，我们将式17.10转化成：

$$\mathbf{x}^{k+1} = \mathbf{x}^k + h\mathbf{v}^{k+1} \tag{17.13}$$

$$\mathbf{v}^{k+1} = \mathbf{v}^k + h\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^{k+1}) \tag{17.14}$$

可以看出，它和显式欧拉积分的区别在于，隐式欧拉法选取 t_{k+1} 时刻 (而非 t_k 时刻) 的位置和速度来近似整个时间步内的值．

隐式欧拉法中我们无法直接计算 $\mathbf{x}^{k+1}$ 和 $\mathbf{v}^{k+1}$，而是需要求解方程组．将式17.14代入到式17.13中，得到：

$$\mathbf{x}^{k+1} = \mathbf{x}^k + h\mathbf{v}^k + h^2\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^{k+1}) \tag{17.15}$$

然后我们将 $\mathbf{f}(\mathbf{x}^{k+1})$ 分成内力和外力两部分：

$$\mathbf{f}(\mathbf{x}^{k+1}) = \mathbf{f}_{\text{int}}(\mathbf{x}^{k+1}) + \mathbf{f}_{\text{ext}} \tag{17.16}$$

前面已经提到过，$\mathbf{f}_{\text{ext}}$ 与位置无关，可以视为已知量，那么式17.15变为：

$$\mathbf{x}^{k+1} = (\mathbf{x}^k + h\mathbf{v}^k + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}) + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{int}}(\mathbf{x}^{k+1}) \tag{17.17}$$

等号右边前半部分都是已知量，记 $\mathbf{y}^k = \mathbf{x}^k + h\mathbf{v}^k + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$，那么我们要求解的是如下所示的一个关于 $\mathbf{x}^{k+1}$ 的方程组：

$$\mathbf{x}^{k+1} - \mathbf{y}^k - h^2\mathbf{M}^{-1}\mathbf{f}_{\text{int}}(\mathbf{x}^{k+1}) = \mathbf{x}^{k+1} - \mathbf{y}^k + h^2\mathbf{M}^{-1}\frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^{k+1}) = 0 \tag{17.18}$$

它等价于求解如下的优化问题 (读者可以利用目标函数关于 $\mathbf{x}$ 的梯度等于 $\mathbf{0}$ 证明等价性)：

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \frac{1}{2h^2}\left\|\mathbf{x} - \mathbf{y}^k\right\|_{\mathbf{M}}^2 + E(\mathbf{x}) \tag{17.19}$$

其中 $\|\mathbf{r}\|_{\mathbf{M}}^2 = \mathbf{r}^\top \mathbf{M}\mathbf{r}$. 这就是说，隐式欧拉积分等价于求解一个能量最小化问题，这个能量包含惯性项 (inertia) $\frac{1}{2h^2}\left\|\mathbf{x}-\mathbf{y}^k\right\|_{\mathbf{M}}^2$ 和弹性项 (elasticity) $E(\mathbf{x})$ (还记得式17.6定义的弹性势能吗？$E(\mathbf{x})$ 就定义为每个弹簧势能之和 $E(\mathbf{x}) = \sum_{(i,j)} E_{ij}$)，这也就是为什么我们说隐式欧拉法可以解决稳定性问题——它可以在任意长的时间步下稳定，因为它时刻保证系统的能量最小化.

所以在很多情况下，我们会采用隐式欧拉法. 隐式欧拉法的单步计算代价较高，但是允许更大的时间步长，这反而提高了仿真的效率，并且极大地改善了系统的稳定性. 二者对比的一个实例可参见图17.7.

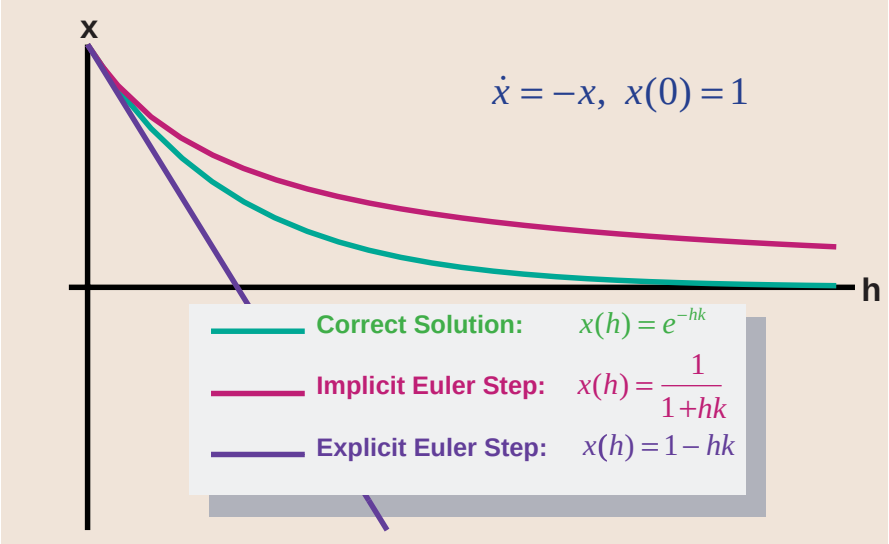


图 17.7: 显式欧拉积分与隐式欧拉积分的单步对比. 对于微分方程 $\dot{x} = -kx$，考察分别按照显式欧拉法和隐式欧拉法递进时间步长为 h 的一步后与准确解之间的差距. 以 $x(0) = 1$ 为初始状态，取系数 $k = 1$，可作出图像. 可以看到，当时间步长较大，显式欧拉法的误差将快速增长并因此容易导致仿真崩溃，而隐式欧拉法则允许更大的时间步长，因此具有较好的稳定性. [3]

17.1.5　数值求解

接下来我们尝试对隐式欧拉积分中的方程组进行求解，设需要最小化的目标能量函数为 $g(\mathbf{x}) = \frac{1}{2h^2}\left\|\mathbf{x}-\mathbf{y}^k\right\|_{\mathbf{M}}^2 + E(\mathbf{x})$. 一个经典方法是牛顿法，它是一个基于迭代的优化算法，其思想是每轮迭代都用二次函数逼近目标函数，并在该轮迭代中找到二次函数的最小值作为新的尝试解. 迭代算法会依次求解出一个尝试序列 $\{\mathbf{x}_i\}_{i=0}^{m-1}$，其中 $\mathbf{x}_i$ 表示第 i 轮迭代的尝试解，每一轮迭代算法会从 $\mathbf{x}_i$ 计算下一轮 $\mathbf{x}_{i+1}$，一般来讲序列 $\{\mathbf{x}_i\}$ 会收敛到一个稳定解，最后我们把这个稳定解作为全局最小值点的近似 (当然对于很复杂的优化问题，我们可能只能找到一个局部极小值点，甚至无法让算法收敛). 如何选取一个最好的二次函数呢？我们可以对目标函数在当前尝试解 $\mathbf{x}_i$ 处进行二阶泰勒展开：

$$g(\mathbf{x}) = g(\mathbf{x}_i) + \nabla g(\mathbf{x}_i)\cdot(\mathbf{x}-\mathbf{x}_i) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_i)^\top \mathbf{H}_g(\mathbf{x}_i)(\mathbf{x}-\mathbf{x}_i) + O(\|\mathbf{x}-\mathbf{x}_i\|^3) \tag{17.20}$$

[3] Online Siggraph '97 Course notes

这里我们把梯度写成列向量 $\nabla g(\mathbf{x}_i) = \begin{bmatrix} \frac{\partial g(\mathbf{x}_i)}{\partial x} \\ \frac{\partial g(\mathbf{x}_i)}{\partial y} \\ \frac{\partial g(\mathbf{x}_i)}{\partial z} \end{bmatrix}$，$\mathbf{H}_g(\mathbf{x}_i) = \begin{bmatrix} \frac{\partial^2 g(\mathbf{x}_i)}{\partial x^2} & \frac{\partial^2 g(\mathbf{x}_i)}{\partial x \partial y} & \frac{\partial^2 g(\mathbf{x}_i)}{\partial x \partial z} \\ \frac{\partial^2 g(\mathbf{x}_i)}{\partial x \partial y} & \frac{\partial^2 g(\mathbf{x}_i)}{\partial y^2} & \frac{\partial^2 g(\mathbf{x}_i)}{\partial y \partial z} \\ \frac{\partial^2 g(\mathbf{x}_i)}{\partial x \partial z} & \frac{\partial^2 g(\mathbf{x}_i)}{\partial y \partial z} & \frac{\partial^2 g(\mathbf{x}_i)}{\partial z^2} \end{bmatrix}$ 是 g 的海瑟矩阵 (Hessian Matrix)．我们忽略三阶小量 $O(\|\mathbf{x} - \mathbf{x}_i\|^3)$，就得到了一个用于近似 $g(\mathbf{x})$ 的二次函数 (请注意，这个近似仅仅是对 $g(\mathbf{x})$ 在 $\mathbf{x}_i$ 附近的近似，当 $\mathbf{x}$ 太远的时候三阶“小”量将不能忽略！)，下一轮迭代的尝试解 $\mathbf{x}_{i+1}$ 取这个二次函数的极小值，通过对式17.20两边求梯度，再代入 $\mathbf{x}_{i+1}$ 可得

$$\nabla g(\mathbf{x}_{i+1}) \approx \nabla g(\mathbf{x}_i) + \mathbf{H}_g(\mathbf{x}_i)(\mathbf{x}_{i+1} - \mathbf{x}_i) = \mathbf{0} \tag{17.21}$$

依此求得下一轮的尝试解：

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{H}_g^{-1}(\mathbf{x}_i)\nabla g(\mathbf{x}_i) \tag{17.22}$$

牛顿法相比于普通的梯度下降法 (Gradient Descent) 能做到更快的收敛．而它的缺点在于，每步迭代中需要求解 $\mathbf{H}_g$ 矩阵的逆，计算代价较大，且一般要求 $\mathbf{H}_g$ 为正定矩阵，否则方程可能无解 (牛顿法一般用于解凸优化问题) 或解不出正确下降方向．

在此基础上还有拟牛顿法 (Quasi-Newton method)、BFGS 拟牛顿法等．为了避免海瑟矩阵的计算和存储，共轭梯度法 (Conjugated Gradient) 也是常用的优化方法之一．更多的优化算法这里不再赘述．

对于本节中的弹簧质点系统，$g(\mathbf{x})$ 性质足够好，我们认为只需要进行一步牛顿迭代即可求得最小值点 (这里是一个简化的假设，事实上 $g(\mathbf{x})$ 不是凸函数)．我们只需要将式17.22中的 $\mathbf{x}_i$ 和 $\mathbf{x}_{i+1}$ 分别替换成 $\mathbf{x}^k$ 和 $\mathbf{x}^{k+1}$ 即可．所以我们需要解如下关于 $\mathbf{x}^{k+1} - \mathbf{x}^k$ 的方程组，随后即可计算出 $\mathbf{x}^{k+1}$ 和 $\mathbf{v}^{k+1}$：

$$\mathbf{H}_g(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) = -\nabla g(\mathbf{x}^k) \tag{17.23}$$

现在我们来计算 $\nabla g(\mathbf{x}^k)$ 和 $\mathbf{H}_g(\mathbf{x}^k)$，由 $g(\mathbf{x})$ 的定义得

$$\begin{aligned} \nabla g(\mathbf{x}^k) &= \frac{1}{h^2}\mathbf{M}(\mathbf{x}^k - \mathbf{y}^k) + \nabla E(\mathbf{x}^k) \\ \mathbf{H}_g(\mathbf{x}^k) &= \frac{1}{h^2}\mathbf{M} + \mathbf{H}(\mathbf{x}^k) \end{aligned} \tag{17.24}$$

其中 $\mathbf{H}(\mathbf{x}^k)$ 是弹性势能 $E(\mathbf{x})$ 的海瑟矩阵．我们在式17.7中已经写出了对于一根弹簧的能量 E_{ij} 关于一个质点 i 的梯度 $\nabla_i E_{ij}(\mathbf{x}^k)$ 的表达式，那么总能量对质点 i 的梯度

$$\nabla_i E(\mathbf{x}^k) = \sum_j \nabla_i E_{ij}(\mathbf{x}^k) \tag{17.25}$$

总能量关于所有质点位置的梯度就是将关于每个质点的梯度拼接起来 (假设总共有 n 个质点)：

$$\nabla E(\mathbf{x}^k) = \begin{bmatrix} \nabla_1 E(\mathbf{x}^k) \\ \vdots \\ \nabla_n E(\mathbf{x}^k) \end{bmatrix} \tag{17.26}$$

同样地，我们考虑一个弹簧 (i,j) 关于质点 i 的海瑟矩阵，即对式17.7两边求梯度，得到

$$\begin{aligned}\mathbf{H}_e := \frac{\partial^2 E_{ij}(\mathbf{x}^k)}{\partial \mathbf{x}_i^2} &= k_{ij}\frac{(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^\top}{\|\mathbf{x}_i-\mathbf{x}_j\|^2} + k_{ij}\left(1-\frac{l_{ij}}{\|\mathbf{x}_i-\mathbf{x}_j\|}\right)\left(\mathbf{I}-\frac{(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^\top}{\|\mathbf{x}_i-\mathbf{x}_j\|^2}\right)\\ \frac{\partial^2 E_{ij}(\mathbf{x}^k)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} &= -\mathbf{H}_e\\ \frac{\partial^2 E_{ij}(\mathbf{x}^k)}{\partial \mathbf{x}_j^2} &= \mathbf{H}_e\end{aligned} \tag{17.27}$$

那么这个弹簧关于所有质点坐标的海瑟矩阵可以写成如下形式：

$$\mathbf{H}_{ij}(\mathbf{x}^k) = \begin{bmatrix} & \vdots & & \vdots & \\ \cdots & \mathbf{H}_e & \cdots & -\mathbf{H}_e & \cdots \\ & \vdots & & \vdots & \\ \cdots & -\mathbf{H}_e & \cdots & \mathbf{H}_e & \cdots \\ & \vdots & & \vdots & \end{bmatrix} \tag{17.28}$$

将 $\mathbf{H}_{ij}(\mathbf{x}^k)$ 划分成 $n\times n$ 个 3×3 的块，则第 i 行第 i 列与第 j 行第 j 列的块为 $\mathbf{H}_e$，而第 i 行第 j 列与第 j 行第 i 列的块为 $-\mathbf{H}_e$，其余块均为零矩阵．总能量的海瑟矩阵即为所有弹簧海瑟矩阵之和：

$$\mathbf{H}(\mathbf{x}^k) = \sum_{(i,j)} \mathbf{H}_{ij}(\mathbf{x}^k) \tag{17.29}$$

最后，我们将式17.26,17.29代入到式17.24中计算 $\nabla g(\mathbf{x}^k)$ 和 $\mathbf{H}_g(\mathbf{x}^k)$，然后即可求解方程17.23.

17.1.6　更多弹性体的离散格式

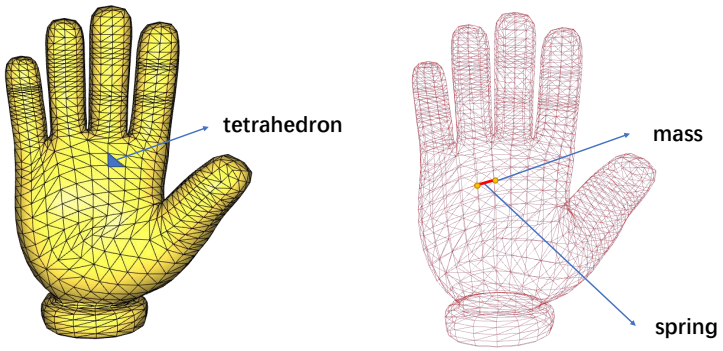


图 17.8: 三维弹性体的四面体元表示及弹簧质点表示

弹簧质点系统具有广泛而重要的应用，如头发、布料、软体的模拟，但这并不是唯一的模拟弹性体的离散格式．如图17.8中所示，对于连续介质固体，我们可以用紧密的四面体元 (左) 或者以弹簧相连的质点系统 (右)，在这两种格式中我们分别将相关信息存储在体元和质点上．

另一种比弹簧质点更加符合物理真实情况的离散格式叫做有限元法 (Finite Element Method, FEM)，它的空间离散形式就是将一个连续弹性体分成许多小体积元 (例如，将一维物体分成许多线段，二维物体分成三角形或四边形网格，三维物体分成四面体或立方体网格)．相对于弹簧质点系统中利用弹簧的长度变化来表示形变，FEM 方法使用形变梯度 (deformation gradient) 来刻画物体的形变，其弹性势能则是关于形变梯度的一个函数：

$$\Psi(\mathbf{F}) = \mu\|\mathbf{F}\|_F^2 + \frac{\lambda}{2}\operatorname{Tr}^2(\mathbf{F}) \tag{17.30}$$

其中 $\mathbf{F}$ 即为形变梯度，是一个二阶张量，可以视为一个矩阵，$\|\cdot\|_F$ 表示矩阵的 Frobenius 范数，$\operatorname{Tr}(\cdot)$ 表示矩阵的迹．而牛顿第二定律在 FEM 方法下体现为下式：

$$\rho\ddot{\mathbf{x}} = \nabla\sigma + \mathbf{f}_{\text{ext}} \tag{17.31}$$

其中 ρ 为弹性体的密度，此处的弹性力由应力张量 (stress tensor) σ 算得．

17.2　流体模拟

接下来我们要进入到另外一个领域——流体模拟．在视觉上，它和弹簧质点系统有很大的差别，但同样作为物理模拟，我们仍然可以从它的物理模型、时空离散化、数值求解这几个角度入手．

17.2.1　物理模型

在了解流体的动力学方程之前，我们首先需要知道如何刻画一个正在运动的流体系统．我们可以借助“场”的概念，用速度场 $\mathbf{v}(\mathbf{x})$ 来表示流体的速度在空间中的分布——事实上这样就足够了，如果我们知道每个时刻的速度场 $\mathbf{v}(\mathbf{x},t)$，以及初始时流体的分布情况，就可以唯一地还原出流体的完整运动情况，因为对于初始时流体中的任何一点，都可以借助 $\mathbf{v}(\mathbf{x},t)$ 唯一还原出它随流体运动的完整轨迹．但是为了方便计算流体运动状态的改变，我们还需要表示出流体的其他物理量，具体来讲，用标量场 $\rho(\mathbf{x},t)$ 表示流体的密度分布，用标量场 $p(\mathbf{x},t)$ 表示流体的压强分布．在后续的式子中，我们将省略场的位置和时间参数 $(\mathbf{x},t)$．

NS 方程

与弹簧质点系统一样，流体所遵循的最基本的运动学方程仍然是牛顿第二定律 $\mathbf{f}=m\ddot{\mathbf{x}}$，只是流体内部的力要更为复杂一些，所以方程会变得稍微复杂一点：

$$\rho\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t} = -\nabla p + \rho\mathbf{g} + \mu\nabla^2\mathbf{v} \tag{17.32}$$

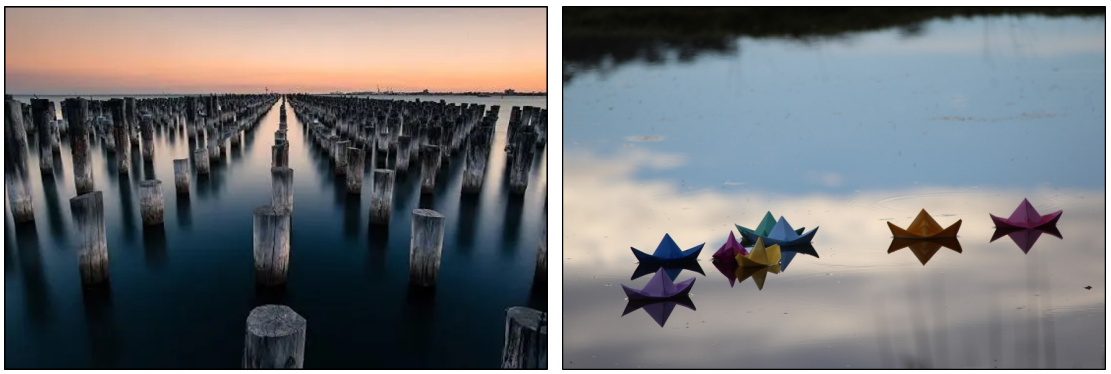
这就是著名的 NS 方程 (Navier-Stokes Equations)．

为了看懂这个方程，我们首先取某一时刻流体中的一个点，由于体积无穷小，所以质量也是无穷小的，那么现在有意义的物理量就是密度 ρ (即单位体积内的质量)，它取代了牛顿第二定律中的 m；$\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t}$ 是速度的随体导数 (material derivative)，它的含义是，假设我们在流体中选取的点是随着流体一起运动的，那么这个点在当前时刻的速度的导数——这完美地对应了牛顿第二定律中的加速度．总而言之，NS 方程等号左边的部分 $\rho\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t}$ 对应于牛顿第二定律中的 $m\ddot{\mathbf{x}}$，它的意义是“单位体积内流体的质量乘以加速度”．

接下来再依次分析等号右边的每一项．在中学我们就知道压强 p 的定义是“单位面积内受压力大小”，那么 $-\nabla p$ 可以理解为“单位距离上的压强差”，也就是“单位体积内所受

压力”，由于压力是从高压强处指向低压强处，所以要带上负号．$\mathbf{g}$ 是外力 (包括重力等) 提供的加速度，所以 $\rho\mathbf{g}$ 就是单位体积内流体所受的外力．$\mu\nabla^2\mathbf{v}$ 是粘性项，流体的粘性阻止流体产生形变，所以与速度的拉普拉斯成正比，在一般的流体模拟中会直接忽略这一项．将这些加到一起，我们就能够知道等号右边的含义就是“单位体积内流体所受的力”，这个力由压强、重力和粘性提供，对应于牛顿第二定律中的 $\mathbf{f}$.

描述流体的两种视角



(a) 欧拉视角　　(b) 拉格朗日视角

图 17.9: 欧拉视角与拉格朗日视角对比

前面我们遇到的 NS 方程是以**欧拉视角**来描述流体的运动的，欧拉视角是指将流体的所有物理量看成空间上的一个场，然后描述这个场随时间的变化．这好比在水中插有无限多的木桩 (如图17.9a所示)，每个木桩上装有检测水的流速、密度、压强等的传感器，我们描述的是所有传感器上的数值变化．

另一种描述流体的视角叫做**拉格朗日视角**，它将流体看成由无限多个质元组成，流体的运动就是每个质元的运动，我们关注的是所有的质元在每个时刻的位置、速度等物理量．这好比在水上放有无穷多个随波逐流的小船 (如图17.9b所示)，在船上安放检测水各种物理量的传感器，并描述它们的数值变化．

借助这个概念我们可以更加深刻地理解随体导数 $\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t}$ 的含义，它可以看做是随波逐流的小船的加速度，这个导数又称为拉格朗日导数 (Lagrangian derivative)．我们也可以将其用欧拉导数 (Eulerian derivative) 来表示，假设我们关注流体中某一个点的运动，这个点的运动轨迹为 $\mathbf{x}(t)$，则它的速度随时间变化可以表示为函数 $\mathbf{v}(\mathbf{x}(t),t)$，那么随体导数就是对这个函数关于时间求全微分：

$$\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t}=\frac{\mathrm{d}\mathbf{v}(\mathbf{x}(t),t)}{\mathrm{d}t}=\frac{\partial\mathbf{v}}{\partial t}+\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\cdot\nabla\mathbf{v}=\frac{\partial\mathbf{v}}{\partial t}+\mathbf{v}\cdot\nabla\mathbf{v} \tag{17.33}$$

由此可见，欧拉视角和拉格朗日视角下描述的流体动力学方程是等价的，即便 NS 方程是一个欧拉视角下的方程，我们完全可以用拉格朗日视角的方法去模拟．

不可压性质

要想模拟出真实的水的效果，仅靠 NS 方程是不够的，因为水还有一个不可压性质，即任意时刻任意处的体积不变．想要描述这个性质，我们先从物理中的质量守恒方程 (mass conservation equation) 开始：

$$\frac{\partial\rho}{\partial t}=-\nabla\cdot(\rho\mathbf{v}) \tag{17.34}$$

在一般的流体模拟 (包括水、烟雾模拟等) 中，我们都会有一个不可压的假设，即密度恒定，ρ 在各处、任意时刻都为常数．在这样的假设下，式17.34就变成：

$$\nabla \cdot \mathbf{v} = 0 \tag{17.35}$$

即速度场无散．结合高斯公式，我们可以发现，流体速度场无散说明，在流体内部任取一个由有限块光华双侧曲面围成的有界闭区域，则流进该区域的流量和流出的流量相等，即该区域内流体体积守恒．

那么在物理上，流体的不可压性质是如何实现的呢？还记得 NS 方程 (式17.33) 中的压强项 $-\nabla p$ 吗，没错，压强就是用来维持流体的不可压性质的．接下来我们模拟的基本流程就是：先按照速度场进行对流，然后加上外力项，最后我们要计算合适的压强分布使得在压强的作用下速度能重新变成一个 (近似的) 无散场．

17.2.2　空间离散化

对于流体，我们同样可以用粒子系统来进行空间离散化，这样，就是在用拉格朗日视角进行模拟，所以这种离散化方法属于拉格朗日方法．这里我们介绍一种最简单的粒子法——光滑粒子流体动力学 (Smoothed Particles Hydrodynamics, SPH)．

在 SPH 中，我们可以认为每一个粒子代表一小部分流体，但是要注意每个粒子所代表的流体质量是恒定的．我们在每个粒子上存储它的质量 m_i、位置 $\mathbf{x}_i$ 和速度 $\mathbf{v}_i$ (下标表示粒子的编号)，下一步的工作就是估计出流体的密度场．

核函数

在估计密度场之前，我们首先介绍一个概念——核函数 (Kernel Functions)，这个概念贯穿了整个 SPH 方法．不仅仅是密度场，其他许多物理量都需要用核函数来估计．

SPH 的一大优点已经体现在它的名字当中——光滑，之所以称为“光滑粒子”，就是因为采用它的离散格式所近似出来的场是光滑的．如果想要用一群均匀采样的粒子离散一个标量场，最直接的想法便是在每个粒子的位置上采样，对于不在粒子上的点，用离它最近的粒子上的值来近似——这个方法的一个致命缺点就是它不连续，每个粒子都对应于一个区域，区域内任一点到它最近，在这个区域的边界上就会有数值的跳变 (因为“最近粒子”变成了另一个)，在流体模拟中这会带来很不自然的模拟效果．既然每个位置上的取值只依赖于一个粒子会带来不连续性，那么为了“平滑”这种不连续性，我们可以让取值依赖于附近的多个粒子，并且可以很自然地想到，离得越远的粒子权重越低，这样在我们连续移动取值点时，离得远的粒子的进出就不会带来太多数值变化，从而就可以实现连续了．

SPH 的核函数采用的正是这一思想，它一般是一个定义在球形邻域上的非负函数，并且越靠近邻域的边界值越接近 0，如图17.10所示．核函数正是为了给邻居粒子上的值加权，为了保证乘上权重之后不会导致数值被无故地放大或缩小，一般核函数 W 的选取要满足 $\int W(\|\mathbf{x}\|)\,\mathrm{d}\mathbf{x} = 1$．

一个常用的核函数表达式如下 (我们称之为 poly6 核)：

$$W_{\text{poly6}}(r) = \alpha \begin{cases} (h^2 - r^2)^3 & 0 \le r \le h \\ 0 & \text{otherwise} \end{cases} \tag{17.36}$$

[4]维基百科

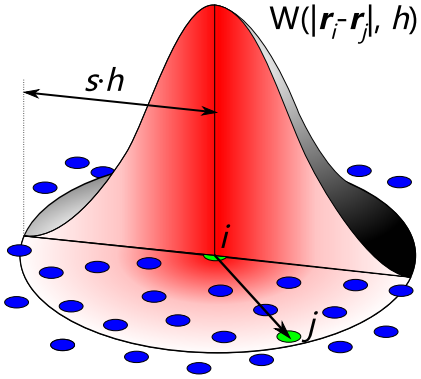


图 17.10: 核函数[4]

其中 α 在二维情况下取 $\frac{4}{\pi h^8}$，三维情况下取 $\frac{316}{64\pi h^9}$；h 称为核半径，在核半径以外的地方核函数取值为 0.

密度场与压强场

有了核函数之后，我们就可以通过粒子的位置和质量来估计出密度场：

$$\rho(\mathbf{x}) = \sum_i m_i W\left(\|\mathbf{x} - \mathbf{x}_i\|\right) \tag{17.37}$$

类似地，我们通过以下式子估算压强场和压强的梯度：

$$p(\mathbf{x}) = \sum_i p_i \frac{m_i}{\rho_i} W\left(\|\mathbf{x} - \mathbf{x}_i\|\right) \tag{17.38}$$

$$\nabla p(\mathbf{x}) = \sum_i p_i \frac{m_i}{\rho_i} \nabla W\left(\|\mathbf{x} - \mathbf{x}_i\|\right) \tag{17.39}$$

其中，

$$p_i = k(\rho_i - \rho_0)^\gamma \tag{17.40}$$

这里 k, γ 均为常数，ρ_0 为流体的静止密度．p_i 表示粒子 i 处的压强大小，它随粒子密度 ρ_i 增加而增加，那么 NS 方程中的压强项 $-\nabla p$ 就会将粒子从密度高的地方推向密度低的地方，以维持一个近似不可压的状态．

注意到式17.40被称为状态方程法 (Equation of State, EOS)，这样计算出来的压强不能够维持一个严格的无散速度场，所以模拟出来的流体实际上是弱可压 (weakly compressible) 流体．当然，我们可以通过调节 k 和 γ 来调整流体有多不可压．

另外，我们还可以用下式替换式17.39来求压强梯度：

$$\nabla p(\mathbf{x}_i) = \rho_i \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \nabla_i W\left(\|\mathbf{x}_j - \mathbf{x}_i\|\right) \tag{17.41}$$

这个式子的好处在于任意两个粒子给对方的压力恰好大小相等、方向相反，从而能够保证整个系统的动量守恒．

17.2.3　SPH 算法流程

总结一下，我们可以得出在流体模拟中，每个时间步需要执行的步骤：

- 用粒子受到的外力更新速度 $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t m_i \mathbf{g}$，然后用速度更新粒子位置 $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$；
- 用式17.37计算每个粒子处的密度 $\rho_i = \rho(\mathbf{x}_i)$；
- 用式17.40计算每个粒子处的压强；
- 用式17.39或式17.41计算压强梯度 $\nabla p(\mathbf{x}_i)$，从而得到每个粒子受到的压力 $\mathbf{f}_i = \frac{m_i}{\rho_i}\nabla p(\mathbf{x}_i)$；
- 用压力更新粒子速度和位置 $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t \mathbf{f}_i, \mathbf{x}_i \leftarrow \mathbf{x}_i + (\Delta t)^2 \mathbf{f}_i$.

17.2.4　更多流体模拟算法

我们当然也可以用欧拉视角进行流体模拟，这种方法往往将流体所在的空间离散化为网格，将物理量 (如压强场、速度场等) 存储在网格的中心或面上，并在网格上将 NS 方程转换为线性方程组进行求解，如图17.11所示.

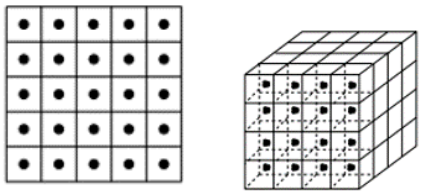


图 17.11: 网格法[5]

此外，还有结合拉格朗日视角和欧拉视角的方法，这类方法旨在保留二者的优点——拉格朗日法擅长对流，欧拉视角擅长保持流体的不可压性质. 这类方法往往需要同时维护粒子和网格两套数据结构，典型的例子有 PIC(particle-in-cell)、APIC(affine particle-in-cell)、MPM(material point method) 等.

[5]Stable Fluids