Lab6 质点弹簧系统仿真

By PB17111585 张永停

一、实验内容

- 在给定的网格框架上完成作业,实现
 - 质点弹簧仿真模型的欧拉隐式方法
 - 质点弹簧仿真模型的加速模拟方法

二、算法描述

(一) 欧拉隐式方法

•

$$egin{aligned} oldsymbol{x}_{n+1} &= oldsymbol{x}_n + holdsymbol{v}_{n+1}, \ oldsymbol{v}_{n+1} &= oldsymbol{v}_n + holdsymbol{M}^{-1}(oldsymbol{f}_{int}(t_{n+1}) + oldsymbol{f}_{ext}) \end{aligned}$$

记

$$oldsymbol{y} = oldsymbol{x}_n + holdsymbol{v}_n + h^2oldsymbol{M}^{-1}oldsymbol{f}_{ext},$$
 (*)

则原问题转化为求解关于x的方程:

$$oldsymbol{g}(oldsymbol{x}) = oldsymbol{M}(oldsymbol{x} - oldsymbol{y}) - h^2 oldsymbol{f}_{int}(oldsymbol{x}) = 0,$$

利用牛顿法求解该方程,考虑到程序运行时间,至多迭代50次,若50次后还未收敛,直接计算下一帧

$$oldsymbol{x}^{(k+1)} = oldsymbol{x}^{(k)} - (
abla oldsymbol{g}(oldsymbol{x}^{(k)}))^{-1} oldsymbol{g}(oldsymbol{x}^{(k)}).$$

迭代初值选为 $x^{(0)} = y$.

迭代得到位移x后更新速度 $v_{n+1} = (x_{n+1} - x_n)/h$

• 关于($\nabla g(x^{(k)})$, $g(x^{(k)})$)可以看作n个3*3的矩阵对角排列,对每个点的梯度拼接即可设单个弹簧(端点为 x_1 , x_2),劲度系数为k,原长为l,有:

$$m{x}_1$$
所受弾力: $m{f}_1(m{x}_1,m{x}_2) = k(||m{x}_1-m{x}_2||-l)rac{m{x}_2-m{x}_1}{||m{x}_1-m{x}_2||},$ $m{x}_2$ 所受弾力: $m{f}_2(m{x}_1,m{x}_2) = -m{f}_1(m{x}_1,m{x}_2),$

对

$$oldsymbol{h}(oldsymbol{x}) = k(||oldsymbol{x}||-l)rac{-oldsymbol{x}}{||oldsymbol{x}||},$$

求导有

$$rac{doldsymbol{h}}{doldsymbol{x}} = k(rac{l}{||oldsymbol{x}||} - 1)oldsymbol{I} - kl||oldsymbol{x}||^{-3}oldsymbol{x}oldsymbol{x}^T.$$

带入弹力公式得:

$$rac{\partial oldsymbol{f}_1}{\partial oldsymbol{x}_1} = rac{\partial oldsymbol{h}(oldsymbol{x}_1 - oldsymbol{x}_2)}{\partial oldsymbol{x}_1} = k(rac{l}{||oldsymbol{r}||} - 1)oldsymbol{I} - kl||oldsymbol{r}||^{-3}oldsymbol{r}oldsymbol{r}^T,$$
其中 $oldsymbol{r} = oldsymbol{x}_1 - oldsymbol{x}_2, oldsymbol{I}$ 为单位阵

(二)加速方法

对于内力(为保守力)有:

$$oldsymbol{f}_{int}(x) = -
abla E(oldsymbol{x})$$

故对方程(*)的求解可以转为为一个最小化问题:

$$oldsymbol{x}_{n+1} = \min_{oldsymbol{x}} rac{1}{2} (oldsymbol{x} - oldsymbol{y})^T oldsymbol{M} (oldsymbol{x} - oldsymbol{y}) + h^2 E(oldsymbol{x})$$

同时对于弹簧的弹性势能可以描述为一个最小化问题:

$$rac{1}{2}k(||m{p}_1-m{p}_2||-r)^2 = rac{1}{2}k\min_{||m{d}||=r}||m{p}_1-m{p}_2-m{d}||^2,$$

从而原问题转化为:

$$oldsymbol{x}_{n+1} = \min_{oldsymbol{x}} rac{1}{2} oldsymbol{x}^T (oldsymbol{M} + h^2 oldsymbol{L}) oldsymbol{x} - h^2 oldsymbol{x}^T oldsymbol{J} oldsymbol{d} - oldsymbol{x}^T oldsymbol{M} oldsymbol{y}$$

其中

$$oldsymbol{U} = \{oldsymbol{d} = (oldsymbol{d}_1, oldsymbol{d}_2, \ldots, oldsymbol{d}_s), oldsymbol{d}_s \in R^3, ||oldsymbol{d}_i|| = l_i\}(l_i$$
为第 i 个弹簧原长 $),$

从而可以采用Global/Local对 x,d 迭代优化求得该优化问题的解:

$$oldsymbol{x}$$
优化:求解方程 $(oldsymbol{M}+h^2oldsymbol{L})oldsymbol{x}=h^2oldsymbol{J}oldsymbol{d}+oldsymbol{M}oldsymbol{y}+h^2oldsymbol{f}_{ext}$

$$m{d}$$
优化: $m{d}_i = l_i rac{m{p}_{i_1} - m{p}_{i_2}}{||m{p}_{i_1} - m{p}_{i_2}||}$ (这里 l_i 为第 i 个弹簧原长, $m{p}_{i_1}$, $m{p}_{i_2}$ 为其两端点),

由于L, M, J与时间无关,故可以在初始化的时候进行计算与分解,以在迭代时候节省时间

(三)约束

- 外力约束本次实验只考虑重力,是个常量
- 考虑真正的坐标,降低维数

将所有 \mathbf{n} 个质点的坐标列为列向量 $x \in R^{3n}$,将所有 \mathbf{m} 个自由质点坐标(无约束坐标)列为列向量 $x_f \in R^{3m}$,则两者关系:

$$egin{aligned} oldsymbol{x}_f &= oldsymbol{K} oldsymbol{x}, \ oldsymbol{x} &= oldsymbol{K}^T oldsymbol{x}_f + oldsymbol{b}, \end{aligned}$$

其中 $K \in \mathbb{R}^{3m \times 3n}$ 为单位阵删去约束坐标序号对应行所得的稀疏矩阵,b 为与约束位移有关的向量,计算为 $b = x - K^T K x$,若约束为固定质点则 b 为常量。由此我们将原本的关于 x 的优化问题转化为对 x_f 的优化问题: 欧拉隐式方法中求解方程为:

$$egin{aligned} oldsymbol{g}_1(oldsymbol{x}_f) &= K(oldsymbol{M}(oldsymbol{x} - oldsymbol{y}) - h^2 oldsymbol{f}_{int}(oldsymbol{x})) = 0, \ & ext{$$
梯度: $abla_{x_f} oldsymbol{g}_1(oldsymbol{x}_f) = K
abla_x oldsymbol{g}(oldsymbol{x}) K^T, \end{aligned}$

加速方法中优化问题中x 迭代步骤转化为求解关于 x_f 的方程:

$$K(\boldsymbol{M} + h^2 \boldsymbol{L})K^T \boldsymbol{x}_f = K(h^2 \boldsymbol{J} \boldsymbol{d} + \boldsymbol{M} \boldsymbol{y} - (\boldsymbol{M} + h^2 \boldsymbol{L}) \boldsymbol{b})$$

三、代码框架

(一) 隐式方法

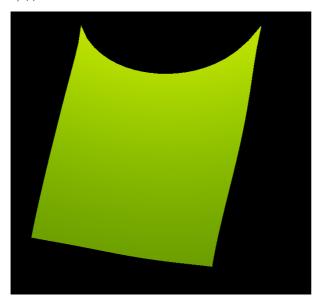
```
private:
 1
 2
           double g;
 3
           std::vector<double> force_ext;
 4
           std::vector<double> force_int;
 5
           std::vector<double> mass;
 6
           std::vector<double> 1;
                                              //弹簧原长
 7
           std::vector<double> xk:
                                              //迭代过程中的$x^{n}$
8
9
           std::vector<double> x;
           std::vector<double> xk_1;
10
                                             //迭代过程中的$x^{n+1}$
           std::vector<double> b;
                                              //约束里的b
11
12
           std::vector<double> gx;
                                              //g(xk)
13
14
           Eigen::MatrixXd gx_m;
                                              //矩阵形式的g(xk)
15
           Eigen::MatrixXd K;
                                             //约束矩阵K
           Eigen::MatrixXd inverG;
                                              //$\nabla g(x)$的逆
16
           Eigen::SparseMatrix<double> diff;
17
                                             18
           std::set<int> fix;
                                              //固定的点
19
20
21
    public:
22
           void SetInitG();
                                              //重力初始化
23
           void CacForce();
                                              //内力(即弹力)计算
           void GetSet();
                                              //从fixed_id转化到fix
24
25
           void CacX();
                                              //牛顿迭代计算坐标
26
           void CacGX();
                                              //计算g(xk)
27
           void Cack();
                                              //计算转换矩阵
28
           void CacGxM();
           void CacDiff():
                                              //计算\nabla g(x)
29
30
           bool isconv();
                                              //牛顿迭代收敛判断
31
           void CacV();
                                              //计算速度
32
           void UpdateX();
                                              //将xf转化成x
33
           void UpdatePos();
                                              //更新坐标
34
           void SetFix();
                                              //固定点
```

(二)加速方法

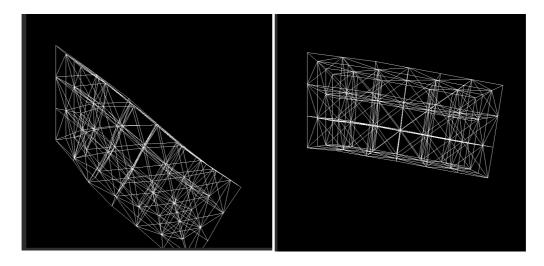
```
1
    private:
 2
            int iter;
                                                //迭代次数
 3
            std::vector<double> d;
            std::vector<double> y_;
 4
 5
            std::vector<double> xx;
                                                //存储x_{k-1}
 6
            Eigen::MatrixXd J;
 7
            Eigen::MatrixXd L;
8
            Eigen::MatrixXd M;
9
            Eigen::SparseMatrix<double> A;
10
                                               //求解的方程的左端
            Eigen::SimplicialLLT<Eigen::SparseMatrix<double> > LLT_;
11
12
13
            bool isfast;
                                                //是否采用fast的方法
14
15
    public:
```

四、实验结果

布料



• 长方体



左: k=1e3

右:k=1e4

• 具体见视频,视频命名物体_模拟方法 (euler/fast)_stiff

五、实验总结

- 本次实验的一些细节
 - 。 在求 $(\nabla g(x))^{-1}$,相当于求 $\nabla g(x)$ A = I_n ,从而调用Eigen的LU解方程得到逆
 - o 提前分解加速计算
 - o Release真的比debug快很多,刚开始用debug不动以为是自己代码有问题调了好久
 - o 牛顿迭代可能会不收敛,故应该控制迭代次数避免一直循环

- o 对弹力求导的时候是用的 $F = k(||x||-l)\frac{-x}{||x||}$ 来求导的,不是 $F = k_s(\frac{||x||}{l}-1)\frac{-x}{||x||}$,这里应和前面计算内力的时候对应
- o 部分类似代码复制的时候记得改变量。。。比如CacJ,CacL,相似的过程,我最开始复制的时候忘了改最后的变量,导致J是0,找了好久
- 如果需要调用SetLeftFix(),请注释掉SetFix()的内容(SetFix()主要是固定正方形的两个项点的)
- 本次实验debug花了超级多的时间,其中隐式的调试过程持续了整整两天,主要是当时写的时候觉得不难,代码写的太乱了,犯了好多低级错误,下次争取白天写代码以及把代码写规范点(