几何与物理动画

张淦淦

目录

1 几何动画

1.1 蒙皮骨骼动画

1.2 刚体运动学(Kinematics)

2 弹簧质点系统

胡克定律

$$f_{ij} = -k(||x_i - x_j||_2 - l_{ij})\frac{x_i - x_j}{||x_i - x_j||_2}$$

每个点的受力

$$f_i = \sum_{j,j\neq i} f_{ij}$$

牛顿第二定理得到点i的运动信息

$$\frac{\partial v_i}{\partial t} = \frac{1}{m_i} f_i$$

$$\frac{\partial x_i}{\partial t} = v_i$$

2.1 显示欧拉与隐式欧拉

显示欧拉 (explicit)

$$v_{t+1} = v_t + \triangle t M^{-1} f(x_t)$$

$$x_{t+1} = x_t + \triangle t v_t$$

半隐式欧拉 (Semi-implicit Euler,symplectic Euler,explicit)

$$v_{t+1} = v_t + \triangle t M^{-1} f(x_t)$$

$$x_{t+1} = x_t + \triangle t v_{t+1}$$

隐式欧拉 (Backward Euler)

$$x_{t+1} = x_t + \triangle t v_{t+1}$$

$$\begin{aligned} v_{t+1} =& v_t + \triangle t M^{-1} f(x_{t+1}) \\ =& v_t + \triangle t M^{-1} f(x_t + \triangle t v_{t+1}) \\ \approx& v_t + \triangle t M^{-1}\left[f(x_t) + \frac{\partial f}{\partial x}(x_t)\triangle t v_{t+1})\right] \end{aligned}$$

$$\big[I - \triangle t^2 M^{-1}\frac{\partial f}{\partial x}(x_t)\big]v_{t+1} = v_t + \triangle t M^{-1} f(x_t)$$

可以用Jacobi/Gauss-Seidel iterations或者Conjugate gradients求解.

融合explicit与implicit积分器

$$\big[I - \beta\triangle t^2 M^{-1}\frac{\partial f}{\partial x}(x_t)\big]v_{t+1} = v_t + \triangle t M^{-1} f(x_t)$$

- $\beta = 0$时是forward/semi-implicit(explicit)
- $\beta = 1/2$时是middle-point(implicit)
- $\beta = 1$时是backward Euler(implicit)

对于Explicit方法(forward,symplectic,RK,...),容易实现,但$\triangle t$选取过大也容易爆炸,不适用于stiff材料.

$$\triangle t \le c\sqrt{\frac{m}{k}},\ c \sim 1$$

Implicit方法(backward Euler,middle-point),一般比较难实现,每一步变得更加昂贵但是$\triangle t$可以调得大一点.

2.2 Verlet方法

基本Verlet 假设例子$t+\triangle t$时刻的位置是$r(t+\triangle t)$,将其泰勒展开

$$r(t+\triangle t) = r(t) + v(t)\triangle t + \frac{f(t)}{2m}\triangle t^2 + \frac{\partial r(t)}{3!\partial t}\triangle t^3 + O(\triangle t^4)$$

也将$r(t+\triangle t)$泰勒展开

$$r(t-\triangle t) = r(t) - v(t)\triangle t + \frac{f(t)}{2m}\triangle t^2 - \frac{\partial r(t)}{3!\partial t}\triangle t^3 + O(\triangle t^4)$$

故

$$r(t+\triangle t) + r(t-\triangle t) = 2r(t) + \frac{f(t)}{m}\triangle t^2 + O(\triangle t^4)$$

$$r(t+\triangle t) = 2r(t) - r(t-\triangle t) + \frac{f(t)}{m}\triangle t^2 + O(\triangle t^4)$$

速度只有二阶精度

$$v(t) = \frac{r(t+\triangle t) - r(t-\triangle t)}{2\triangle t} + O(\triangle t^2)$$

LeapFrog 速度和位置在时间上不同步

$$v(t-\triangle t/2) = \frac{r(t) - r(t-\triangle t)}{\triangle t}$$

$$v(t+\triangle t/2) = \frac{r(t+\triangle t) - r(t)}{\triangle t}$$

$$r(t+\triangle t) = r(t) + \triangle t v(t+\triangle t/2)$$

$$v(t+\triangle t) = v(t-\triangle t/2) + \triangle t\frac{f(t)}{m}$$

Velocity Verlet

$$v(t+\frac{1}{2}\triangle t) = v(t) + \frac{1}{2}\frac{f(t)}{m}\triangle t$$

$$r(t+\triangle t) = r(t) + v(t+\frac{1}{2}\triangle t)\triangle t$$

$$v(t+\triangle t) = v(t+\frac{\triangle t}{2}) + \frac{1}{2}\frac{f(t+\triangle t)}{m}\triangle t$$

$$r(t+\triangle t) = r(t) + v(t)\triangle t + \frac{1}{2}\frac{f(t)}{m}\triangle t^2$$

$$v(t+\triangle t) = v(t) + \frac{1}{2}(\frac{f(t)}{m} + \frac{f(t+\triangle t)}{m})\triangle t$$

3 PBD方法

下述算法是对牛顿第二运动定律的忠实描述.

Algorithm 1 Cloth simulation

$\mathbf{v}_0 \leftarrow \mathbf{0}$
for $t = 1$ **to** n **do**
  $\mathbf{M}, \mathbf{f} \leftarrow$ compute_forces$(\mathbf{x}, \mathbf{v})$
  $\mathbf{a}_t \leftarrow \mathbf{M}^{-1}\mathbf{f}$
  $\mathbf{v}_t \leftarrow \mathbf{v}_{t-1} + \mathbf{a}_t\Delta t$
  $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \mathbf{v}_t\Delta t$
  $\mathbf{x}_t \leftarrow \mathbf{x}_t +$ collision_response$(\mathbf{x}_t, \mathbf{v}_t, \mathbf{x}_t^{obs}, \mathbf{v}_t^{obs})$
  $\mathbf{v}_t \leftarrow (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$
end for

如果Δt过大,某时刻$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \boldsymbol{v}_t\Delta t$可能出现在真实世界中不可能出现的位置,从而让$\boldsymbol{f}_t(\boldsymbol{x}_t)$很大,$\boldsymbol{f}_t$又反过来影响$\boldsymbol{x}_{t+1}$,造成正反馈从而系统不稳定.

从以上分析可以发现,仿真不稳定的主要原因是,$\boldsymbol{x}$出现在了某些不该出现的位置,如果为系统加入针对位置的约束$C(\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_{n_j}})$(如,针对衣物,限制衣物质点间距离不得小于$l_0$,不得大于$l_1$),则可以保证系统的稳定性.但是如果只是限制距离,无法保证系统动量和角动量的守恒,引入额外的ghost force.

PBD系列算法解决上文提到加入质点位置约束后,保证动量角动量守恒的问题(个人理解).具体做法如下

We represent a dynamic object by a set of N vertices and M constraints. A vertex $i \in [1, \ldots, N]$ has a mass m_i, a position $\mathbf{x}_i$ and a velocity $\mathbf{v}_i$.

A constraint $j \in [1, \ldots, M]$ consists of

- a cardinality n_j,
- a function $C_j : \mathbb{R}^{3n_j} \rightarrow \mathbb{R}$,
- a set of indices $\{i_1, \ldots i_{n_j}\}$, $i_k \in [1, \ldots N]$,
- a stiffness parameter $k_j \in [0 \ldots 1]$ and
- a type of either *equality* or *inequality*.

forall vertices i
  initialize $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$
endfor
loop
  forall vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
  dampVelocities($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
  forall vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
  forall vertices i **do** generateCollisionConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
  loop solverIterations **times**
    projectConstraints($C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
  endloop
  forall vertices i
    $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
    $\mathbf{x}_i \leftarrow \mathbf{p}_i$
  endfor
  velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
endloop

解约束,已知$\boldsymbol{p}$,求$\Delta\boldsymbol{p}$使$C(\boldsymbol{p}+\Delta\boldsymbol{p})=0$,当质点质量大小一致时,当$\Delta\boldsymbol{p}$在$\nabla C_{\boldsymbol{p}}$方向上, 下降最快.

$$\Delta\boldsymbol{p} = \lambda \nabla C_{\boldsymbol{p}}$$

又由

$$C(\boldsymbol{p}+\Delta\boldsymbol{p}) \approx C(\boldsymbol{p}) + \nabla_{\boldsymbol{p}} C(\boldsymbol{p})\Delta\boldsymbol{p} = 0$$

得

$$\Delta\boldsymbol{p} = -\frac{C(\boldsymbol{p})}{|\nabla_{\boldsymbol{p}} C(\boldsymbol{p})|^2}\nabla_{\boldsymbol{p}} C(\boldsymbol{p})$$

$$\Delta\boldsymbol{p}_i = -\frac{C(\boldsymbol{p})}{\sum_j |\nabla_{\boldsymbol{p}_j} C(\boldsymbol{p})|^2}\nabla_{\boldsymbol{p}_i} C(\boldsymbol{p})$$

具体求解策略流程上类似于Gauss-Seidel,对多个约束C_i逐个更新得$\Delta\boldsymbol{p}_i^{(k)}$后,求$\boldsymbol{p} = \Delta\boldsymbol{p}_i^{(k)} + \boldsymbol{p}$,直到满足$\sum_i |C_i(\boldsymbol{p})| = 0$,可以通过设置最大迭代次数保证实时性

$\boldsymbol{p} = \boldsymbol{p}_0$
while $\sum_i |C_i(\boldsymbol{p})| != 0$ **do**
  for $i = 0 \ldots n$ **do**
    $\Delta\boldsymbol{p}_i = -\frac{C_i(\boldsymbol{p})}{|\nabla_{\boldsymbol{p}} C_i(\boldsymbol{p})|^2}\nabla_{\boldsymbol{p}_i} C_i(\boldsymbol{p})$
    $\boldsymbol{p} = \boldsymbol{p} + \Delta\boldsymbol{p}_i$
  end for
end while

当质点大小不一样时, 为了保重动量守恒, $\sum_i \boldsymbol{p}_i m_i = 0$,设$w_i = 1/m_i$.

$$\Delta\boldsymbol{p}_i = -\frac{w_i C(\boldsymbol{p})}{\sum_j w_j |\nabla_{\boldsymbol{p}_j} C(\boldsymbol{p})|^2}\nabla_{\boldsymbol{p}_i} C(\boldsymbol{p})$$

3.1 XPBD

PBD中两个关键参数对模拟的效果会产生直接影响, 一个是时间步长 time step,另一个是解算迭代次数(solverIterations times). 具体直观表现是time step越小,迭代次数越大其模拟的直观表现为越硬,反之表现的越软绵绵的效果, 这样的话, 这样的话刚度系数意义不大, 理想状态下刚性系数对魔力物体强度有直接最大影响，XPBD解决这个问题.

3.1.1 基本流程

Algorithm 1 XPBD simulation loop

predict position $\tilde{\mathbf{x}} \Leftarrow \mathbf{x}^n + \Delta t \mathbf{v}^n + \Delta t^2 \mathbf{M}^{-1}\mathbf{f}_{ext}(\mathbf{x}^n)$

initialize solve $\mathbf{x}_0 \Leftarrow \tilde{\mathbf{x}}$
initialize multipliers $\boldsymbol{\lambda}_0 \Leftarrow \mathbf{0}$
while $i < solverIterations$ **do**
  for all constraints **do**
    compute $\Delta\lambda$ using Eq (18)
    compute $\Delta\mathbf{x}$ using Eq (17)
    update $\lambda_{i+1} \Leftarrow \lambda_i + \Delta\lambda$
    update $\mathbf{x}_{i+1} \Leftarrow \mathbf{x}_i + \Delta\mathbf{x}$
  end for
  $i \Leftarrow i + 1$
end while

update positions $\mathbf{x}^{n+1} \Leftarrow \mathbf{x}_i$
update velocities $\mathbf{v}^{n+1} \Leftarrow \frac{1}{\Delta t}\left(\mathbf{x}^{n+1} - \mathbf{x}^n\right)$

$$\Delta\boldsymbol{x} = \boldsymbol{M}^{-1}\nabla\boldsymbol{C}(\boldsymbol{x}_i)^T\Delta\boldsymbol{\lambda} \quad (17)$$

$$\Delta\lambda_j = \frac{-C_j(\boldsymbol{x}_i) - \widehat{\alpha}_j\lambda_{ij}}{\nabla C_j \boldsymbol{M}^{-1}\nabla C_j^T + \widehat{\alpha}_j} \quad (18)$$

3.1.2 原理

$$\boldsymbol{M}\frac{\partial^2\boldsymbol{x}}{\partial t^2} = -\nabla\boldsymbol{U}^T(\boldsymbol{x})$$

$$\boldsymbol{U}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{C}(\boldsymbol{x})^T\boldsymbol{\alpha}^{-1}\boldsymbol{C}(\boldsymbol{x})$$

$$\boldsymbol{f}_{elastic} = -\nabla\boldsymbol{U}^T(\boldsymbol{x}) = -\nabla\boldsymbol{C}^T\boldsymbol{\alpha}^{-1}\boldsymbol{C}$$

将运动方程对时间离散化

$$\boldsymbol{M}\left(\frac{\boldsymbol{x}^{i+1} - 2\boldsymbol{x}^i + \boldsymbol{x}^{i-1}}{\Delta t^2}\right) = -\nabla\boldsymbol{U}^T(\boldsymbol{x}^{i+1})$$

设

$$\boldsymbol{\lambda}_{elastic} = -\widehat{\boldsymbol{\alpha}}^{-1}\boldsymbol{C}, \quad \widehat{\boldsymbol{\alpha}} = \frac{\boldsymbol{\alpha}}{\Delta t^2}$$

则运动方程可以写为方程组

$$\boldsymbol{M}(\boldsymbol{x}^{i+1} - 2\boldsymbol{x}^i + \boldsymbol{x}^{i-1}) - \nabla\boldsymbol{C}(\boldsymbol{x}^{i+1})^T\boldsymbol{\lambda}^{i+1} = \mathbf{0}$$

$$\boldsymbol{C}(\boldsymbol{x}^{i+1}) + \widehat{\boldsymbol{\alpha}}\boldsymbol{\lambda}^{i+1} = \mathbf{0}$$

记为

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{\lambda}) = \mathbf{0}$$

$$\boldsymbol{h}(\boldsymbol{x},\boldsymbol{\lambda}) = \mathbf{0}$$

目标是求解, 对于某一时刻

$$\boldsymbol{g}(\boldsymbol{x}_i,\boldsymbol{\lambda}_i) \neq \mathbf{0}$$

$$\boldsymbol{h}(\boldsymbol{x}_i,\boldsymbol{\lambda}_i) \neq \mathbf{0}$$

找到$\Delta\boldsymbol{x}$和$\Delta\boldsymbol{\lambda}$, 令

$$\boldsymbol{g}(\boldsymbol{x}_i + \Delta\boldsymbol{x}, \boldsymbol{\lambda}_i + \Delta\boldsymbol{\lambda}) = \mathbf{0}$$

$$\boldsymbol{h}(\boldsymbol{x}_i+\Delta\boldsymbol{x},\boldsymbol{\lambda}_i+\Delta\boldsymbol{\lambda})=\boldsymbol{0}$$

然后更新得

$$\boldsymbol{x}_{i+1}=\boldsymbol{x}_i+\Delta\boldsymbol{x}$$

$$\boldsymbol{\lambda}_{i+1}=\boldsymbol{\lambda}_i+\Delta\boldsymbol{\lambda}$$

线性化方程组

$$\begin{pmatrix}\frac{\partial \boldsymbol{g}(\boldsymbol{x}_i,\boldsymbol{\lambda}_i)}{\partial \boldsymbol{x}} & -\nabla \boldsymbol{C}^T(\boldsymbol{x}_i)\\ \nabla\boldsymbol{C}(\boldsymbol{x}_i) & \widehat{\boldsymbol{\alpha}}\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{x}\\ \Delta\boldsymbol{\lambda}\end{pmatrix}=-\begin{pmatrix}\boldsymbol{g}(\boldsymbol{x}_i,\boldsymbol{\lambda}_i)\\ \boldsymbol{h}(\boldsymbol{x}_i,\boldsymbol{\lambda}_i)\end{pmatrix}$$

为了简化运算, 提出两个假设

- 为了避免求约束的Hessian矩阵, 假设

$$\frac{\partial \boldsymbol{g}(\boldsymbol{x}_i,\boldsymbol{\lambda}_i)}{\partial \boldsymbol{x}}\approx \boldsymbol{M}$$

  省去了stiffness 和约束Hessian, 这个优化会改变收敛rate, 但不会改变最终的解
- 如果约束梯度变化不快, $\boldsymbol{g}$会一直在一个比较小的值

$$\boldsymbol{g}(\boldsymbol{x}_i,\boldsymbol{\lambda}_i)\approx 0$$

$$\begin{pmatrix}\boldsymbol{M} & -\nabla \boldsymbol{C}^T(\boldsymbol{x}_i)\\ \nabla\boldsymbol{C}(\boldsymbol{x}_i) & \widehat{\boldsymbol{\alpha}}\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{x}\\ \Delta\boldsymbol{\lambda}\end{pmatrix}=-\begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{h}(\boldsymbol{x}_i,\boldsymbol{\lambda}_i)\end{pmatrix}$$

根据Schur Complement方法求解可以得

$$[\Delta\boldsymbol{C}(\boldsymbol{x}_i)\boldsymbol{M}^{-1}\Delta\boldsymbol{C}(\boldsymbol{x}_i)^T+\widehat{\boldsymbol{\alpha}}]\Delta\boldsymbol{\lambda}=-\boldsymbol{C}(\boldsymbol{x}_i)-\widehat{\boldsymbol{\alpha}}\boldsymbol{\lambda}_i$$

$$\Delta\boldsymbol{x}=\boldsymbol{M}^{-1}\nabla\boldsymbol{C}(\boldsymbol{x}_i)^T\Delta\boldsymbol{\lambda}\quad (17)$$

对于$\Delta\boldsymbol{\lambda}$ 用Gauss-Seidel更新

$$\Delta\lambda_j=\frac{-C_j(\boldsymbol{x}_i)-\widehat{\alpha}_j\lambda_{ij}}{\nabla C_j\boldsymbol{M}^{-1}\nabla C_j^T+\widehat{\alpha}_j}\quad (18)$$

4 刚体仿真

4.1 刚体运动参数化表达

旋转矩阵 定义$SO(n)$为$\mathcal{R}^n$上的特殊正交群,$SO(n)$是以$\boldsymbol{I}$为单位元素,矩阵乘法为群操作的群.

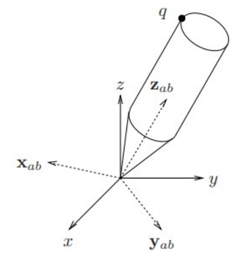
$$SO(n)=\{\boldsymbol{R}\in\mathcal{R}^{n\times n}:\boldsymbol{R}\boldsymbol{R}^T=\boldsymbol{I},det(\boldsymbol{R})=\pm 1\}$$

性质:

$$\boldsymbol{R}(\boldsymbol{v}\times\boldsymbol{u})=(\boldsymbol{R}\boldsymbol{v})\times(\boldsymbol{R}\boldsymbol{w})$$

$$\boldsymbol{R}\widehat{\boldsymbol{w}}\boldsymbol{R}^T=\widehat{(\boldsymbol{R}\boldsymbol{w})}$$

可以证明旋转是刚体运动



$$\boldsymbol{R}_{ab}=\begin{pmatrix}\boldsymbol{x}_{ab} & \boldsymbol{x}_{ab} & \boldsymbol{x}_{ab}\end{pmatrix}$$

$se(3)$群的Exponential coordinates表达 是旋转矩阵的一种参数化方法,参数空间是$\boldsymbol{\omega},\theta$, 3维空间中,给定旋转轴$\boldsymbol{\omega}$,和旋转角度$\theta$, 求满足同样旋转的旋转矩阵$\boldsymbol{R}$.

设旋转体上有点$\boldsymbol{q}$,此时若以$\boldsymbol{\omega}$为单位旋转向量,则线速度

$$\frac{\partial \boldsymbol{q}(t)}{\partial t}=\boldsymbol{\omega}\times\boldsymbol{q}(t)=\widehat{\boldsymbol{\omega}}\boldsymbol{q}(t)$$

其中

$$\widehat{\boldsymbol{\omega}}=\begin{pmatrix}0 & -a_3 & a_2\\ a_3 & 0 & -a_1\\ -a_2 & a_1 & 0\end{pmatrix}$$

定义$so(n)=\{\boldsymbol{S}\in\mathcal{R}^{n\times n}:\boldsymbol{S}^T=-\boldsymbol{S}$,$\widehat{\boldsymbol{\omega}}\in so(3)$

则微分方程解为

$$\boldsymbol{q}(t)=\boldsymbol{e}^{\widehat{\boldsymbol{\omega}}t}\boldsymbol{q}(0)$$

如果以$\boldsymbol{\omega}$为轴以单位角速度旋转θ时间,则

$$\widehat{\boldsymbol{\omega}}^2=\boldsymbol{\omega}\boldsymbol{\omega}^T-||\boldsymbol{\omega}||^2\boldsymbol{I}$$

$$\widehat{\boldsymbol{\omega}}^3=-||\boldsymbol{\omega}||^2\widehat{\boldsymbol{\omega}}$$

$$\begin{aligned}\boldsymbol{R}(\boldsymbol{\omega},\theta)=&\boldsymbol{e}^{\widehat{\boldsymbol{\omega}}\theta}=\boldsymbol{I}+\theta\widehat{\boldsymbol{\omega}}+\frac{\theta^2}{2!}\widehat{\boldsymbol{\omega}}^2+\frac{\theta^3}{3!}\widehat{\boldsymbol{\omega}}^3\dots\\ =&\boldsymbol{I}+(\theta-\frac{\theta^3}{3!}+\frac{\theta^5}{5!}+\dots)\widehat{\boldsymbol{\omega}}+(\frac{\theta^2}{2!}-\frac{\theta^4}{4!}+\frac{\theta^6}{6!}+\dots)\boldsymbol{\omega}^2\\ =&\boldsymbol{I}+\widehat{\boldsymbol{\omega}}\sin\theta+\widehat{\boldsymbol{\omega}}^2(1-\cos\theta)\end{aligned}$$

已知旋转矩阵,求其exponential coordinates表达

$$\theta=\cos^{-1}(\frac{trace(\boldsymbol{R})-1}{2})$$

$$\boldsymbol{\omega}=\frac{1}{\sin\theta}\begin{pmatrix}r_{32}-r_{23}\\ r_{13}-r_{31}\\ r_{21}-r_{12}\end{pmatrix}$$

刚体 是点的集合,无论何种力或运动作用,每个点之间的距离都是固定的.

刚体运动 一个映射$g:\mathcal{R}^3\rightarrow\mathcal{R}^3$当满足如下两个条件时是刚体运动:

- 对于所有$\boldsymbol{p},\boldsymbol{q}\in\mathcal{R}^3$

$$||g(\boldsymbol{p})-g(\boldsymbol{q})||=||\boldsymbol{p}-\boldsymbol{q}||$$

- 设$\boldsymbol{v}=\boldsymbol{q}-\boldsymbol{p}$,$g_*(\boldsymbol{v})=g(\boldsymbol{q})-g(\boldsymbol{p})$,则对于所有$\boldsymbol{v},\boldsymbol{w}\in\mathcal{R}^3$.

$$g_*(\boldsymbol{v}\times\boldsymbol{w})=g_*(\boldsymbol{v})\times g_*(\boldsymbol{w})$$

定义特殊欧拉群

$$SE(n)=\{(\boldsymbol{t},\boldsymbol{R}):\boldsymbol{t}\in\mathcal{R}^n,\boldsymbol{R}\in SO(n)\}=\mathcal{R}^n\times SO(3)$$

设$g_{ab}=(\boldsymbol{t}_{ab},\boldsymbol{R}_{ab})$,则

$$\boldsymbol{q}_a=g_{ab}(\boldsymbol{q}_b)=\boldsymbol{t}_{ab}+\boldsymbol{R}_{ab}\boldsymbol{q}_b$$

可以证明$g\in SE(n)$是刚体运动.

$$\boldsymbol{q}_b=g_{ab}^{-1}(\boldsymbol{q}_a)=\boldsymbol{R}_{ab}^T\boldsymbol{q}_a-\boldsymbol{R}_{ab}^T\boldsymbol{t}_{ab}$$

$SE(3)$群的齐次坐标系表达 如果$g=(\boldsymbol{t},\boldsymbol{R})\in SE(3)$

$$g=\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\\ 0 & 1\end{pmatrix}$$

$SE(3)$群的exponential coordinates表达 设$\boldsymbol{p}$是刚体上的某个点,绕轴$\boldsymbol{\omega}$以单位角速度旋转,$\boldsymbol{q}$是旋转轴上的某点.

$$\frac{\partial\boldsymbol{p}(t)}{\partial t}=\boldsymbol{\omega}\times(\boldsymbol{p}(t)-\boldsymbol{q})\Longrightarrow\frac{\partial\overline{\boldsymbol{p}}(t)}{\partial t}=\begin{pmatrix}\widehat{\boldsymbol{\omega}} & -\boldsymbol{w}\times\boldsymbol{q}\\ 0 & 1\end{pmatrix}\overline{\boldsymbol{p}}=\widehat{\boldsymbol{\varepsilon}}\overline{\boldsymbol{p}}$$

$$\overline{\boldsymbol{p}}(t)=\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}t}\overline{\boldsymbol{p}}(0)$$

设$\boldsymbol{v}=-\boldsymbol{w}\times\boldsymbol{q}$, 定义,$se(n)=\{(\boldsymbol{v},\widehat{\boldsymbol{\omega}}),\boldsymbol{v}\in\mathcal{R}^n,\widehat{\boldsymbol{\omega}}\in so(n)\}$,在齐次空间$\widehat{\boldsymbol{\varepsilon}}\in so(3)$.

$$\widehat{\boldsymbol{\varepsilon}}=\begin{pmatrix}\widehat{\boldsymbol{\omega}} & \boldsymbol{v}\\ 0 & 0\end{pmatrix}$$

定义$\boldsymbol{\varepsilon}=(\boldsymbol{v},\boldsymbol{\omega})\in\mathcal{R}^6$是twist$\widehat{\boldsymbol{\varepsilon}}$的twist 坐标.

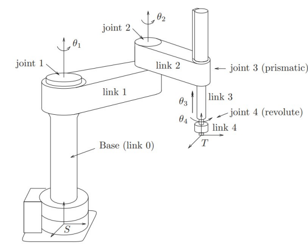
对于$\widehat{\boldsymbol{\varepsilon}}\in se(3),\theta\in\mathcal{R}$有

$$\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}\theta}=\begin{pmatrix}\boldsymbol{e}^{\widehat{\boldsymbol{\omega}}\theta} & (\boldsymbol{I}-\boldsymbol{e}^{\widehat{\boldsymbol{\omega}}\theta})(\boldsymbol{\omega}\times\boldsymbol{v})+\boldsymbol{\omega}\boldsymbol{\omega}^T\boldsymbol{v}\boldsymbol{\theta}\\ 0 & 1\end{pmatrix}\in SE(3)$$

$SE(3)$群的screw表达

4.2 刚体速度的表示

4.3 作用于刚体的力的表示



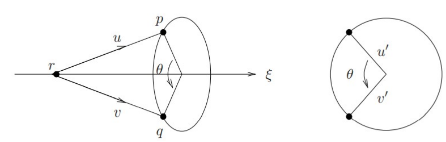
4.4 Forward Kinematics

Forward Kinematic解决的问题是, 给定各个关节之间相对configuration,求end effector在base frame S下的相应位置.

4.5 Inverse Kinematics

给定tool frame T期望的位置, 求各个关节之间相对的configuration.可以通过Paden Kahan子问题方法解决.

子问题1

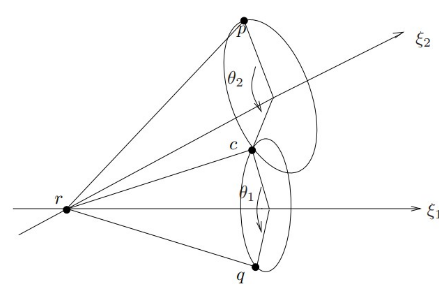


(a) (b)

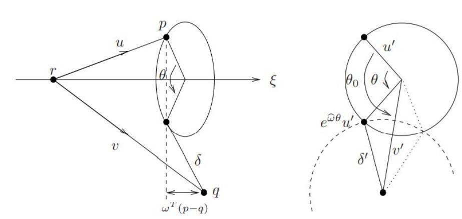
假设$\boldsymbol{\varepsilon}$不产生平移,求$\boldsymbol{\theta}$令

$$\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}\theta}\boldsymbol{p}=\boldsymbol{q}$$

子问题2



子问题3



使用子问题解决总问题　若$\boldsymbol{p}$在$\boldsymbol{\varepsilon}$的轴上,则有

$$\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}\theta}\boldsymbol{p}=\boldsymbol{p}$$

假设已知 $g,\boldsymbol{\varepsilon}_1,\boldsymbol{\varepsilon}_2,\boldsymbol{\varepsilon}_3$求$\theta_1,\theta_2,\theta_3$

$$\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_1\theta_1}\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_2\theta_2}\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_3\theta_3}=\boldsymbol{g}$$

设$\boldsymbol{p}$在$\boldsymbol{\varepsilon}$轴上则, 在两边同时乘上$\boldsymbol{p}$

$$\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_1\theta_1}\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_2\theta_2}\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_3\theta_3}\boldsymbol{p}=\boldsymbol{g}\boldsymbol{p}$$

$$\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_1\theta_1}\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_2\theta_2}\boldsymbol{p}=\boldsymbol{g}\boldsymbol{p}$$

此时变成子问题2,可以求解出θ_1,θ_2,对于θ_3,可以假设$\boldsymbol{\varepsilon}_1,\boldsymbol{\varepsilon}_2$交于$\boldsymbol{p}$.

$$\begin{aligned}\delta=&||g\boldsymbol{p}-\boldsymbol{q}||=||\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_1\theta_1}\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_2\theta_2}\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_3\theta_3}\boldsymbol{p}-\boldsymbol{q}||\\=&||\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_1\theta_1}\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_2\theta_2}(\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_3\theta_3}\boldsymbol{p}-\boldsymbol{q})||\\=&||\boldsymbol{e}^{\widehat{\boldsymbol{\varepsilon}}_3\theta_3}\boldsymbol{p}-\boldsymbol{q}||\end{aligned}$$

4.6 刚体动力学

刚体系统的运动方程可以描述为

$$\boldsymbol{\tau}=\boldsymbol{H}(q)\frac{\partial^2\boldsymbol{q}}{\partial t^2}+\boldsymbol{C}(\boldsymbol{q},\frac{\partial\boldsymbol{q}}{\partial t})$$

其中, $\boldsymbol{\tau}$是施加力, $\boldsymbol{H}$是描述inertia term的矩阵, $\boldsymbol{C}$是外力?.

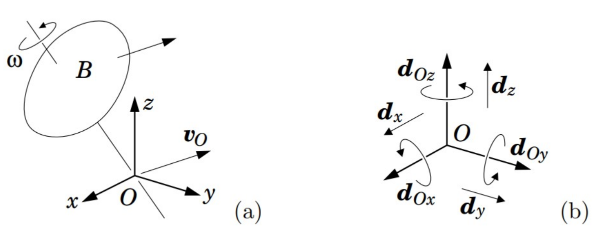
对于Forward Dynamic 和 Inverse Dynamic问题分别可以描述为

$$\frac{\partial^2\boldsymbol{q}}{\partial t^2}=FD(model,\boldsymbol{q},\frac{\partial\boldsymbol{q}}{\partial t},\boldsymbol{\tau})$$

$$\boldsymbol{\tau}=ID(model,\boldsymbol{q},\frac{\partial\boldsymbol{q}}{\partial t},\frac{\partial^2\boldsymbol{q}}{\partial t^2})$$

4.6.1 Spatial Vector Algebra

Plucker Basis on $\mathcal{M}^6$



在欧式空间中表达速度,

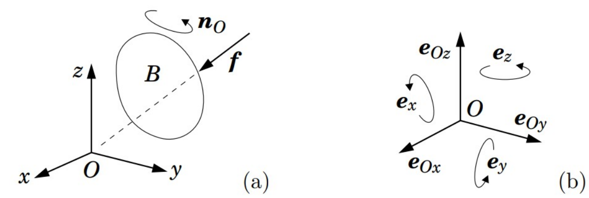
$$\boldsymbol{v}_p=\boldsymbol{v}_o+\boldsymbol{\omega}\times\overrightarrow{OP}$$

用Plucker Basis表达

$$\boldsymbol{v}=\omega_x\boldsymbol{d}_{Ox}+\omega_y\boldsymbol{d}_{Oy}+\omega_z\boldsymbol{d}_{Oz}+v_{Ox}\boldsymbol{d}_x+v_{Oy}\boldsymbol{d}_y+v_{Oz}\boldsymbol{d}_z$$

$$\underline{\widehat{\boldsymbol{v}}}_O=\begin{pmatrix}\omega_x\\\omega_y\\\omega_z\\v_{Ox}\\v_{Oy}\\v_{Oz}\end{pmatrix}=\begin{pmatrix}\underline{\boldsymbol{\omega}}\\\underline{\boldsymbol{v}}_O\end{pmatrix}$$

Plucker Basis on $\mathcal{F}^6$



在欧式空间中表达动量

$$\boldsymbol{n}_P=\boldsymbol{n}_O+\boldsymbol{f}\times\overrightarrow{OP}$$

在Plucker Basis中表达

$$\widehat{\boldsymbol{f}}=n_{Ox}\boldsymbol{e}_x+n_{Oy}\boldsymbol{e}_y+n_{Oz}\boldsymbol{e}_z+f_x\boldsymbol{e}_{Ox}f_y\boldsymbol{e}_{Oy}f_z\boldsymbol{e}_{Oz}$$

$$\underline{\widehat{\boldsymbol{v}}}_O=\begin{pmatrix}n_{Ox}\\n_{Oy}\\n_{Oz}\\f_x\\f_y\\f_z\end{pmatrix}=\begin{pmatrix}\underline{\boldsymbol{n}}_O\\\underline{\boldsymbol{f}}\end{pmatrix}$$

5 弹性体仿真

5.1 弹性体模型

设$\boldsymbol{X},\boldsymbol{x}$分别是弹性体形变前和形变后的位置. 定义deformation function $\phi:\mathcal{R}^3\rightarrow\mathcal{R}^3$, 则

$$\boldsymbol{x}=\phi(\boldsymbol{X})$$

定义deformation gradient tensor, $\boldsymbol{F}\in\boldsymbol{R}^{3\times3}$

$$\boldsymbol{F}=\frac{\partial(\phi_1,\phi_2,\phi_3)}{\partial(\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{X}_3)}=\begin{pmatrix}\frac{\partial\phi_1}{\partial\boldsymbol{X}_1}&\frac{\partial\phi_1}{\partial\boldsymbol{X}_2}&\frac{\partial\phi_1}{\partial\boldsymbol{X}_3}\\\frac{\partial\phi_2}{\partial\boldsymbol{X}_1}&\frac{\partial\phi_2}{\partial\boldsymbol{X}_2}&\frac{\partial\phi_2}{\partial\boldsymbol{X}_3}\\\frac{\partial\phi_3}{\partial\boldsymbol{X}_1}&\frac{\partial\phi_3}{\partial\boldsymbol{X}_2}&\frac{\partial\phi_3}{\partial\boldsymbol{X}_3}\end{pmatrix}$$

Strain energy 定义为$\Phi(\boldsymbol{F})$,

定义force density为$f(\boldsymbol{X})$描述force per unit undefomed volume, 定义 traction $\tau(\boldsymbol{X})$为force density function 测量force per unit undeformed area. 若$A\in\Omega,B\in\partial\Omega$

$$f_{aggregate(A)}=\int_A f(\boldsymbol{X})d\boldsymbol{X}$$

$$f_{aggregate(B)} = \int_B \tau(\boldsymbol{X}) dS$$

1st Piola-Kirchhoff stress tensor, $\boldsymbol{P} \in \mathcal{R}^{3\times3}$

$$\tau(\boldsymbol{X}) = -\boldsymbol{P}N$$

N是在reference configuration 下, 边界上朝外的法向量.

$$f(\boldsymbol{X}) = div_{\boldsymbol{X}}\boldsymbol{P}$$

$$f_i = \sum_{j=1}^{3} P_{ij,j} = \frac{\partial P_{i1}}{\partial X_1} + \frac{\partial P_{i2}}{\partial X_2} + \frac{\partial P_{i3}}{\partial X_3}$$

对于超弹性材料

$$\boldsymbol{P}(\boldsymbol{F}) = \frac{\partial \Psi(\boldsymbol{F})}{\partial \boldsymbol{F}}$$

Strain Measure Green strain tensor, $\boldsymbol{E} \in \mathcal{R}^{3\times3}$.

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{I})$$

$$\boldsymbol{E}(\boldsymbol{F}) = \underbrace{\boldsymbol{E}(\boldsymbol{I})}_{=0} + \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{F}}|_{\boldsymbol{F}=\boldsymbol{I}} : (\boldsymbol{F} - \boldsymbol{I})$$

由

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{F}} : \delta\boldsymbol{F} = \delta\boldsymbol{E} = \frac{1}{2}(\delta\boldsymbol{F}^T\boldsymbol{F} + \boldsymbol{F}^T\delta\boldsymbol{F})$$

得

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{F}} : (\boldsymbol{F}-\boldsymbol{I}) = \frac{1}{2}[(\boldsymbol{F}-\boldsymbol{I})^T\boldsymbol{I} + \boldsymbol{I}^T(\boldsymbol{F}-\boldsymbol{I})] = \frac{1}{2}(\boldsymbol{F}+\boldsymbol{F}^T) - \boldsymbol{I}$$

记$\epsilon = \boldsymbol{E}(\boldsymbol{F})$, ϵ 称为infinitesimal strain tensor 或 small strain tensor. 化非线性为线性, 在计算上更为快速.

$$\boldsymbol{\epsilon} = \frac{1}{2}(\boldsymbol{F} + \boldsymbol{F}^T) - \boldsymbol{I}$$

Linear elasticity

$$\Psi(\boldsymbol{F}) = \mu\boldsymbol{\epsilon} : \boldsymbol{\epsilon} + \frac{\lambda}{2} tr^2(\boldsymbol{\epsilon})$$

$$\mu = \frac{k}{2(1+\nu)}, \quad \lambda = \frac{k\nu}{(1+\nu)(1-2\nu)}$$

其中μ, λ是Lame coefficient, k是Young's modulus(用于measure stretch of resistance), ν 是 Poisson's ratio(a measure of incompressibility).

$$\delta\Psi = 2\mu\boldsymbol{\epsilon} : \delta\boldsymbol{\epsilon} + \lambda tr(\boldsymbol{\epsilon}) tr(\delta\boldsymbol{\epsilon}) = \underbrace{[2\mu\boldsymbol{\epsilon} + \lambda tr(\boldsymbol{\epsilon})\boldsymbol{I}]}_{=\partial\Psi/\partial\boldsymbol{F}} : \delta\boldsymbol{F}$$

$$\boldsymbol{\epsilon} : \delta\boldsymbol{\epsilon} = \boldsymbol{\epsilon} : Sym\{\delta\boldsymbol{F}\} = \boldsymbol{\epsilon} : \delta\boldsymbol{F}$$

$$tr(\delta\boldsymbol{\epsilon}) = \boldsymbol{I} : Sym\{\delta\boldsymbol{F}\} = \boldsymbol{I} : \delta\boldsymbol{F}$$

$$\delta\boldsymbol{\epsilon} = \frac{1}{2}(\delta\boldsymbol{F} + \delta\boldsymbol{F}^T) = Sym\{\delta\boldsymbol{F}\}$$

所以

$$\begin{aligned}\boldsymbol{P} =& 2\mu\boldsymbol{\epsilon} + \lambda tr(\boldsymbol{\epsilon})\boldsymbol{I} \\ =& \mu(\boldsymbol{F} + \boldsymbol{F}^T - 2\boldsymbol{I}) + \lambda tr(\boldsymbol{F} - \boldsymbol{I})\boldsymbol{I}\end{aligned}$$

$\boldsymbol{P}$是关于$\boldsymbol{F}$的线性方程, 在实现时计算速度可以很快, 此外small strain tensor只适用于小的形变场景, 不能保证rotational invariance.

StVK模型 性质: 保证rotational invariance, 但是poor resisitance to forceful compression(StVK elastic body 是compressed的).

$$\Psi(\boldsymbol{F}) = \mu\boldsymbol{E} : \boldsymbol{E} + \frac{\lambda}{t} r^2(\boldsymbol{E})$$

$$\delta\boldsymbol{E} = \frac{1}{2}(\delta\boldsymbol{F}^T\boldsymbol{F} + \boldsymbol{F}^T\delta\boldsymbol{F}) = Sym\{\boldsymbol{F}^T\delta\boldsymbol{F}\}$$

$$\boldsymbol{E} : \delta\boldsymbol{E} = \boldsymbol{E} : \{\boldsymbol{F}^T\delta\boldsymbol{F}\} = \{\boldsymbol{F}\boldsymbol{E}\} : \delta\boldsymbol{F}$$

$$tr(\delta\boldsymbol{E}) = \boldsymbol{I} : \{\boldsymbol{F}^T\delta\boldsymbol{F}\} = \boldsymbol{F} : \delta\boldsymbol{F}$$

$$\delta\Psi = 2\mu\boldsymbol{E} : \delta\boldsymbol{E} + \lambda tr(\boldsymbol{E}) tr(\delta\boldsymbol{E}) = \boldsymbol{F}[2\mu\boldsymbol{E} + \lambda tr(\boldsymbol{E})\boldsymbol{I}] : \delta\boldsymbol{F}$$

$$\boldsymbol{P}(\boldsymbol{F}) = \boldsymbol{F}[2\mu\boldsymbol{E} + \lambda tr(\boldsymbol{E})\boldsymbol{I}]$$

Corotated linear elasticity 结合linear material 和一些非线性特性保证rotational invariance

$$\boldsymbol{F} = \boldsymbol{R}\boldsymbol{S} \quad \boldsymbol{\epsilon} = \boldsymbol{S} - \boldsymbol{I}$$

$$\Psi(\boldsymbol{F}) = \mu\boldsymbol{\epsilon}_C : \boldsymbol{\epsilon}_C + \frac{\lambda}{2} tr^2(\boldsymbol{\epsilon}_C) = \mu||\boldsymbol{S} - \boldsymbol{I}||_F^2 + \frac{\lambda}{2} tr^2(\boldsymbol{S} - \boldsymbol{I})$$

$$\begin{aligned}\boldsymbol{P}(\boldsymbol{F}) =& \boldsymbol{R}[2\mu\boldsymbol{\epsilon}_C + \lambda tr(\boldsymbol{\epsilon}_C)\boldsymbol{I}] \\ =& \boldsymbol{R}[2\mu(\boldsymbol{S} - \boldsymbol{I}) + \lambda tr(\boldsymbol{S} - \boldsymbol{I})\boldsymbol{I}] \\ =& 2\mu(\boldsymbol{F} - \boldsymbol{R}) + \lambda tr(\boldsymbol{R}^T\boldsymbol{F} - \boldsymbol{I})\boldsymbol{R}\end{aligned}$$

Isotropic material and invariants

Neohookean模型 Neo-Hookean是最常用的非线性超弹性模型

$$\psi(\boldsymbol{F}) = \frac{\mu}{2}(tr(\boldsymbol{F}^T\boldsymbol{F}) - d) - \mu\log(\boldsymbol{J}) + \frac{\lambda}{2}\log^2(\boldsymbol{J})$$

$d = 2, 3$取决于物体维度,μ, λ与Young's modulus$E$和Poisson ratio的关系是

$$\mu = \frac{E}{2(1+v)}, \lambda = \frac{Ev}{(1+v)(1-2v)}$$

$$\boldsymbol{P} = \frac{\partial\psi}{\partial\boldsymbol{F}} = \mu(\boldsymbol{F})(\boldsymbol{F} - \boldsymbol{F}^{-T}) + \lambda(\boldsymbol{F})\log(\boldsymbol{J}\boldsymbol{F}^{-T})$$

Fixed Corotated Constitutive模型 简单且广泛使用,taichi语言中很多例子使用这种模型,假设polar SVD

$$\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

$$\psi(\boldsymbol{F}) = \hat{\psi}(\boldsymbol{\Sigma}) = \mu\sum_{i=1}^{d}(\sigma_i - 1)^2 + \frac{\lambda}{2}(J-1)$$

$$J = \prod_{i=1}^{d}\sigma_i$$

$$\boldsymbol{P}(\boldsymbol{F}) = \frac{\partial\psi}{\partial\boldsymbol{F}}(\boldsymbol{F}) = 2\mu(\boldsymbol{F})(\boldsymbol{F}-\boldsymbol{R}) + \lambda(\boldsymbol{F})(J-1)J\boldsymbol{F}^{-T}$$

综上,计算$\boldsymbol{P}$的流程

$$\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

$$\psi(\boldsymbol{F}) = \hat{\psi}(\boldsymbol{\Sigma})$$

$$\boldsymbol{R} = \boldsymbol{R}\boldsymbol{V}^T$$

$$\boldsymbol{P} = \boldsymbol{U}\hat{\boldsymbol{P}}\boldsymbol{V}^T = \boldsymbol{U}diag(\frac{\hat{\psi}}{\partial\sigma_0}, \dots, \frac{\hat{\psi}}{\partial\sigma_i}, \dots)\boldsymbol{V}^T$$

5.2 FEM方法

在有限的顶点$\boldsymbol{X}_1, \dots, \boldsymbol{X}_N$上储存deformation map, $\Phi(\boldsymbol{X})$, $\boldsymbol{x}_i = \phi(\boldsymbol{X}_i)$, 记$\boldsymbol{x} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)$, 对于超弹性材料, 对于任意给定的deformation $\phi(\boldsymbol{x})$的starin energy

$$E(\boldsymbol{\phi}) = \int_{\Omega}\Psi(\boldsymbol{F})d\boldsymbol{X}$$

对于离散情况, 如利用四面体离散出来的弹性体, 通过插值得出$\boldsymbol{X}$对应的$\boldsymbol{x}$, 进而求出$\widehat{\Phi}(\boldsymbol{X}, \boldsymbol{x})$

$$E(\boldsymbol{x}) = E[\widehat{\Phi}(\boldsymbol{X}, \boldsymbol{x})] = \int_{\Omega}\Psi(\widehat{\boldsymbol{F}}(\boldsymbol{X}, \boldsymbol{x}))d\boldsymbol{X}$$

其中$\widehat{\boldsymbol{F}}(\boldsymbol{X}, \boldsymbol{x}) = \partial\widehat{\phi}(\boldsymbol{X}, \boldsymbol{x})/\partial\boldsymbol{X}$

Mesh上每个点的受力为

$$\boldsymbol{f}_i(\boldsymbol{x}_0) = \frac{\partial E(\boldsymbol{x})}{\partial\boldsymbol{x}_i}$$

实际计算中, 对于每个elements, Ω_e

$$E(\boldsymbol{x}) = \sum_e E^e(\boldsymbol{x}) = \sum_e\int_{\Omega_e}\Psi(\widehat{\boldsymbol{F}}(\boldsymbol{X}, \boldsymbol{x})d\boldsymbol{X}$$

每个点上的力$\boldsymbol{f}_i$可以通过all elements in neighborhood $\mathcal{N}_i$计算.

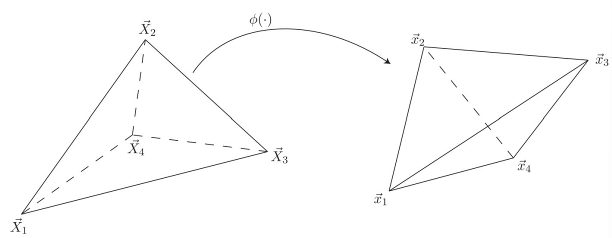
$$\boldsymbol{f}_i(\boldsymbol{x}_0) = \sum_{e\in\mathcal{N}_i}\boldsymbol{f}_i^e(\boldsymbol{x}), \quad \boldsymbol{f}_i^e(\boldsymbol{x}) = -\frac{\partial E^e(\boldsymbol{x})}{\partial\boldsymbol{x}_i}$$

对于任意一个四面体$\mathcal{T}_i$

$$\widehat{\phi}(\boldsymbol{X}) = \boldsymbol{A}_i\boldsymbol{X} + \boldsymbol{b}, \boldsymbol{X} \in \mathcal{T}_i$$

其中$\boldsymbol{A}_i \in \mathcal{R}^{3\times3}$且$\boldsymbol{b} \in \mathcal{R}^3$,由$\boldsymbol{F} = \partial\widehat{\phi}/\partial\boldsymbol{X} = \boldsymbol{A}_i$.

$$\widehat{\phi}(\boldsymbol{X}) = \boldsymbol{F}\boldsymbol{X} + \boldsymbol{b}, \boldsymbol{X} \in \mathcal{T}_i$$



$$\begin{cases}\boldsymbol{x}_1 = \boldsymbol{F}\boldsymbol{X}_1 + \boldsymbol{b}\\ \boldsymbol{x}_2 = \boldsymbol{F}\boldsymbol{X}_2 + \boldsymbol{b}\\ \boldsymbol{x}_3 = \boldsymbol{F}\boldsymbol{X}_3 + \boldsymbol{b}\\ \boldsymbol{x}_4 = \boldsymbol{F}\boldsymbol{X}_4 + \boldsymbol{b}\end{cases} \Rightarrow \begin{cases}\boldsymbol{x}_1 - \boldsymbol{x}_4 = \boldsymbol{F}(\boldsymbol{X}_1 - \boldsymbol{X}_4)\\ \boldsymbol{x}_2 - \boldsymbol{x}_4 = \boldsymbol{F}(\boldsymbol{X}_2 - \boldsymbol{X}_4)\\ \boldsymbol{x}_3 - \boldsymbol{x}_4 = \boldsymbol{F}(\boldsymbol{X}_3 - \boldsymbol{X}_4)\end{cases}$$

$$\boldsymbol{D}_s = \boldsymbol{F}\boldsymbol{D}_m$$

$$\boldsymbol{D}_s = \begin{pmatrix}\boldsymbol{x}_1 - \boldsymbol{x}_4 & \boldsymbol{x}_2 - \boldsymbol{x}_4 & \boldsymbol{x}_3 - \boldsymbol{x}_4\end{pmatrix}$$

$$\boldsymbol{D}_m = \begin{pmatrix}\boldsymbol{X}_1 - \boldsymbol{X}_4 & \boldsymbol{X}_2 - \boldsymbol{X}_4 & \boldsymbol{X}_3 - \boldsymbol{X}_4\end{pmatrix}$$

四面体的undeformed volume等于$W = \frac{1}{6}|det\boldsymbol{D}_m|$

$$\boldsymbol{E}_i = \int_{\mathcal{T}_i}\Psi(\boldsymbol{F})d\boldsymbol{X} = \Psi(\boldsymbol{F}_i)\int_{\mathcal{T}_i}d\boldsymbol{X} = W\cdot\Psi(\boldsymbol{F}_i)$$

对于四个点上的力$\boldsymbol{f}_{ik} = -\partial E_i(\boldsymbol{x})/\partial\boldsymbol{x}_k$.

$$\boldsymbol{H} = \begin{pmatrix}\boldsymbol{f}_1 & \boldsymbol{f}_2 & \boldsymbol{f}_3\end{pmatrix} = -\boldsymbol{W}\boldsymbol{P}(\boldsymbol{F})\boldsymbol{D}_m^{-T}, \quad \boldsymbol{f}_4 = -\boldsymbol{f}_1 - \boldsymbol{f}_2 - \boldsymbol{f}_3$$

计算tetrahedral mesh 上所有顶点的力

Algorithm 1 Batch computation of elastic forces on a tetrahedral mesh

procedure PRECOMPUTATION($\mathbf{x}, \mathbf{B}_m[1\dots M], W[1\dots M]$)
  for each $\mathcal{T}_e = (i,j,k,l) \in \mathcal{M}$ **do** ▷ M is the number of tetrahedra
    $\mathbf{D}_m \leftarrow \begin{bmatrix} X_i - X_l & X_j - X_l & X_k - X_l \\ Y_i - Y_l & Y_j - Y_l & Y_k - Y_l \\ Z_i - Z_l & Z_j - Z_l & Z_k - Z_l \end{bmatrix}$
    $\mathbf{B}_m[e] \leftarrow \mathbf{D}_m^{-1}$
    $W[e] \leftarrow \frac{1}{6}\det(\mathbf{D}_m)$ ▷ W is the undeformed volume of $\mathcal{T}_e$
  end for
end procedure
procedure COMPUTEELASTICFORCES($\mathbf{x}, \mathbf{f}, \mathcal{M}, \mathbf{B}_m[], W[]$)
  $\mathbf{f} \leftarrow \mathbf{0}$ ▷ $\mathcal{M}$ is a tetrahedral mesh
  for each $\mathcal{T}_e = (i,j,k,l) \in \mathcal{M}$ **do**
    $\mathbf{D}_s \leftarrow \begin{bmatrix} x_i - x_l & x_j - x_l & x_k - x_l \\ y_i - y_l & y_j - y_l & y_k - y_l \\ z_i - z_l & z_j - z_l & z_k - z_l \end{bmatrix}$
    $\mathbf{F} \leftarrow \mathbf{D}_s\mathbf{B}_m[e]$
    $\mathbf{P} \leftarrow \mathbf{P}(\mathbf{F})$ ▷ From the constitutive law
    $\mathbf{H} \leftarrow -W[e]\mathbf{P}(\mathbf{B}_m[e])^T$
    $\vec{f}_i \mathrel{+}= \vec{h}_1, \vec{f}_j \mathrel{+}= \vec{h}_2, \vec{f}_k \mathrel{+}= \vec{h}_3$ ▷ $\mathbf{H} = [\vec{h}_1\ \vec{h}_2\ \vec{h}_3]$
    $\vec{f}_l \mathrel{+}= (-\vec{h}_1 - \vec{h}_2 - \vec{h}_3)$
  end for
end procedure

显式欧拉算法即可进行仿真过程, 定义$\boldsymbol{f}_e(\boldsymbol{x}^*)$是$\boldsymbol{x}^*$上的弹力,$\boldsymbol{K}(\boldsymbol{x}^*) = -\frac{\partial\boldsymbol{f}_e}{\partial\boldsymbol{x}}|_{\boldsymbol{x}^*}$是elasticity stiffness matrix, $\boldsymbol{f}_d(\boldsymbol{x}^*, \boldsymbol{v}^*) = -\gamma\boldsymbol{K}(\boldsymbol{x}^*)\boldsymbol{v}^*$ Damping forces, $\boldsymbol{f}(\boldsymbol{x}^*, \boldsymbol{v}^*) = \boldsymbol{f}_e(\boldsymbol{x}^*) + \boldsymbol{f}_d(\boldsymbol{x}^*, \boldsymbol{v}^*)$ 是 aggregate forces. $\boldsymbol{M}$是mass matrix,$\boldsymbol{x}^{n+1}, \boldsymbol{v}^{n+1}$是$t^{n+1}$时刻的位置和速度.

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \triangle t\boldsymbol{v}^{n+1}$$

$$
\begin{aligned}
\boldsymbol{v}^{n+1} =& \boldsymbol{v}^{n} + \triangle t \boldsymbol{M}^{-1}\boldsymbol{f}(\boldsymbol{x}^{n+1}, \boldsymbol{v}^{n+1}) \\
=& \boldsymbol{v}^{n} + \triangle t \boldsymbol{M}^{-1}(\boldsymbol{f}_e(\boldsymbol{x}^{n+1}) + \boldsymbol{f}_d(\boldsymbol{x}^{n+1}, \boldsymbol{v}^{n+1}))
\end{aligned}
$$

由于上述Backward Euler System 是非线性的, 所以用一个序列$\boldsymbol{x}^{n+1}_{(k)}, \boldsymbol{v}^{n+1}_{(k)}$趋近$\boldsymbol{x}^{n+1}, \boldsymbol{v}^{n+1}$.

$$
\triangle \boldsymbol{x}_{(k)} = \boldsymbol{x}^{n+1}_{(k+1)} - \boldsymbol{x}^{n+1}_{(k)}, \quad \triangle \boldsymbol{v}_{(k)} = \boldsymbol{v}^{n+1}_{(k+1)} - \boldsymbol{v}^{n+1}_{(k)},
$$

$$
\begin{aligned}
\boldsymbol{v}^{n+1}_{(k)} + \triangle \boldsymbol{v}_{(k)} =& \boldsymbol{v}^{n} + \triangle t \boldsymbol{M}^{-1}(\boldsymbol{f}_e(\boldsymbol{x}^{n+1}_{(k)} + \frac{\partial \boldsymbol{f}_e}{\partial \boldsymbol{x}}|_{\boldsymbol{x}^{n+1}_{(k)}} \cdot \triangle \boldsymbol{x} \\
& - \gamma \boldsymbol{K}(\boldsymbol{x}^{n+1}_{(k)})(\boldsymbol{v}^{n+1}_{(k)} + \triangle \boldsymbol{v}))
\end{aligned}
$$

$$
\begin{aligned}
\frac{1}{\triangle t^2}\boldsymbol{M}\triangle \boldsymbol{x} =& \frac{1}{\triangle t}\boldsymbol{M}(\boldsymbol{v}^{n} - \boldsymbol{v}^{n+1}_{(k)}) + (\boldsymbol{f}_e(\boldsymbol{x}^{n+1}_{(k)} \\
& - \boldsymbol{K}(\boldsymbol{x}^{n+1}_{(k)}\triangle \boldsymbol{x} - \gamma \boldsymbol{K}(\boldsymbol{x}^{n+1}_{(k)})(\boldsymbol{v}^{n+1}_{(k)} + \triangle \boldsymbol{v}))
\end{aligned}
$$

$$
\begin{aligned}
&[(1 + \frac{\gamma}{\triangle t})\boldsymbol{K}(\boldsymbol{x}^{n+1}_{(k)} + \frac{1}{\triangle t^2}\boldsymbol{M}\triangle \boldsymbol{x}] = \\
&= \frac{1}{\triangle t}\boldsymbol{M}(\boldsymbol{v}^{n} - \boldsymbol{v}^{n+1}_{(k)} + (\boldsymbol{f}_e(\boldsymbol{x}^{n+1}_{(k)}) - \gamma \boldsymbol{K}(\boldsymbol{x}^{n+1}_{(k)}\boldsymbol{v}^{n+1}_{(k)}) \\
&= \frac{1}{\triangle t}\boldsymbol{M}(\boldsymbol{v}^{n} - \boldsymbol{v}^{n+1}_{(k)} + (\boldsymbol{f}_e(\boldsymbol{x}^{n+1}_{(k)}) + \boldsymbol{f}_d(\boldsymbol{x}^{n+1}_{(k)}, \boldsymbol{v}^{n+1}_{(k)}) \\
&= \frac{1}{\triangle t}\boldsymbol{M}(\boldsymbol{v}^{n} - \boldsymbol{v}^{n+1}_{(k)} + \boldsymbol{f}(\boldsymbol{x}^{n+1}_{(k)}, \boldsymbol{v}^{n+1}_{(k)})
\end{aligned}
$$

5.3 MPM方法

MPM(Material Point Method)方法最开始是针对各向同性体弹性物体(如:流体,果冻,雪等)的仿真方法.

针对各向同性体弹性物体仿真最朴素的观点就是将物体视作很多个弹性“粒子”构成.粒子间相互作用力决定了物体运动(拉格朗日观点). 描述成粒子可以方便的表达..,但存在.. 的问题.

5.3.1 APIC仿真框架

插值函数定义及其性质

$$
w^n_{ip} = N_i(\boldsymbol{x}^n_p)
$$

$$
N_i(\boldsymbol{x}_p) = N(\frac{1}{h}(x_p - x_i))N(\frac{1}{h}(y_p - y_i))N(\frac{1}{h}(z_p - z_i))
$$

常用的插值函数有quadratic B splines或cubic B splines(分别如下),后者比前者计算开销大但是数值更稳定.

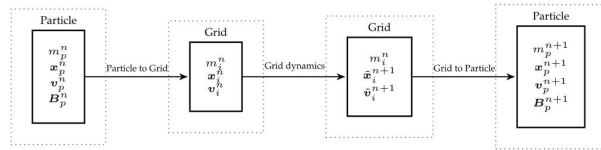
$$
\mathrm{N}(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3} & 0 \leqslant |x| < 1 \\ \frac{1}{6}(2 - |x|)^3 & 1 \leqslant |x| < 2 \\ 0 & 2 \leqslant |x| \end{cases} \qquad \mathrm{N}(x) = \begin{cases} \frac{3}{4} - |x|^2 & 0 \leqslant |x| < \frac{1}{2} \\ \frac{1}{2}\left(\frac{3}{2} - |x|\right)^2 & \frac{1}{2} \leqslant |x| < \frac{3}{2} \\ 0 & \frac{3}{2} \leqslant |x| \end{cases}
$$

$$
\sum_i w^n_{ip} = 1
$$

$$
\sum_i w^n_{ip}\boldsymbol{x}^n_i = \boldsymbol{x}^n_p
$$

$$
\sum_i w^n_{ip}(\boldsymbol{x}^n_i - \boldsymbol{x}^n_p) = 0
$$

APIC的算法更新迭代流程,可证明这个迭代过程中可以保证G2P和P2G的动量以及角动量守恒.



从particles到grid

$$
\begin{aligned}
m^n_i &= \sum_p w^n_{ip} m_p \\
\boldsymbol{D}^n_p &= \sum_i w^n_{ip}(\boldsymbol{x}^n_i - \boldsymbol{x}^n_p)(\boldsymbol{x}^n_i - \boldsymbol{x}^n_p)^T = \sum_i w^n_{ip}\boldsymbol{x}^n_i(\boldsymbol{x}^n_i)^T - \boldsymbol{x}^n_p(\boldsymbol{x}^n_p)^T \\
m^n_i \boldsymbol{v}^n_i &= \sum_p w^n_{ip} m_p(\boldsymbol{v}^n_p + \boldsymbol{B}^n_p(\boldsymbol{D}^n_p)^{-1}(\boldsymbol{x}^n_i - \boldsymbol{x}^n_p))
\end{aligned}
$$

从grid到particle

$$
\begin{aligned}
\boldsymbol{v}^{n+1}_p &= \sum_i w^n_{ip}\tilde{\boldsymbol{v}}^{n+1}_i \\
\boldsymbol{B}^{n+1}_p &= \sum_i w^n_{ip}\tilde{\boldsymbol{v}}^{n+1}_i(\boldsymbol{x}^n_i - \boldsymbol{x}^n_p)^T.
\end{aligned}
$$

在quadratic 情况下

$$
\boldsymbol{D}^n_p = \frac{1}{4}\triangle x^2 \boldsymbol{I}
$$

在cubic 情况下

$$
\boldsymbol{D}^n_p = \frac{1}{3}\triangle x^2 \boldsymbol{I}
$$

$\triangle x$是网格间距距离(从代码Demo中习得,需要考证3维下是否一致)

故APIC流程伪代码

1. Particle to grid (P2G)
   - $(m\mathbf{v})^{n+1}_i = \sum_p w_{ip}[m_p \mathbf{v}^n_p + m_p \mathbf{C}^n_p(\mathbf{x}_i - \mathbf{x}^n_p)]$ (Grid momentum)
   - $m^{n+1}_i = \sum_p m_p w_{ip}$ (Grid mass)
2. Grid operations
   - $\hat{\mathbf{v}}^{n+1}_i = (m\mathbf{v})^{n+1}_i / m^{n+1}_i$ (Grid velocity)
   - Apply Chorin-style pressure projection: $\mathbf{v}^{n+1} = \textbf{Project}(\hat{\mathbf{v}}^{n+1})$
3. Grid to particle (G2P)
   - $\mathbf{v}^{n+1}_p = \sum_i w_{ip}\mathbf{v}^{n+1}_i$ (Particle velocity)
   - $\mathbf{C}^{n+1}_p = \frac{4}{\Delta x^2}\sum_i w_{ip}\mathbf{v}^{n+1}_i(\mathbf{x}_i - \mathbf{x}^n_p)^T$ (Particle velocity gradient)
   - $\mathbf{x}^{n+1}_p = \mathbf{x}^n_p + \Delta t \mathbf{v}^{n+1}_p$ (Particle position)

上图写漏了Deformation Gradient的更新

$$
\boldsymbol{F}^{n+1}_p = (\boldsymbol{I} + \triangle t \sum_i \boldsymbol{v}^{n+1}_i(\nabla w^n_{ip})^T)\boldsymbol{F}^n_p
$$

在Grid Operation中,要通过力来更新速度

$$
\boldsymbol{v}^{n+1}_i = \hat{\boldsymbol{v}}^{n+1}_i - \frac{\boldsymbol{f}_i}{m_i}
$$

$$
\boldsymbol{f}_i = -\frac{\partial e}{\partial \boldsymbol{x}_i}(\boldsymbol{x}) = \frac{-\partial \sum_p V^0_p \psi_p(\boldsymbol{F}_p)}{\partial \boldsymbol{x}}
$$

除此之外需要额外做碰撞检测(如:布料与人体的碰撞),添加重力和摩擦力等,排除额外处理,物体的运动规律完全由$\psi_p(\boldsymbol{F_p})$控制.

5.3.2 MLSMPM仿真框架

MLSMPM可以视作是对APIC的优化,Particle 中要储存的信息,位置$\boldsymbol{x}_p$,速度$\boldsymbol{v}_p$,Deformation gradient$\boldsymbol{F}_p$,APIC中的Affine momentum$\boldsymbol{C}_p$

Grid 中只要储存速度和质量$\boldsymbol{v}_i, m_i$

1. Particle to grid (P2G)
   - $\mathbf{F}_p^{n+1} = (\mathbf{I} + \Delta t \mathbf{C}_p^n)\mathbf{F}_p^n, \ldots$ **(Deformation update)**
   - $(m\mathbf{v})_i^{n+1} = \sum_p w_{ip}\{m_p \mathbf{v}_p^n + [m_p \mathbf{C}_p^n - \frac{4\Delta t}{\Delta x^2}\sum_p V_p^0 \mathbf{P}(\mathbf{F}_p^{n+1})(\mathbf{F}_p^{n+1})^T](\mathbf{x}_i - \mathbf{x}_p^n)\}$ **(Grid momentum)**
   - $m_i^{n+1} = \sum_p m_p w_{ip}$ (Grid mass)
2. Grid operations
   - $\hat{\mathbf{v}}_i^{n+1} = (m\mathbf{v})_i^{n+1}/m_i^{n+1}$ (Grid velocity)
   - $\mathbf{v}_i^{n+1} = \mathrm{BC}(\hat{\mathbf{v}}_i^{n+1})$ (Grid boundary condition. BC is the boundary condition operator.)
3. Grid to particle (G2P)
   - $\mathbf{v}_p^{n+1} = \sum_i w_{ip}\mathbf{v}_i^{n+1}$ (Particle velocity)
   - $\mathbf{C}_p^{n+1} = \frac{4}{\Delta x^2}\sum_i w_{ip}\mathbf{v}_i^{n+1}(\mathbf{x}_i - \mathbf{x}_p^n)^T$ (Particle velocity gradient)
   - $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1}$ (Particle position)

BC表示网格Boundray Condition,比如,衣物(particle)如果和人体(some grid node)发生碰撞,将发生碰撞地方速度归零$\boldsymbol{v}_i = 0$

物体的运动规律完全由$\boldsymbol{P}(\boldsymbol{F}_p) = \frac{\partial \psi_p}{\partial \boldsymbol{F}_p}$控制.

具体物理对象的仿真 从上文的讨论中可以发现,μ, λ系数决定了弹性物体的运动规律.以冰雪仿真为例(mls-mpm88.cpp)

$$\boldsymbol{F} = \boldsymbol{F}_E \boldsymbol{F}_P$$

$$\mu(\boldsymbol{F}_p) = \mu_0 e^{\epsilon(1-J_p)}, \lambda(\boldsymbol{F}_p) = \lambda_0 e^{\epsilon(1-J_p)}$$

$$J_p = det(\boldsymbol{F}_p)$$

$\boldsymbol{F}_P$计算流程

$$\hat{\boldsymbol{F}}_E^{n+1} = \boldsymbol{F}^{n+1}(\boldsymbol{F}_P^n)^{-1}$$

$$\hat{\boldsymbol{F}}_E^{n+1} = \boldsymbol{U}_E^{n+1}\hat{\boldsymbol{\Sigma}}_E^{n+1}(\boldsymbol{V}_E^{n+1})^T$$

$$\sigma_{Ei}^{n+1} = clamp(\hat{\sigma}_{Ei}^{n+1}, 1-\theta_c, 1+\theta_s)$$

$$\boldsymbol{F}_E^{n+1} = \boldsymbol{U}_E^{n+1}\boldsymbol{\Sigma}_E^{n+1}(\boldsymbol{V}_E^{n+1})^T$$

$$\boldsymbol{F}_p^{n+1} = (\boldsymbol{F}_P^{n+1})^{-1}\boldsymbol{F}^{n+1}$$

ϵ是hardening因子,这里取10.0,μ, λ是Lame系数,由E, v表达,E, v分别取1e4,0.2

$$\mu_0 = \frac{E}{2(1+v)}, \lambda_0 = \frac{Ev}{(1+v)(1-2v)}$$

6 流体仿真

6.1 原理

对于一个温度和密度各处均匀的流体, 其状态可以用向量场$\boldsymbol{u}$和压力场p来描述, $\boldsymbol{u}, p$受边界条件影响随时间和空间变化 , 假设边界条件和和$\boldsymbol{u}(t_0, \boldsymbol{x}), p(t_0, \boldsymbol{x})$已知,则之后任意时刻流体状态$\boldsymbol{u}(t, \boldsymbol{x}), p(t, \boldsymbol{x})$可由 Navier-Stokes 方程推出

$$\nabla \cdot \boldsymbol{u} = 0$$
$$\frac{\partial \boldsymbol{u}}{\partial t} = -(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \frac{1}{\rho}\nabla p + v\nabla^2 \boldsymbol{u} + \boldsymbol{f}$$

其中ρ为流体密度, v为流体动力粘度(kinematic viscosity), $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$, $\nabla^2 = \nabla \cdot \nabla$.假设在流体被限制在领域D内.

根据Helmholtz-Hodge Decomposition定理, 对于任意一个向量场$\boldsymbol{w}$, 可以分解唯一分解为一个mass conserving field和一个 gradient field之和, 设

$$\boldsymbol{w} = \boldsymbol{u} + \nabla p, \quad \nabla \cdot \boldsymbol{u} = 0$$

$$\mathrm{N}(x) = \begin{cases} \frac{3}{4} - |x|^2 & 0 \leqslant |x| < \frac{1}{2} \\ \frac{1}{2}\left(\frac{3}{2} - |x|\right)^2 & \frac{1}{2} \leqslant |x| < \frac{3}{2} \\ 0 & \frac{3}{2} \leqslant |x| \end{cases}$$

已知$\boldsymbol{w}$, 等式两边同乘∇

$$\nabla^2 p = \nabla \boldsymbol{w}$$

则对于p来说是一个给定Neumann边界条件($\partial p/\partial n = 0$ on ∂D)泊松方程, p可求. 定义操作符$\boldsymbol{P}$

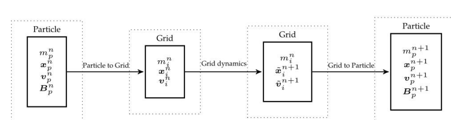
$$\boldsymbol{u} = \boldsymbol{P}\boldsymbol{w} = \boldsymbol{w} - \nabla p$$

改写Navier-Stokes方程为

$$\nabla \cdot \boldsymbol{u} = 0$$
$$\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{P}(-(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + v\nabla^2 \boldsymbol{u} + \boldsymbol{f})$$

以下讨论Navier-Stokes方程求解,使用 [?]提供的方法, 一定程度上牺牲物理真实性, 但是速度快且无条件稳定. 首先讨论时间上离散化

$$\boldsymbol{u}(t, \boldsymbol{x}) = \boldsymbol{w}_0(\boldsymbol{x}) \overbrace{\rightarrow}^{add\ force} \boldsymbol{w}_1(\boldsymbol{x}) \overbrace{\rightarrow}^{advect} \boldsymbol{w}_2(\boldsymbol{x}) \overbrace{\rightarrow}^{diffuse} \boldsymbol{w}_3(\boldsymbol{x}) \overbrace{\rightarrow}^{project} \boldsymbol{w}_4(\boldsymbol{x}) = \boldsymbol{u}(t + \triangle t, \boldsymbol{x})$$



求解采用一种propagation的方式,

$$\frac{\partial \boldsymbol{u}}{\partial t} = f, \ \boldsymbol{u}(0) = \boldsymbol{w}_0 \ \overbrace{\rightarrow}^{\boldsymbol{w}_1 = \boldsymbol{u}(\triangle t)}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} = -\boldsymbol{u} \cdot \nabla \boldsymbol{u}, \ \boldsymbol{u}(0) = \boldsymbol{w}_1 \ \overbrace{\rightarrow}^{\boldsymbol{w}_2 = u(\triangle t)}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} = v\nabla^2 \boldsymbol{u}, \ \boldsymbol{u}(0) = \boldsymbol{w}_2 \ \overbrace{\rightarrow}^{\boldsymbol{w}_3 = \boldsymbol{u}(\triangle t)}$$

$$\boldsymbol{w}_4 = \boldsymbol{P}(\boldsymbol{w}_3)$$

add force

$$\frac{\partial \boldsymbol{u}}{\partial t} = f,\ \boldsymbol{u}(0) = \boldsymbol{w}_0$$

$$\boldsymbol{w}_1(\boldsymbol{x}) = \boldsymbol{w}_0(\boldsymbol{x}) + \triangle t \boldsymbol{f}(\boldsymbol{x}, t)$$

advect

$$\frac{\partial \boldsymbol{u}}{\partial t} = -\boldsymbol{u} \cdot \nabla \boldsymbol{u},\ \boldsymbol{u}(0) = \boldsymbol{w}_1$$

此处做一个近似, 时间步长越大误差越大

$$\frac{\partial \boldsymbol{u}}{\partial t} = -\boldsymbol{w}_1 \cdot \nabla \boldsymbol{u}$$

利用Method of Characteristics 求解此问题,假设欲求

$$\frac{\partial \boldsymbol{a}(\boldsymbol{x}, t)}{\partial t} = -\boldsymbol{v}(\boldsymbol{x}) \cdot \nabla \boldsymbol{a}(\boldsymbol{x}, t),\ \boldsymbol{a}(x, 0) = \boldsymbol{a}_0(\boldsymbol{x})$$

对于某个$\boldsymbol{x}_0$, 假设$\boldsymbol{p}(\boldsymbol{x}_0, t)$, 有

$$\boldsymbol{p}(\boldsymbol{x}_0, 0) = \boldsymbol{x}_0,\ \frac{\mathrm{d}\boldsymbol{p}(\boldsymbol{x}_0, t)}{\mathrm{d}t} = \boldsymbol{v}(\boldsymbol{x}_0)$$

则

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{a}(\boldsymbol{p}(\boldsymbol{x}_0, t), t)}{\mathrm{d}t} =& \nabla \boldsymbol{a} \cdot \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} + \frac{\partial \boldsymbol{a}}{\partial t} \\ =& \nabla \boldsymbol{a} \cdot \boldsymbol{v}(\boldsymbol{x}_0) - \boldsymbol{v}(\boldsymbol{x}_0) \cdot \nabla \boldsymbol{a} = 0\end{aligned}$$

即$\boldsymbol{a}(\boldsymbol{p}(\boldsymbol{x}_0, t), t)$是常数.

advect 步骤, 定义与partial streamline 对应的路径$\boldsymbol{p}(\boldsymbol{x}, s)$, 则

$$\boldsymbol{w}_2(\boldsymbol{x}) = \boldsymbol{w}_1(\boldsymbol{p}(\boldsymbol{x}, -\triangle t))$$

$$\boldsymbol{p}(\boldsymbol{x}, -\triangle t) = \boldsymbol{x} - \triangle t \boldsymbol{w}_1(\boldsymbol{x})$$

$$\mathrm{N}(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3} & 0 \leqslant |x| < 1 \\ \frac{1}{6}(2 - |x|)^3 & 1 \leqslant |x| < 2 \\ 0 & 2 \leqslant |x| \end{cases}$$

diffuse 这种方式牺牲物理上的精确性(?), 但稳定且便于实现.

考虑粘度影响

$$\frac{\partial \boldsymbol{u}}{\partial t} = v \nabla^2 \boldsymbol{u}$$

若采用显示欧拉解原微分方程

$$\frac{\boldsymbol{u}(t + \triangle t) - \boldsymbol{u}(t)}{\triangle t} = v \nabla^2 \boldsymbol{u}(t)$$

$$\boldsymbol{u}(t + \triangle t) = (\boldsymbol{I} + \triangle t v \nabla^2) \boldsymbol{u}(t)$$

则diffuse过程可以写为

$$\boldsymbol{w}_3(\boldsymbol{x}) = (\boldsymbol{I} + v \triangle t \nabla^2) \boldsymbol{w}_2(\boldsymbol{x})$$

显示对步长有要求, 为得到稳定的仿真方法, 使用隐式欧拉

$$\frac{\boldsymbol{u}(t + \triangle t) - \boldsymbol{u}(t)}{\triangle t} = v \nabla^2 \boldsymbol{u}(t + \triangle t)$$

$$\boldsymbol{w}_3(\boldsymbol{x}) = (\boldsymbol{I} - v \triangle t \nabla^2)^{-1} \boldsymbol{w}_2(\boldsymbol{x})$$

project 此步需要解possion方程, 可以使用差分方法, 最后变为求解一个线性方程.

$$\nabla^2 p = \nabla \cdot \boldsymbol{w}_3, \quad \boldsymbol{w}_4 = \boldsymbol{w}_3 - \nabla p$$

6.2 实现

[?]给出了一种实时的实现方法. 考虑2维情况, 密度和速度都定义在网格中心, 空间中实际网格为$N \times N$, 边缘多加一层用于考虑boundary conditions.

$$\begin{aligned} m_i^n &= \sum_p w_{ip}^n m_p \\ \boldsymbol{D}_p^n &= \sum_i w_{ip}^n (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)(\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)^T = \sum_i w_{ip}^n \boldsymbol{x}_i^n (\boldsymbol{x}_i^n)^T - \boldsymbol{x}_p^n (\boldsymbol{x}_p^n)^T \\ m_i^n \boldsymbol{v}_i^n &= \sum_p w_{ip}^n m_p (\boldsymbol{v}_p^n + \boldsymbol{B}_p^n (\boldsymbol{D}_p^n)^{-1} (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)) \end{aligned}$$

```
#define IX(i,j) ((i)+(N+2)*(j))
#define SWAP(x0,x) {float * tmp=x0;x0=x;x=tmp;}
#define FOR_EACH_CELL for ( i=1 ; i<=N ; i++ ) { for ( j=1 ; j<=N ; j++ ) {
#define END_FOR }}
static float * u, * v, * u_prev, * v_prev;
static float * dens, * dens_prev;
```

仿真主程序

```
void sim_main(void) {
  get_force_source( dens_prev, u_prev, v_prev, w_prev );
  // 更新速度场
  vel_step ( N, u, v, w, u_prev, v_prev, w_prev, visc, dt );
  // 更新密度场
  dens_step ( N, dens, dens_prev, u, v, w, diff, dt );
}
void dens_step ( int N, float * x, float * x0,
                 float * u, float * v, float diff, float dt ) {
  add_source ( N, x, x0, dt );
  SWAP ( x0, x ); diffuse ( N, 0, x, x0, diff, dt );
  SWAP ( x0, x ); advect ( N, 0, x, x0, u, v, dt );
}
void vel_step ( int N, float * u, float * v, float * u0,
                float * v0, float visc, float dt ) {
  add_source ( N, u, u0, dt ); add_source ( N, v, v0, dt );
  SWAP ( u0, u ); diffuse ( N, 1, u, u0, visc, dt );
  SWAP ( v0, v ); diffuse ( N, 2, v, v0, visc, dt );
  project ( N, u, v, u0, v0 );
  SWAP ( u0, u ); SWAP ( v0, v );
  advect ( N, 1, u, u0, u0, v0, dt ); advect ( N, 2, v, v0, u0, v0, dt );
  project ( N, u, v, u0, v0 );
}
```

流体有从密度高的地方流向密度低地方的趋势, diffuse过程用于描述这一现象.

$$\begin{aligned} x(i,j) = x_0(i,j) + &a(x_0(i-1,j) + x_0(i+1,j) \\ &+ x_0(i,j-1) + x_0(i,j+1) - 4x_0(i,j)) \end{aligned}$$

然而这样写会在系数a较大时不稳定, 所以改为隐式写法

$$\begin{aligned} x_0(i,j) = x(i,j) - &a(x(i-1,j) + x(i+1,j) \\ &+ x(i,j-1) + x(i,j+1) - 4x(i,j)) \end{aligned}$$

即解线性系统, 矩阵系数, 利用Gauss-Seidel relaxation方式迭代求解.

```
void diffuse ( int N, int b, float * x, float * x0,
               float diff, float dt ) {
  float a=dt*diff*N*N;
  lin_solve ( N, b, x, x0, a, 1+4*a );
}
void lin_solve ( int N, int b, float * x, float * x0,
                 float a, float c ) {
  int i, j, k;
  for ( k=0 ; k<20 ; k++ ) {
    FOR_EACH_CELL
      x[IX(i,j)] = (x0[IX(i,j)] + a*(x[IX(i-1,j)]+
                    x[IX(i+1,j)]+x[IX(i,j-1)]+x[IX(i,j+1)]))/c;
    END_FOR
    set_bnd ( N, b, x );
  }
}
```

advect步骤, 假设待求网格(i,j)中心处的粒子, 前一时刻处于(i',j')中, 则(i,j)处的速度等于上一时刻(i',j')处的速度.

$$\boldsymbol{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\boldsymbol{v}}_i^{n+1}$$

$$\boldsymbol{B}_p^{n+1} = \sum_i w_{ip}^n \tilde{\boldsymbol{v}}_i^{n+1} (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)^T.$$

```
void advect ( int N, int b, float * d, float * d0,
              float * u, float * v, float dt ) {
  int i, j, i0, j0, i1, j1;
  float x, y, s0, t0, s1, t1, dt0;
  dt0 = dt*N;
  FOR_EACH_CELL
    x = i-dt0*u[IX(i,j)]; y = j-dt0*v[IX(i,j)];
    if (x<0.5f) x=0.5f; if (x>N+0.5f) x=N+0.5f; i0=(int)x; i1=i0+1;
    if (y<0.5f) y=0.5f; if (y>N+0.5f) y=N+0.5f; j0=(int)y; j1=j0+1;
    s1 = x-i0; s0 = 1-s1; t1 = y-j0; t0 = 1-t1;
    d[IX(i,j)] = s0*(t0*d0[IX(i0,j0)]+t1*d0[IX(i0,j1)])+
           s1*(t0*d0[IX(i1,j0)]+t1*d0[IX(i1,j1)]);
  END_FOR
  set_bnd ( N, b, d );
}
```

Projection过程要求泊松方程, 二维情况使用五点差分法

$$\nabla^2 \phi = f$$

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \boldsymbol{f}$$

$$\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\triangle x^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\triangle y^2} = f_{i,j}$$

边界上$\phi_{i,j}$为0?. 可以总结出一个线性方程$\boldsymbol{A\phi} = \boldsymbol{f}$, 同样通过Gauss-Seidel relaxation求解.

```
void project ( int N, float * u, float * v,
               float * p, float * div ) {
  int i, j;
  FOR_EACH_CELL
    div[IX(i,j)] = -0.5f*(u[IX(i+1,j)]-u[IX(i-1,j)]+v[IX(i,j+1)]-v[IX(i,j-1)])/N;
    p[IX(i,j)] = 0;
  END_FOR
  set_bnd ( N, 0, div ); set_bnd ( N, 0, p );
  lin_solve ( N, 0, p, div, 1, 4 );
  FOR_EACH_CELL
    u[IX(i,j)] -= 0.5f*N*(p[IX(i+1,j)]-p[IX(i-1,j)]);
    v[IX(i,j)] -= 0.5f*N*(p[IX(i,j+1)]-p[IX(i,j-1)]);
  END_FOR
  set_bnd ( N, 1, u ); set_bnd ( N, 2, v );
}
```

此外有两个辅助函数, 一个用于直接叠加场, 另外一个用于边界条件设置.

```
void add_source ( int N, float * x, float * s, float dt ) {
  int i, size=(N+2)*(N+2);
  for ( i=0 ; i<size ; i++ ) x[i] += dt*s[i];
}
void set_bnd ( int N, int b, float * x ) {
  int i;
  for ( i=1 ; i<=N ; i++ ) {
    x[IX(0  ,i)] = b==1 ? -x[IX(1,i)] : x[IX(1,i)];
    x[IX(N+1,i)] = b==1 ? -x[IX(N,i)] : x[IX(N,i)];
    x[IX(i,0  )] = b==2 ? -x[IX(i,1)] : x[IX(i,1)];
    x[IX(i,N+1)] = b==2 ? -x[IX(i,N)] : x[IX(i,N)];
  }
  x[IX(0  ,0  )] = 0.5f*(x[IX(1,0  )]+x[IX(0  ,1)]);
  x[IX(0  ,N+1)] = 0.5f*(x[IX(1,N+1)]+x[IX(0  ,N)]);
  x[IX(N+1,0  )] = 0.5f*(x[IX(N,0  )]+x[IX(N+1,1)]);
  x[IX(N+1,N+1)] = 0.5f*(x[IX(N,N+1)]+x[IX(N+1,N)]);
}
```

7 附录

7.1 Schur Complement

定义矩阵

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}$$

若$\boldsymbol{D}$可逆,$\boldsymbol{D}$在$\boldsymbol{M}$ 中的舒尔补为:

$$\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$$

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{D}^{-1} \end{pmatrix}$$

$$\boldsymbol{ML} = \begin{pmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{B}\boldsymbol{D}^{-1} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}$$

若$\boldsymbol{A}$可逆,$\boldsymbol{A}$在$\boldsymbol{M}$ 中的舒尔补为:

$$\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$$

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} \\ -\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{pmatrix}$$

$$\boldsymbol{TM} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}^{-1}\boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B} \end{pmatrix}$$

$$\boldsymbol{Ax} + \boldsymbol{By} = \boldsymbol{a}$$

$$\boldsymbol{Cx} + \boldsymbol{Dy} = \boldsymbol{b}$$

对第二个式子左乘$\boldsymbol{BD}^{-1}$可以得到

$$\boldsymbol{BD}^{-1}\boldsymbol{Cx}+\boldsymbol{By}=\boldsymbol{BD}^{-1}\boldsymbol{b}$$

综上可推出

$$(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C})\boldsymbol{x}=\boldsymbol{a}-\boldsymbol{BD}^{-1}\boldsymbol{b}$$

$$\boldsymbol{x}=(\boldsymbol{A}-\boldsymbol{BD}^{-1}\boldsymbol{C})^{-1}(\boldsymbol{a}-\boldsymbol{BD}^{-1}\boldsymbol{b})$$

同理, 对第一个式子左乘$\boldsymbol{CA}^{-1}$

$$\boldsymbol{Cx}+\boldsymbol{CA}^{-1}\boldsymbol{By}=\boldsymbol{CA}^{-1}\boldsymbol{a}$$

综上可以推出

$$(\boldsymbol{D}-\boldsymbol{CA}^{-1}\boldsymbol{B})\boldsymbol{y}=\boldsymbol{b}-\boldsymbol{CA}^{-1}\boldsymbol{a}$$

$$\boldsymbol{y}=(\boldsymbol{D}-\boldsymbol{CA}^{-1}\boldsymbol{B})^{-1}(\boldsymbol{b}-\boldsymbol{CA}^{-1}\boldsymbol{a})$$