PS：这节课感觉太难了 就不一步步翻译说明了 只记录重难点（可以先跳到最后看总结）

**一、开始**

命令行启动：a1.exe r 0.01

**二、需求总结**

1. 欧拉法和梯形法的解算器，以及简单的粒子系统。

用于求解常微分方程，目前先进行一阶系统的测试（即一阶常微分方程，只用到了牛顿第一定律）

2. 钟摆。

实现二阶系统，一个简单的钟摆，包括两个粒子和连接他们的一根弹簧。这需要实现三种力，重力，粘滞阻力和弹簧力，这些力后续都会用于布料模拟。

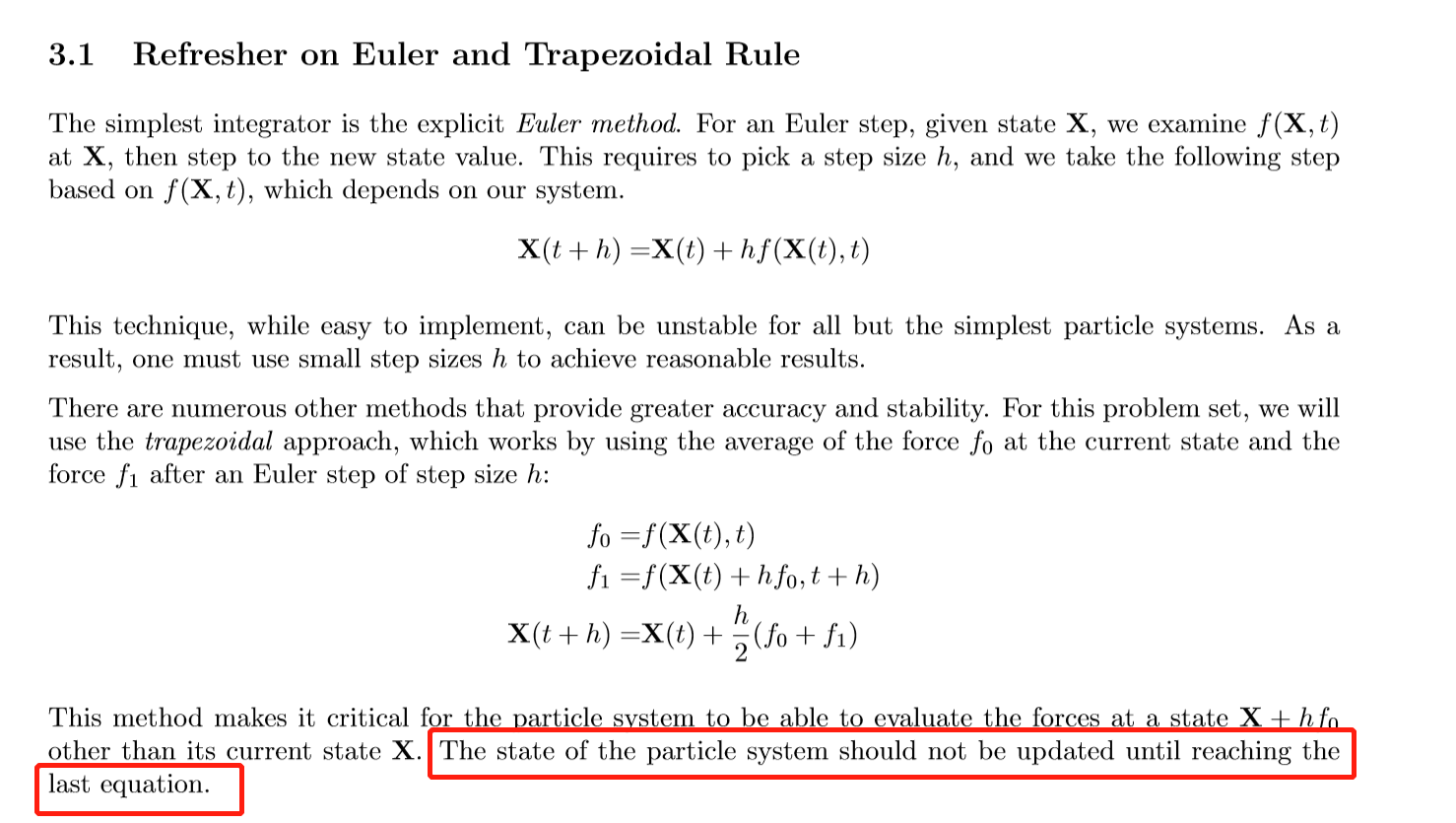
3. 多钟摆和RK4。

扩展单个钟摆成为包括4个粒子的钟摆。实现最后一个积分器，Runge-Kutta4，这可以允许你使用更多的粒子和更大的时间步长。

4. cloth。

最终，使用弹簧来装配一张布，至少是8x8的粒子网格，需要实现结构、剪切和弯曲弹簧。

**三、时间积分器(tiemsteppter.cpp)**



粒子系统状态不应该更新，知道最后一个方程？

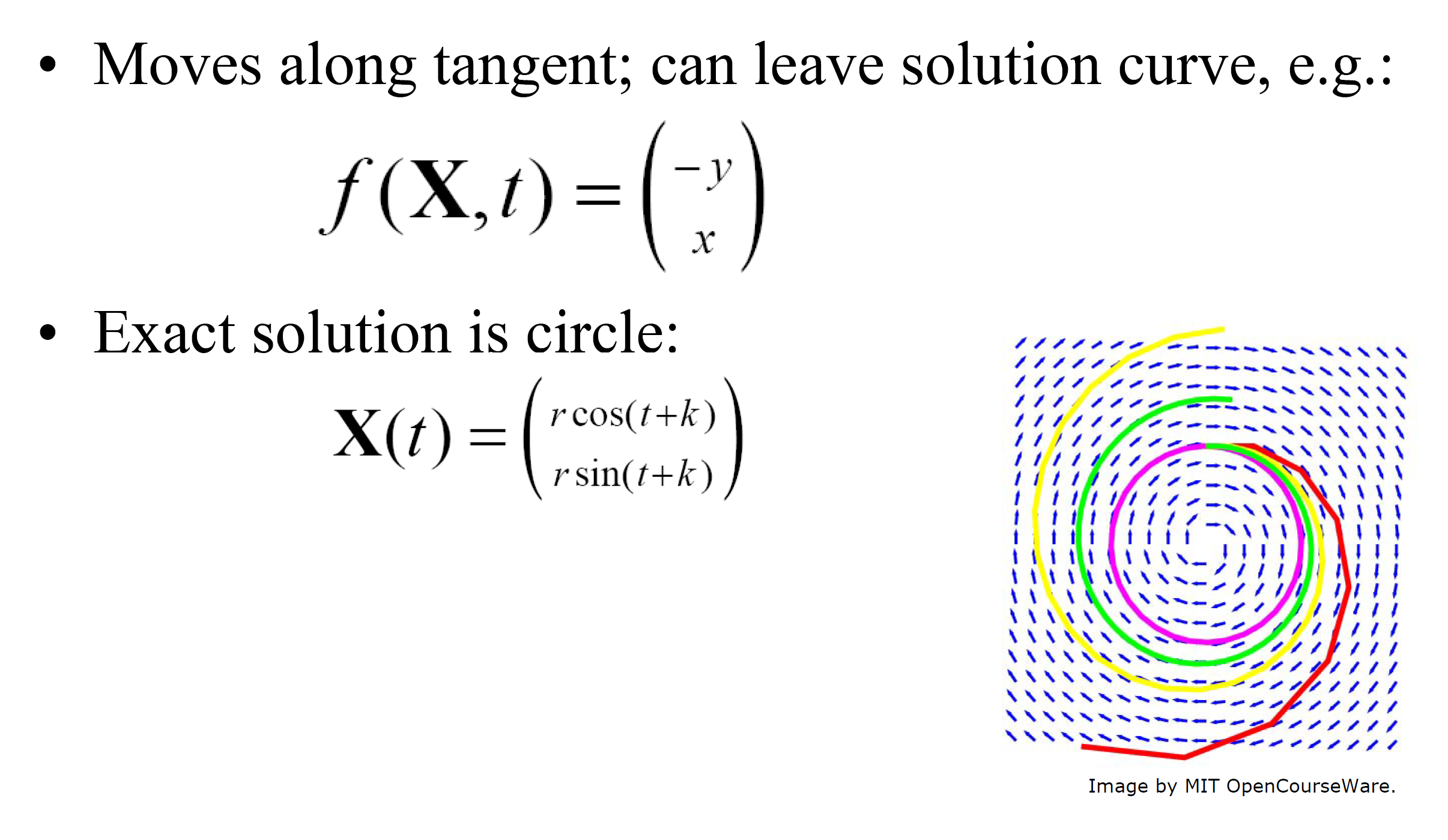
3.2 简单的例子

要实现欧拉法和梯形法。使用粒子系统的方法**evalF**来计算当前状态的导数。

看一看抽象类ParticleSystem.

实现**SimpleSystem**中的**evalF**方法，传递系统给**takeStep**方法，并小心地更新系统状态。

理解：想象粒子有一个位置和速度的向量组，表示当前状态。如果想要求得粒子下一时刻的位置，就需要求得此刻粒子的速度和加速度，使用欧拉法算得粒子下一时刻的位置即为当前位置加上速度乘以时间间隔，而到了下一时刻，速度也变为速度加上时间间隔乘以加速度；这样就可以通过获得向量组的导数向量组，来计算得到下一时刻粒子的状态。



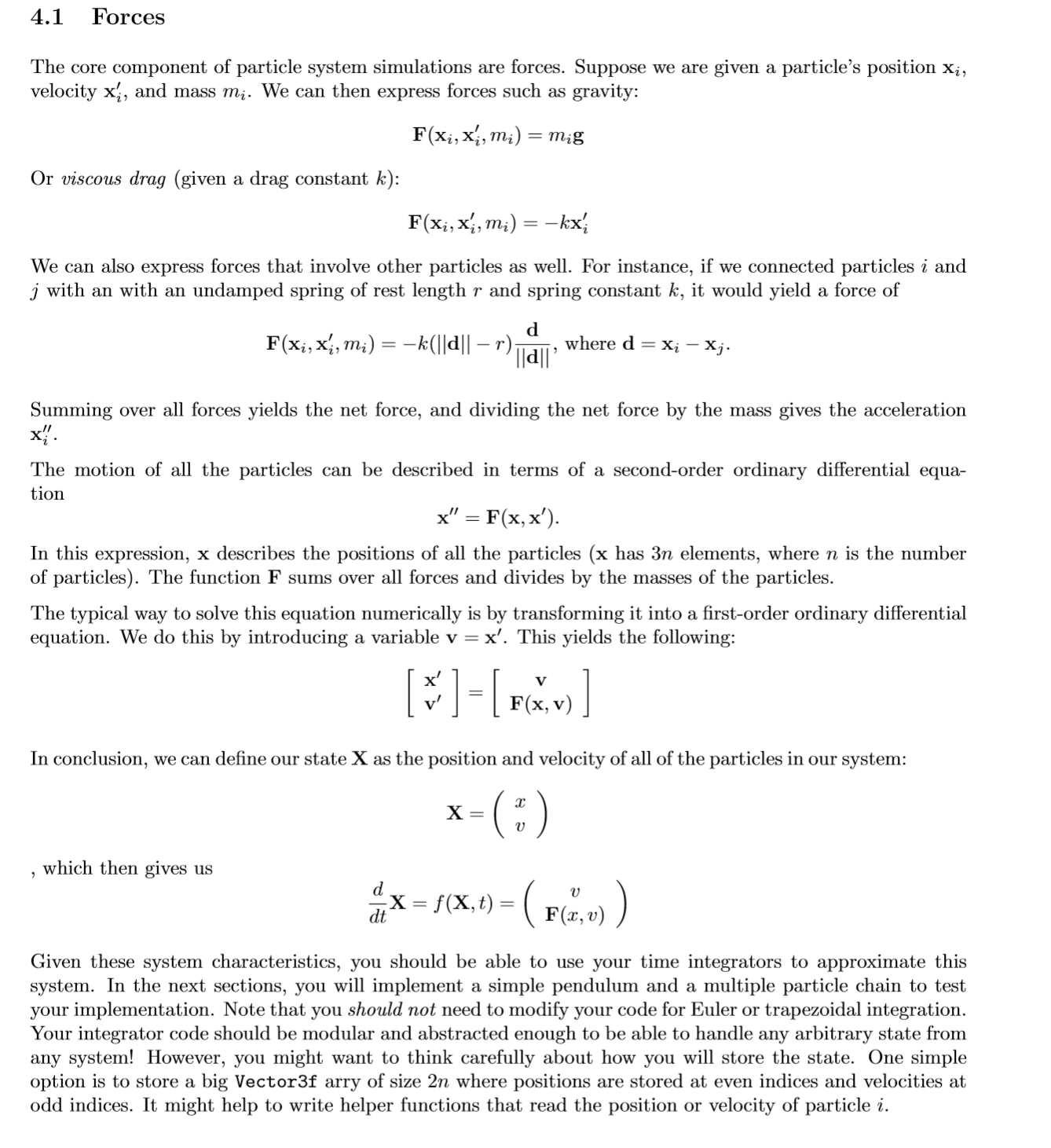
上面的公式是说，X(t) = { x = rcos(t+k), y = rsin(t+k)} k是任意常数 对于三角函数来说 其实不影响

那么X'(t)，即f(X,t) = { x' = -rsin(t+k) = -y , y' = rcos(t+k) = x}

这是方程的精确解，如果沿着切线走，就会逐渐远离解曲线。

在积分器中的evalF方法和你实现的方法，都不应该改变系统的粒子状态，它只是接收了一个粒子状态，并返回粒子状态的导数状态，积分器中的方法会在每一个步骤中自动改变粒子状态。

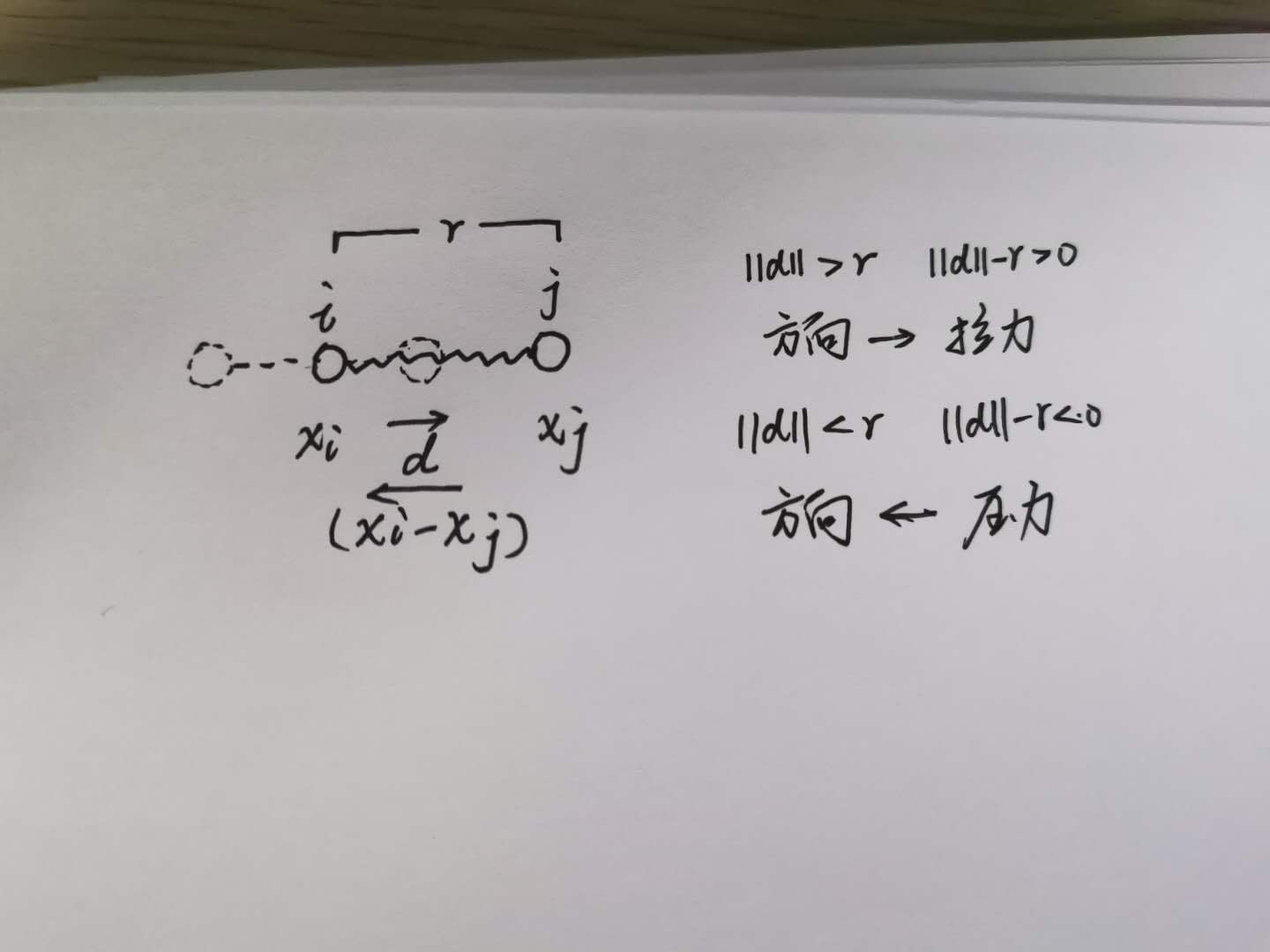
4. 物理模拟



0需要实现一些力，包括重力、粘滞阻力和弹簧力；

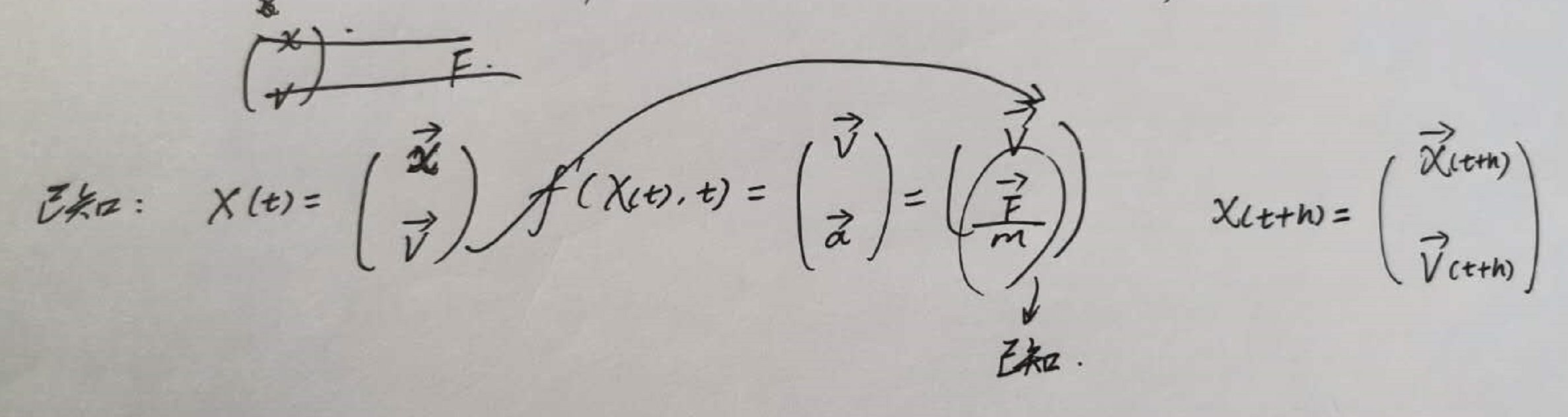
重力就是f1 = mig; 阻力就是 f2 = -kv;(常量系数，跟速度方向相反)；弹簧力：胡克定律；

F方法，计算合力并除以质点的质量；



1. 钟摆系统包括了一个固定的粒子和另外一个由弹簧连接的粒子构成。

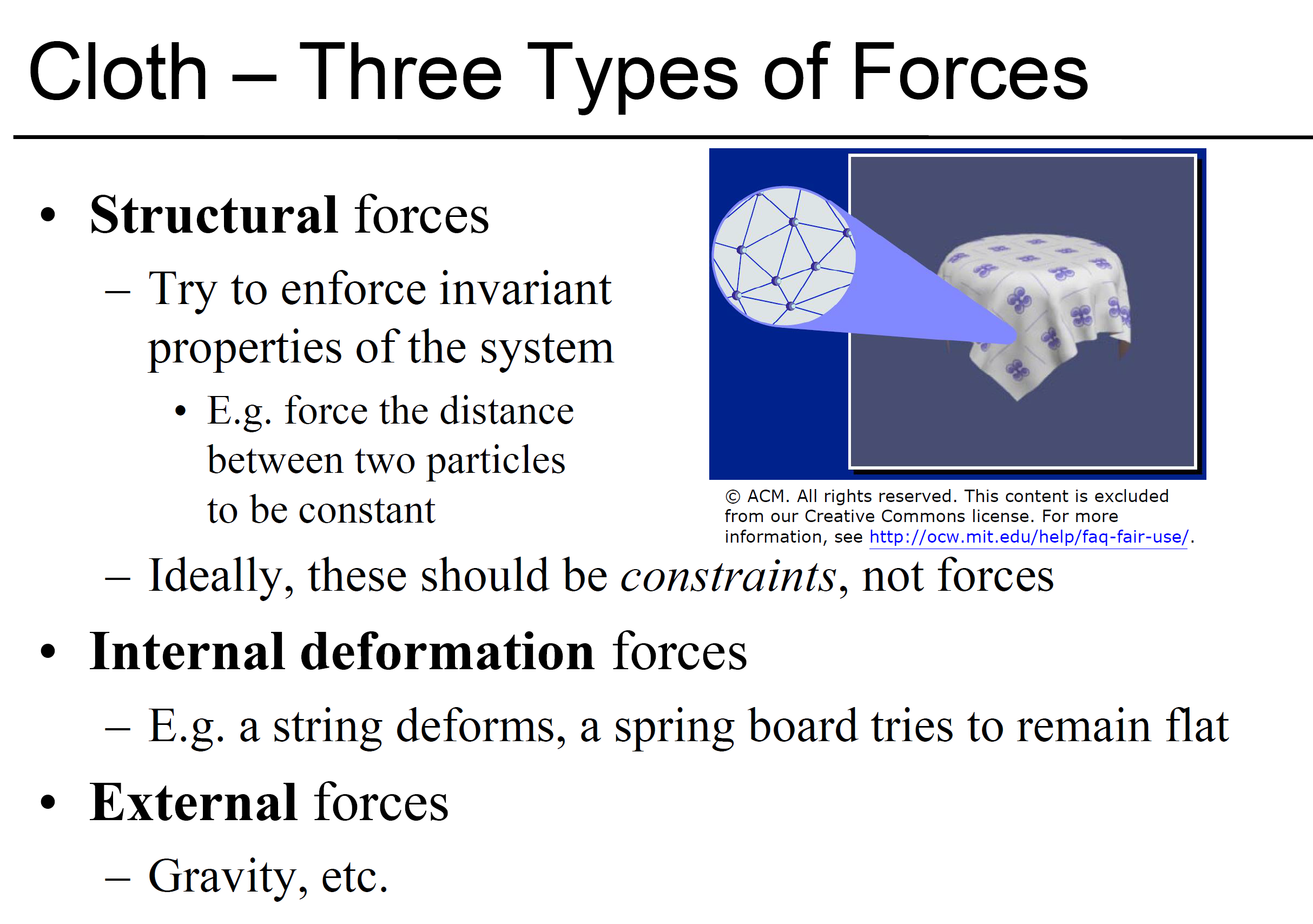
2. 状态包括位置和速度，那么对状态求导，就得到速度和加速度的向量组，其中速度其实在状态那部分就可以获得，速度的求导，需要用到合力以及质量，通过牛顿第二定律来算：



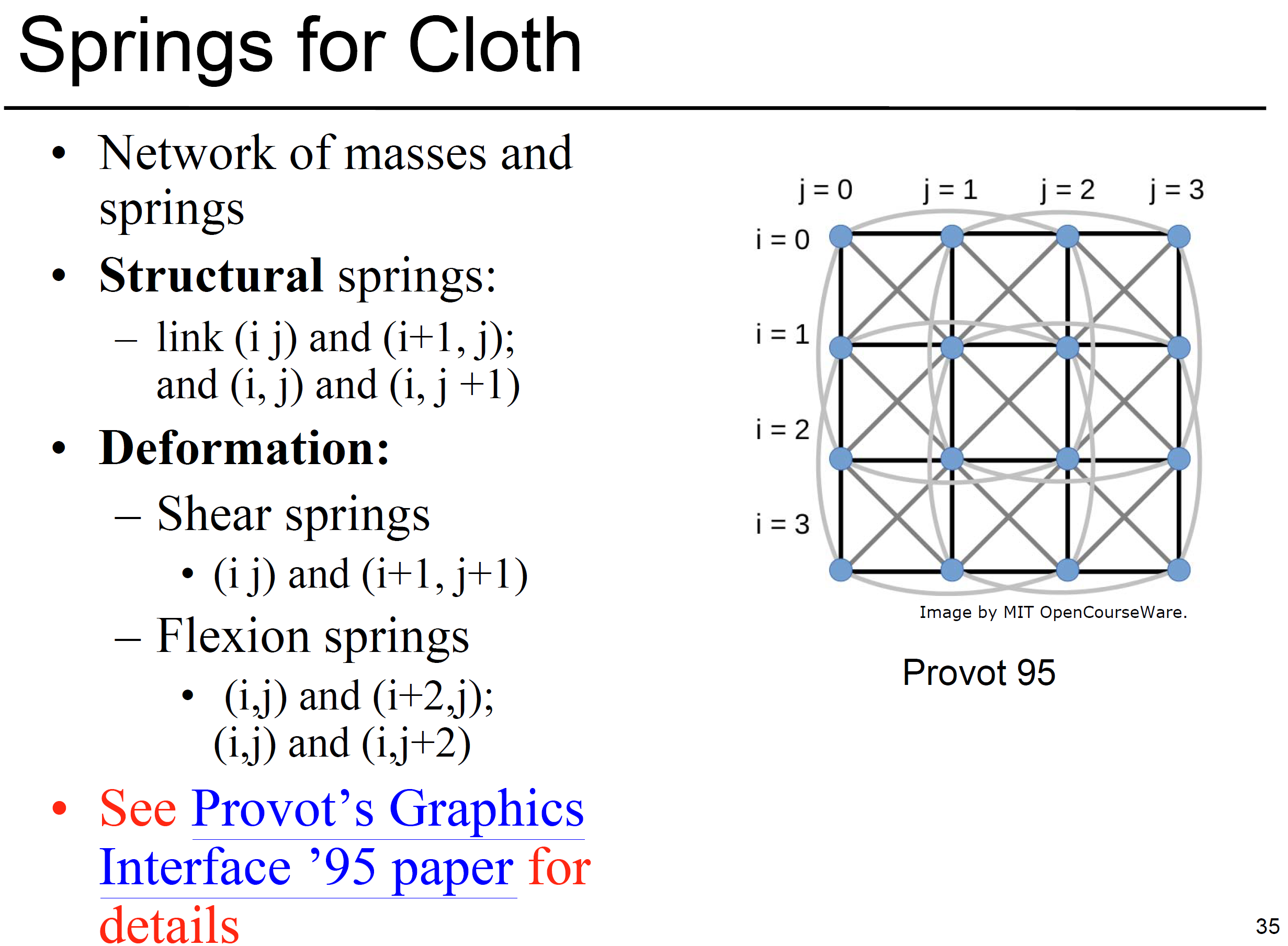
RK4：

<https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods>

5. 实现布料效果



有三种力，结构力，内部形变力，外部力。



注意弹簧连接关系，和两个端点，flexion是弯曲的意思。

也就是说一个质点收到了三种内力，结构力（相邻），剪切力（对角）和弯曲力（间隔相邻），最多共计12个质点的作用力，位于边界条件的情况要特殊处理。

刚度的设置是个问题。

结构力应该最大，剪切力其次，弯曲力再其次。

暂定25，5，2试试；

如果固定左上角和右上角两个点，那么意为着这两个点处的合力为零，在外力的作用下保持了系统的平衡。

**PS：后记**

这个作业刚开始看的时候感觉应该是最难的，所以也留到了最后才做。但是真正做起来的时候发现，却出乎意料地顺利，整个过程没有出现太多的bug。在此再次整理一下整个物理模拟系统的逻辑：

整个物理模拟系统主要由两部分组成：ParticleSystem和TimeStepper

一、ParticleSystem

粒子系统是一个抽象类，可以具体实现它的继承类，比如一个钟摆系统、一个布料系统，在这里系统里最核心的有两个点，1是如何设计或者说如何表达粒子系统的运动，比如用一阶的常微分方程来表示，那就只有一个位置，其导数就是速度；二阶的常微分方程有位置和速度，求导是速度和加速度，这些属性都表示了当前粒子系统的状态state；2是evalF方法，用来对当前状态进行求导，比如一阶，可以求得粒子系统中所有粒子的速度(向量），二阶可以求得所有粒子的速度和加速度。

提醒：其实大家不必觉得常微分方程是个很难很高深的东西(至少对于该作业而言)，它只是表达了，当我们不知道粒子的运动规律(方程)，但是又知道它的导数(或更高阶导数）的时候，如何求得它的运动方程。

比如，对于一个单一质点(只受到重力)，我知道它的起始位置，速度，我想知道它任意时刻的位置，那么我就可以绘制出它的运动轨迹，但是它任意时刻的位置，这个方程怎么求呢？我们还知道，其实这个质点还受到了重力，那么相当于我们知道了它的加速度(由牛顿第二定律a = F / m），即我们知道了它位移的二阶导数。其实这里即使不需要积分知识，仅依靠中学物理知识我们也知道，s = 1/2 \* g \* t²；但其实这就是常微分方程的解！相当于我们知道了粒子的起始状态，s0和v0，并且也能知道它的导数状态方程，s0' = v0, v0' = a0 = mg / m = g；导数状态方程的速度和加速度都是已知的（注意这部分其实就是在evalF里面计算的(计算合力，然后计算加速度)；有了这个导数状态方程，我们就可以使用积分方法去计算粒子系统的下一个状态了。见TimerStepper，比如最简单的前向欧拉法，把当前点的导数当作整个步长step的导数，那么下一个状态的位置，s1 = s0 + h \* v0，速度v1 = v0 + h \* a0。

二、TimerStepper

其实这就是一个积分迭代器，取定某一个步长，采取某一种积分方法（比如欧拉法、梯形法、中点法），来获得在经过这个步长之后粒子系统的下一个状态，这个状态可能包括位置、速度等，这样，经过每一步的迭代，我们就得到每一个时间点上粒子的位置/速度信息，就可以根据这些信息绘制出所有粒子的运动轨迹了。

这个积分迭代器也是一个抽象类，可以由不同的实现积分方法来继承，区别就是精度和稳定性的差异。基本思想就是，获取粒子系统当前状态，根据其系统内部的定义和求导方法(evalF)来求导得到导数状态方程，然后再通过积分迭代器的积分方法，来求出下一个状态的方程。外部渲染循环，就会根据每一个时间点上的状态去绘制粒子系统。

其他：

在最后模拟布料系统这一块，看起来有些复杂，其实只要耐心在图上画出来，找出各个两两粒子之间的弹簧的对应关系(数字索引)，计算合力，就可以了。加油。