统计力学第五次编程实践作业

小组工作主要完成人：

姓名：杨阳	姓名：黄裕函
学号：150*******	学号：150*******
专业：流体力学	专业：流体力学

1.问题题目：在外势场$V^{ext}(\mathrm{r})=\frac{1}{2}r$的势阱中有10个粒子，粒子的温度$T_{ensemble}=1$.相互作用为归一化的Leonard-Jones 6-12势，热浴方法采用标准的全局Nose-Hover热浴。问题：

（1）是否仍然有速度v的Maxwell分布，和势能的$\exp\left(-\frac{V}{T}\right)$分布。

（2）平衡后说明$\rho(\mathrm{u})$和$\rho(r)$是独立的。

（3）利用virial展开定理计算P。

2.基本概念和理论简述

（1）Verlet时间积分方法

在时刻t向前和向后做Taylor展开：

$$r_i(t+\delta t)=r_i(t)+\dot{r}_i(t)\delta t+\frac{1}{2}\ddot{r}_i(t)\delta t^2+O(\delta t^3)$$

$$r_i(t-\delta t)=r_i(t)-\dot{r}_i(t)\delta t+\frac{1}{2}\ddot{r}_i(t)\delta t^2+O(\delta t^3)$$

两式相加得到：

$$r_i(t+\delta t)+r_i(t-\delta t)=2r_i(t)+\ddot{r}_i(t)\delta t^2+O(\delta t^4)$$

整理后得到Verlet时间数值积分公式：

$$r_i(t+\delta t)=2r_i(t)-r_i(t-\delta t)+\ddot{r}_i(t)\delta t^2+O(\delta t^4)$$

其中的加速度由粒子受到的力决定：

$$\ddot{r}_i(t)=\frac{f_i(t)}{m_i}$$

因为Verlet算法采用了类似中心差分的方式，因为得到的精度为四阶。同时Verlet算法是保辛的算法可以基本保证在计算过程中系统哈密顿量守恒。

粒子速度中心差分格式：

$$\dot{r}_i(t)=\frac{r_i(t+\delta t)-r_i(t-\delta t)}{2\delta t}+O(\delta t^2)$$

由于速度要额外采用中心差分得到，因为 Verlet 算法的速度计算精度不高只有 2 阶。

本文的编程模拟中将采用上述的时间积分格式。

（2）全局 Nose-Hover 恒温器

在 Suchi-Nose（1984）年的文章中引入了下面所述的哈密顿量：

$$H_{Nose}=V(q_j)+\sum_i\frac{p_i^{\ 2}}{2m_is^2}+(X+1)k_BT+\frac{p_s^{\ 2}}{2Q}$$

因此对应的正则方程为：

$$\begin{cases}\dfrac{d}{dt}q_i=\dfrac{\partial H_{Nose}}{\partial p_i}\\ \dfrac{d}{dt}p_i=-\dfrac{\partial H_{Nose}}{\partial q_i}\end{cases}\qquad\begin{cases}\dfrac{d}{dt}s=\dfrac{\partial H_{Nose}}{\partial p_s}\\ \dfrac{d}{dt}p_s=\dfrac{\partial H_{Nose}}{\partial s}\end{cases}$$

前面一组为力学状态变量，后面的为额外的恒温器状态变量。将上面的形式展开后得到：

$$\begin{cases}\dfrac{d}{dt}q_i=\dfrac{p_i}{m_is}\\ \dfrac{d}{dt}p_i=f_i(q_j)\end{cases}\qquad\begin{cases}\dfrac{d}{dt}s=\dfrac{p_s}{Q}\\ \dfrac{d}{dt}p_s=\sum_i\dfrac{p_i^{\ 2}}{ms^3}-(X+1)\dfrac{k_BT}{s}\end{cases}$$

因为在 Nose-Hover 恒温器中引入的变量 s 是时间标度因子，因此有：

$$dt=sdt_{old}$$

因此

$$\begin{cases}\dfrac{d}{dt}q_i=\dfrac{p_i}{m_i}\\ \dfrac{d}{dt}p_i=sf_i(q_j)\end{cases}\qquad\begin{cases}\dfrac{d}{dt}s=\dfrac{sp_s}{Q}\\ \dfrac{d}{dt}p_s=\sum_i\dfrac{p_i^{\ 2}}{ms^2}-(X+1)k_BT\end{cases}$$

对 $\dfrac{d}{dt}q_i$ 再求一次时间导数

$$\frac{d}{dt}\left(\frac{d}{dt}q_i\right)=\frac{d}{dt}\left(\frac{\dot{p}_i}{m_is}\right)-\frac{p_i}{m_is}\frac{\dot{s}}{s}$$

整理后得到：

$$\ddot{q}_i = \frac{f_i}{m_i} - \frac{p_s}{Q}\dot{q}_i$$

$$\ddot{q}_i = \frac{f_i}{m_i} - \varsigma\dot{q}_i$$

相当于引入了一个线性阻尼力。而这个阻尼的动力学方程：

$$\dot{\varsigma} = \frac{\dot{p}_s}{Q} = \frac{1}{Q}\left[\sum_i m\dot{q}_i^{\,2} - (X+1)k_B T\right]$$

重新整理并用矢径表示全局 Nose-Hover 恒温器动力学方程组：

$$\ddot{r}_i = \frac{f_i}{m_i} - \varsigma\dot{r}_i$$

$$\dot{\varsigma} = \frac{1}{Q}\left[\sum_i m\dot{r}_i^{2} - (X+1)k_B T\right]$$

在我们的模拟中还要对上面的公式进行更进一步的变化，下面阐述模拟中对于全局 Nose-Hover 恒温器采用的动力学方程组以及时间积分方法：

$$\ddot{r}_i = \frac{f_i}{m_i} - \varsigma\dot{r}_i$$
$$\dot{\varsigma} = \frac{1}{Q}\left[\sum_i m\dot{r}_i^{2} - N_f k_B T\right]$$

其中的 N_f 为系统总的自由度数目。由于我们模拟的时候是采用的无量纲归一化的分子动力学模拟，因而： $k_B = 1$， $m_i = 1$。我们模拟的时候采用的最终公式为：

$$\ddot{r}_i = f_i - \varsigma\dot{r}_i$$
$$\dot{\varsigma} = \frac{1}{Q}\left(\sum_i \dot{r}_i^{2} - N_f T\right)$$

在与速度 Verlet 时间数值积分方法的偶联中，对阻尼系数的动力学方程采用显式 Euler 积分的数值积分方式，显式 Euler 积分的问题是精度较低，同时也不能保证 Nose-Hover 哈密顿量的守恒，但是这个不精确正好相当于给系统的扰动。同时我们注意到如果对阻尼系数用其他更高阶的离散格式进行时间积分那么求解过程就会变为隐式求解需要求解系统粒子数目数量的非线性方程组，可以利用 Newton-Rapson 类的切线法，也可以利用梯度最速下降类算法求解，但是无论如何隐式求解都会带来巨大的计算量。将高阶精度时间积分带来的隐式求解转换为显示求解也有一定的方法，比如利用刘维尔算子得到的显式高阶时间可逆算法。

本文采用显式 Euler 时间积分，其离散格式如下：

$$\varsigma(t+\delta t)=\varsigma(t)+\frac{1}{Q}\left[\sum_i m\dot{r}_i^2-N_f k_B T\right]\delta t$$

考虑到模拟的时候是无量纲归一化的，因此：

$$\varsigma(t+\delta t)=\varsigma(t)+\frac{1}{Q}\left(\sum_i \dot{r}_i^2-N_f T\right)\delta t$$

全局 Nose-hover 恒温器其本质相当于对粒子系统的温度进行反馈积分控制，积分控制器的比例参数由惯量 Q 来调节，因此从这里就知道 Q 比较大的时候控制系统反应比较舒缓一些，温度震荡速度也会平缓一些，Q 比较小的时候控制系统反应会比较大，因此温度震荡速度也会剧烈写。

（3）加速度的计算

对于前面时间积分公式中涉及到的加速度或力进行下面的阐述：

对于保守力对应的加速度可以利用粒子场势的负梯度给出

$$\ddot{r}_i(t)=-\frac{1}{m_i}\sum_{j=1}^{n}\nabla U(r_{ij})$$

其中 $r=|r_i-r_j|=\sqrt{(x_i-x_j)^2}$，$U=U_{exter}(\vec{r}_i)+U_{inter}(r_{ij})$。

外势场由 $U_{exter}(\vec{r}_i)=\frac{1}{2}r_i^2$ 给出。对应的粒子间势函数由无量纲归一化的 Lenard-Jones 6-12 势给出：

$$U_{inter}=4\left[\left(\frac{1}{r}\right)^{12}-\left(\frac{1}{r}\right)^{6}\right]$$

而对应的作用力为：

$$f_{ij}=-\nabla U_{inter}(r_{ij})==-24\frac{r_i-r_j}{|r_i-r_j|}\left[2\left(\frac{1}{|r_i-r_j|}\right)^{13}-\left(\frac{1}{|r_i-r_j|}\right)^{7}\right]$$

$$\ddot{r}_i=\frac{1}{m_i}f_i=\sum_{j=1,j\neq i}^{N}f_{ij}$$

（4）各态遍历假设

在系统达到混沌状态后，粒子能够搜索整个相空间，这个时候各态遍历假设可以得到较好的满足。因此一些量的系综平均等于时间平均等于空间平均。不失

一般性这里采用时间平均来近似系综平均。也就是对时间序列进行统计分析。

各态遍历的相轨迹有下面的特点

$$\Gamma\left(\mathrm{t}=0\right)=\Gamma\left(t=\infty\right)$$

可以对分布函数的密度做下面的估计：

$$\hat{f}_i\left(x\right)=\frac{f_i}{h_i}=\frac{n_i}{Nh_i}$$

其中的h_i为窗宽，f_i为在窗宽内的频率，N为总样本数量，n_i为落在窗宽内的样本数量。这个公式也就是说明我们可以用经验分布密度函数近似求得分布密度函数。这也是本文采用的方法。对于其它量的统计平均也采用时间平均的方式：

$$\left\langle X\right\rangle=\lim_{T\to\infty}\frac{1}{T}\int_0^T X\left(t\right)dt$$

由于时间是离散的，上面的公式又可近似为：

$$\left\langle X\right\rangle=\lim_{N\to\infty}\sum_{i=1}^{N}\frac{1}{N}X_i$$

（5）维里定理

克劳修斯维里函数：

$$W^{total}\left(r_i\right)=\sum_i r_i\cdot f_i^{total}$$

在各态遍历假设下的系综平均：

$$\left\langle W^{total}\right\rangle=\lim_{T\to\infty}\frac{1}{T}\int_0^T\sum_i r_i\left(t\right)\cdot m_i\ddot{r}_i dt$$

利用分部积分得到：

$$\left\langle W^{total}\right\rangle=\lim_{T\to\infty}\frac{r_i\left(T\right)\dot{r}_i\left(T\right)-r_i\left(0\right)\dot{r}_i\left(0\right)}{T}-\lim_{T\to\infty}\frac{1}{T}\int_0^T\sum_i m_i\dot{r}_i^2\left(t\right)dt$$

在各态遍历假设下上式右侧最前面哪项为零，而后面一项可以看出这项与系统平均动能也就是温度有关。整理后得到：

$$\left\langle\sum_i r_i\cdot f_i^{total}\right\rangle=-\lim_{T\to\infty}\frac{1}{T}\int_0^T\sum_i m_i\dot{r}_i^2\left(t\right)dt=-N_f k_B T$$

而作用在分子上的总力可以分解为内力和外力，再考虑到内积的线性性质，有

$$\left\langle\sum_i r_i\cdot f_i^{total}\right\rangle=\left\langle\sum_i r_i\cdot f_i^{external}\right\rangle+\left\langle\sum_i r_i\cdot f_i^{\mathrm{int}ernal}\right\rangle$$

$$\left\langle \sum_i r_i \cdot f_i^{external} \right\rangle = -p\oint r \cdot dS = -p\int \nabla \cdot r dV$$

我们的问题是二维问题，对上面公式整理后得到压强的计算公式：

$$p = \frac{Nk_BT}{V} + \frac{1}{2V}\left\langle \sum_i r_i \cdot f_i^{internal} \right\rangle$$

上式中的 N 为系统粒子数，V 为系统边界内包含的体积。

3. 算法流程

（1）状态初始化

初始时刻粒子速度为零并且分布在半径 $r=1$ 的圆周上, 初始状态无阻尼。 $r_i(0)=g_i,\quad \dot{r}_i(0)=0,\quad \varsigma(0)=0,\ Q=0.5$

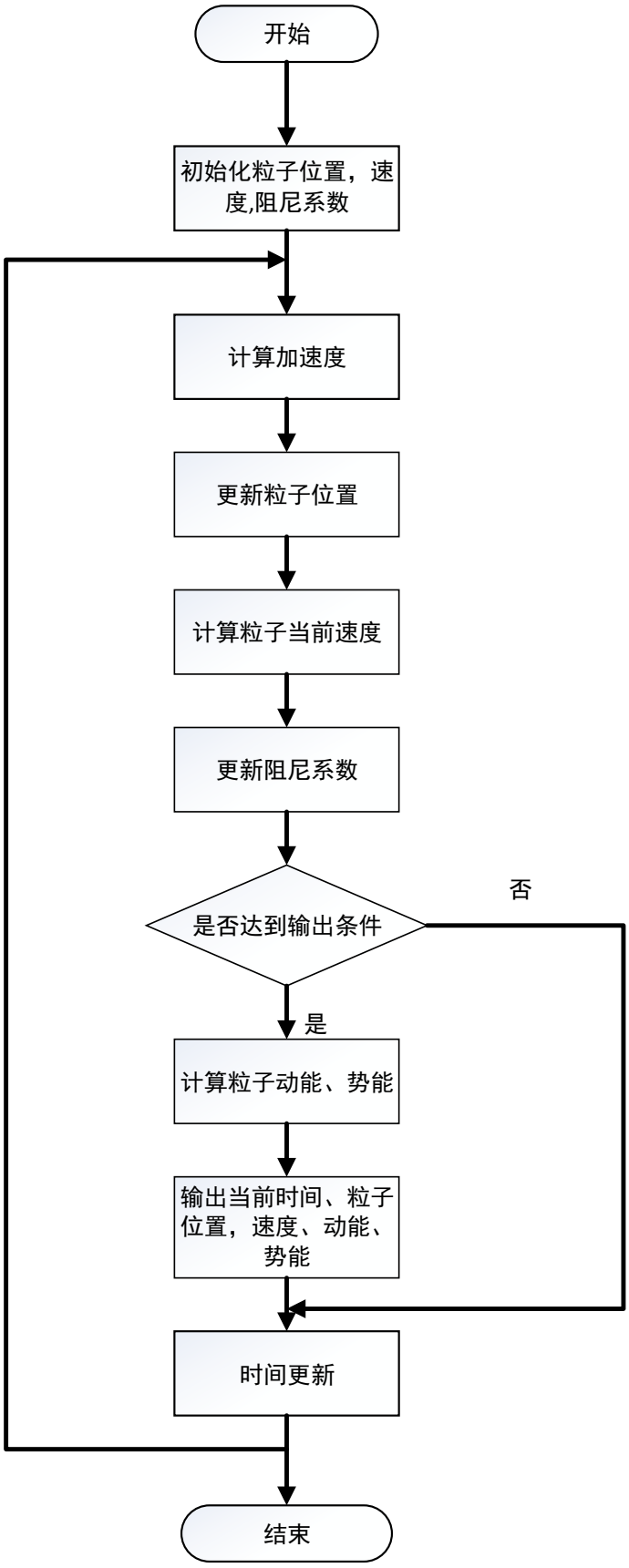
（2）启动步

1. 计算加速度： $\ddot{r}_i(t)=f_i-\varsigma\dot{r}_i$
2. 更新位置： $r_i(t+\delta t)=r_i(t)+\dot{r}_i(t)\delta t+\frac{1}{2}\ddot{r}_i(t)\delta t^2$
3. 更新速度： $\dot{r}_i(t+\delta t)=\dot{r}_i(t)+\ddot{r}_i(t)\delta t$
4. 更新阻尼： $\varsigma(t+\delta t)=\varsigma(t)+\frac{1}{Q}\left(\sum_i \dot{r}_i^2-N_fT\right)\delta t$

（3）时间推进步

1. 计算加速度： $\ddot{r}_i(t)=f_i-\varsigma\dot{r}_i$
2. 更新粒子位置： $r_i(t+\delta t)=2r_i(t)-r_i(t-\delta t)+\ddot{r}_i(t)\delta t^2+O(\delta t^4)$
3. 计算当前速度： $\dot{r}_i(t)=\frac{r_i(t+\delta t)-r_i(t-\delta t)}{2\delta t}+O(\delta t^2)$
4. 更新阻尼： $\varsigma(t+\delta t)=\varsigma(t)+\frac{1}{Q}\left(\sum_i \dot{r}_i^2-N_fT\right)\delta t$
5. 更新时间： $t_{new}=t_{old}+\delta t$ ，跳转到 1 进行下一个时间步循环。

（4）算法流程图


开始
初始化粒子位置，速度,阻尼系数
计算加速度
更新粒子位置
计算粒子当前速度
更新阻尼系数
是否达到输出条件
否
是
计算粒子动能、势能
输出当前时间、粒子位置，速度、动能、势能
时间更新
结束

4. 问题求解与数据分析

（1） 平衡判定

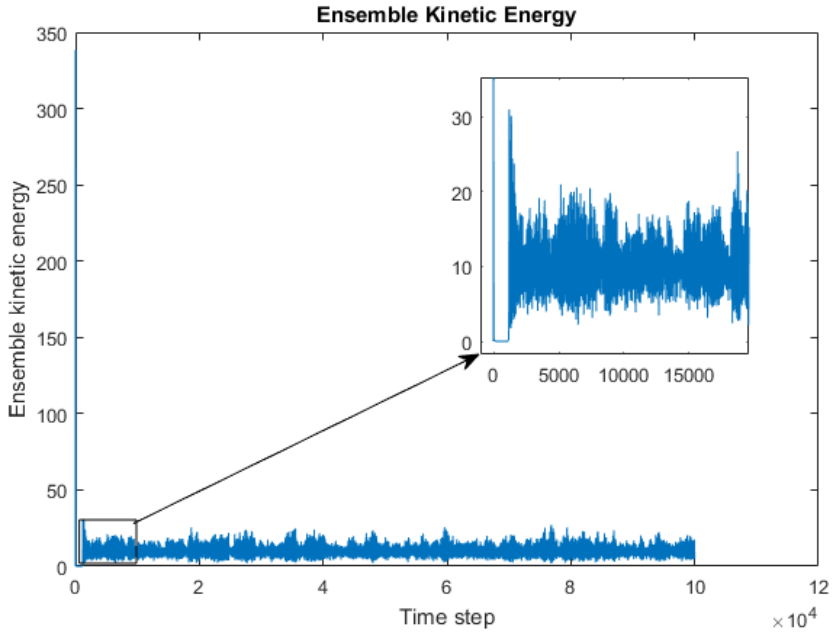


图 4.1 系综动能时间序列

因为我们模拟的是正则系综，因此动能是判断系统是否达到平衡（热平衡）的重要依据，因此我们在数据分析的时候画出系综总动能的时间序列，如图 4.1，从图中看出大约在 1500 次输出后系统就到达了平衡状态，因此我们取 3000 次输出后的数据认为是系统平衡态的时间样本并进行后续的分析。

（2）粒子轨迹与系统相空间庞加莱截面

粒子运动轨迹可以直观的展示出粒子的运动，由于受到外势场的约束，粒子会被约束在一个区域内，从粒子运动轨迹图中可以看出粒子被约束在一个大致半径为 4 的圆域内。因此，在计算系统压强的时候系统边界就按照半径为 4 的圆来计算。

对于十个粒子构成的系综，其相空间的维数为 20 维，为了能够反映一些相空间轨迹的信息，我们这里采用绘制相空间庞加莱截面的方式来绘制相轨迹在二维相空间的子空间里的投影。分析图 4.3 我们可以发现粒子的相轨迹基本上遍历了相空间，达到了混沌的状态，因此各态遍历假设是近似满足的，后面的分析可以建立在这基础上。

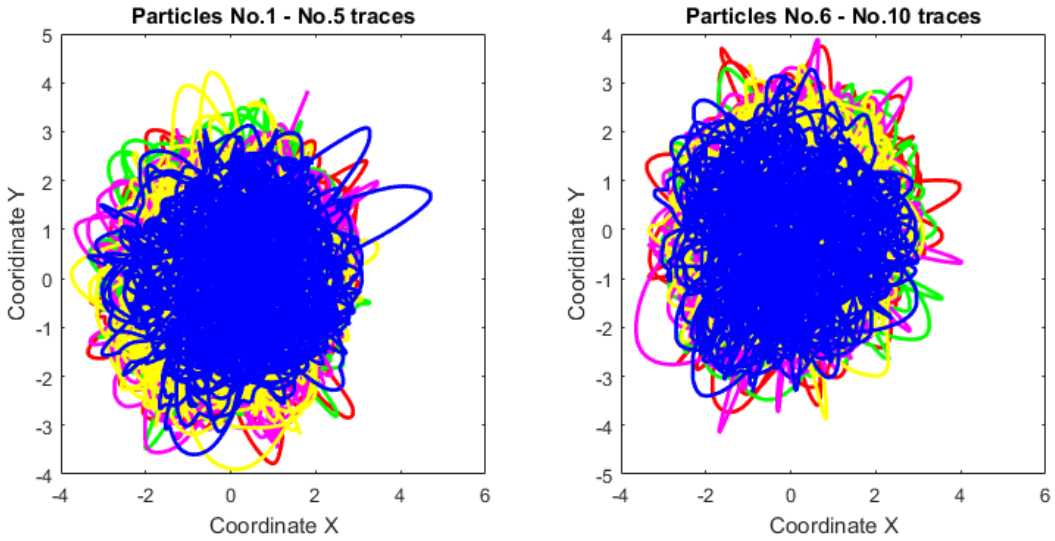


图 4.2 粒子运动轨迹

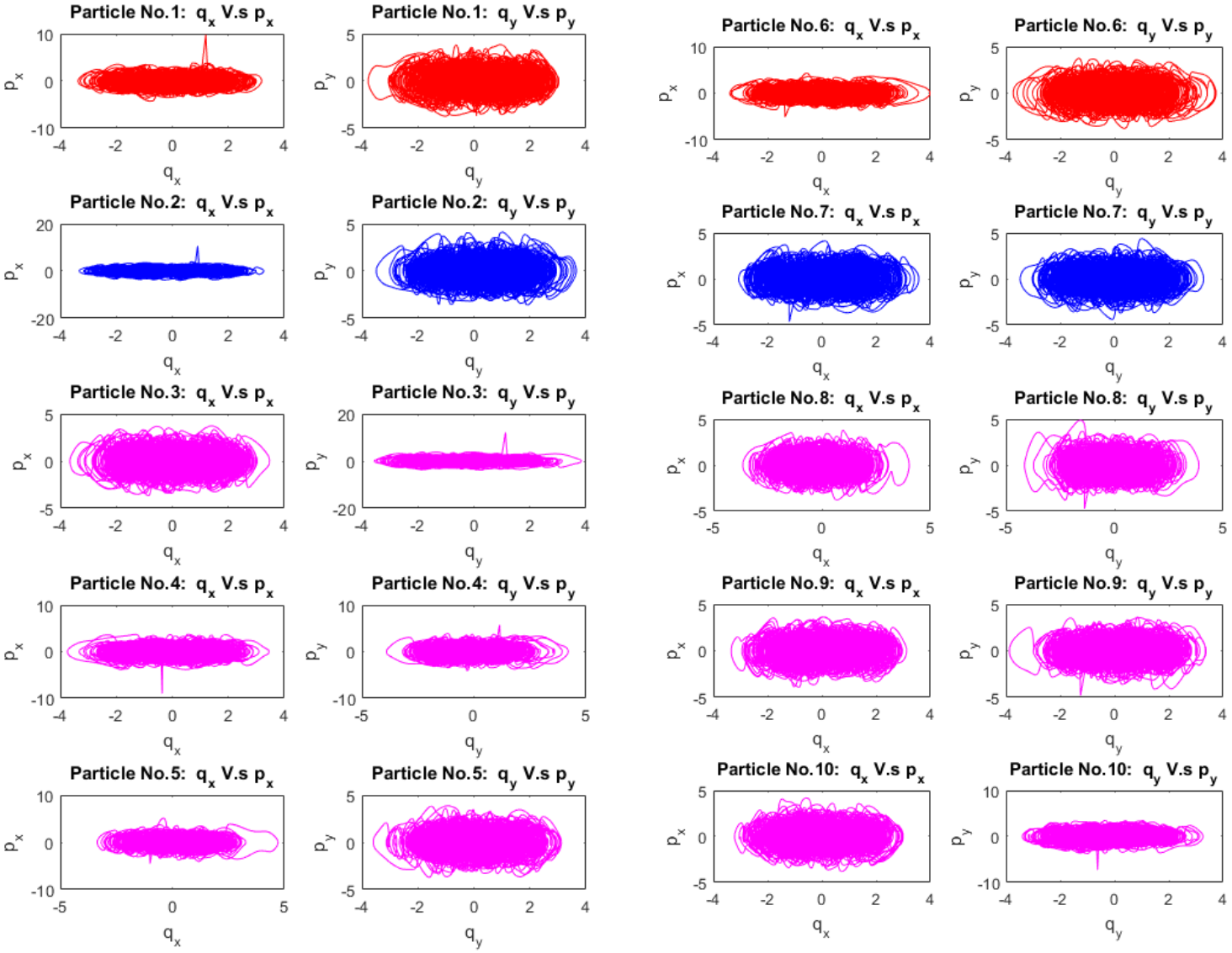


图 4.3 相空间庞加莱截面

（3）数据总体统计分析

下面以及后续的分析属于数理统计分析的内容，因此有必要按照数理统计的常规流程进行问题分析，并在分析过程中回答题目中的问题。由于 MATLAB® 软件具备较强的统计分析工具箱，因此我们使用这个工具箱完成相关的分析。

我们关心的关于每个粒子的统计量主要有下面的几个：粒子的位置，粒子的速度，粒子的动能，粒子的势能，粒子的机械能。箱线图能够很好的反映出这些数据的分布趋势。

粒子位置时间样本箱线图：

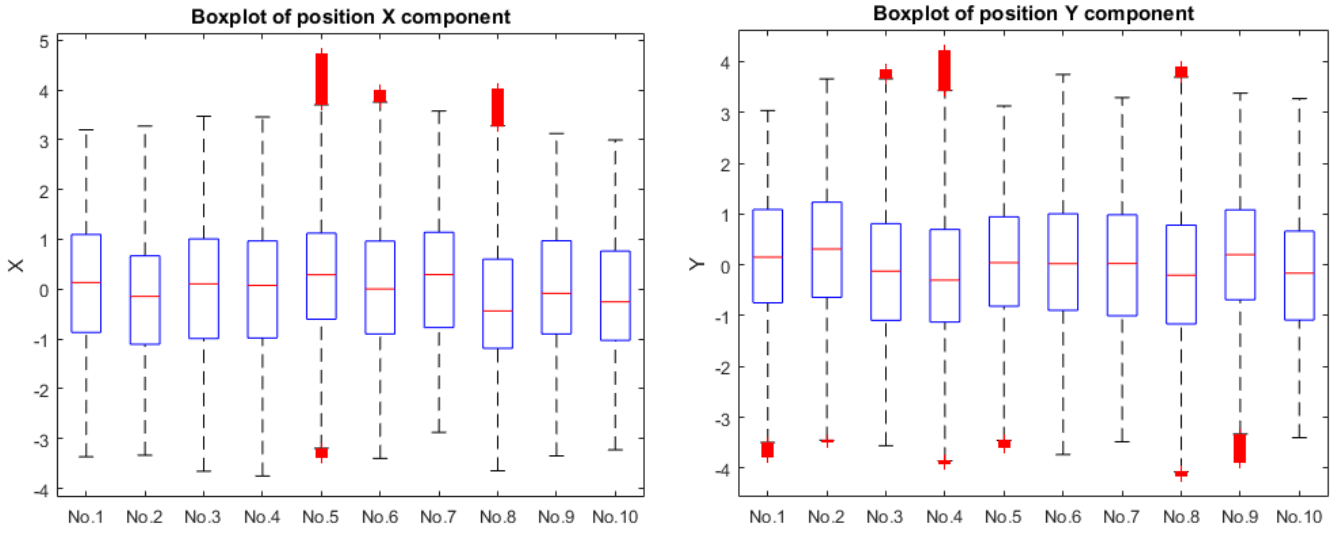


图 4.4 粒子位置样本箱线图

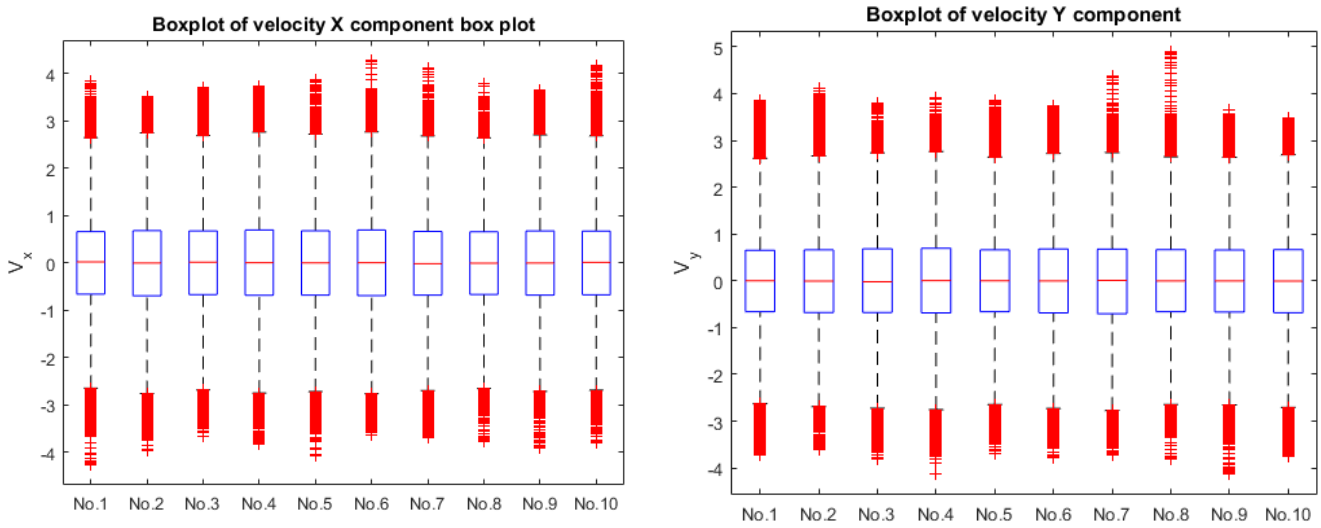


图 4.5 粒子速度样本箱线图

分析粒子位置样本的箱线图（图 4.4）可以看出，不同粒子位置分布的中位数是不相同的，但是上四分位数和下四分位数间间隔基本一致，再看粒子位置分布的上边缘和下边缘可以确定粒子的系统边界大约是一个半径为 4 的圆。分析粒子速度样本的箱线图（图 4.5）我们得到，所有粒子速度分布的中位数都是 0，并且上四分位数和下分位数之间的间隔也是基本相同的，同时数据分布具有很好的对称性，因此速度样本可能存在正态性，从箱线图中还看到偏离整体的离群有些偏多。

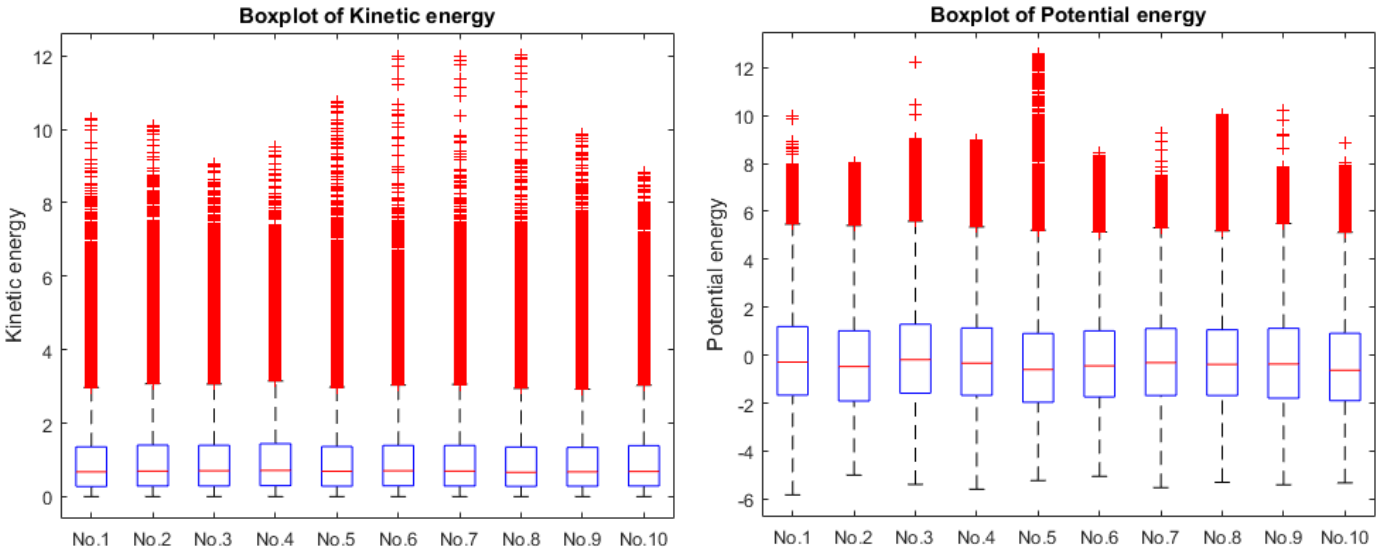


图 4.6 粒子动能与势能样本的箱线图

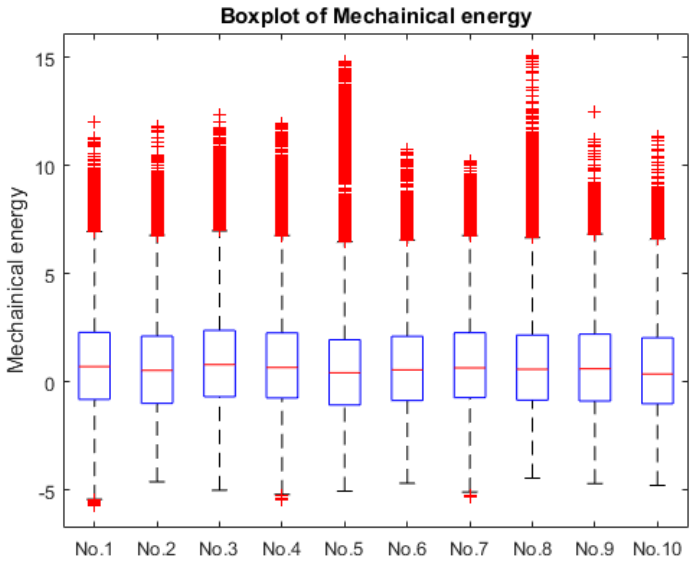


图 4.7 粒子机械能样本的箱线图

分析粒子动能样本的箱线图和粒子势能样本的箱线图（图 4.6），所有粒子动能的中位数都基本上都是 1，说明动能均分在了每个粒子上，并且上下四分之一分位数之间的间隔也基本一致，同时我们也看到每个粒子都有大量的离群数据，这说明我们用的恒温器离散算法性能不是太好，动能随时间震荡剧烈。不同粒子的势能中位数有所不同，同时上下分位数之间的间隔基本一致，造成样本方差的原因可能是相同的，然后势能的分布区间在正负 6 之间。

（4）相关性分析

相关性分析的主要对象是所有粒子速度的所有分量，所有粒子的势能，单个粒子的速度和位置。同时为了方便，在这里使用矩阵图的方式，来观察数据的相关性，不进行严格的线性相关性分析以及假设检验。

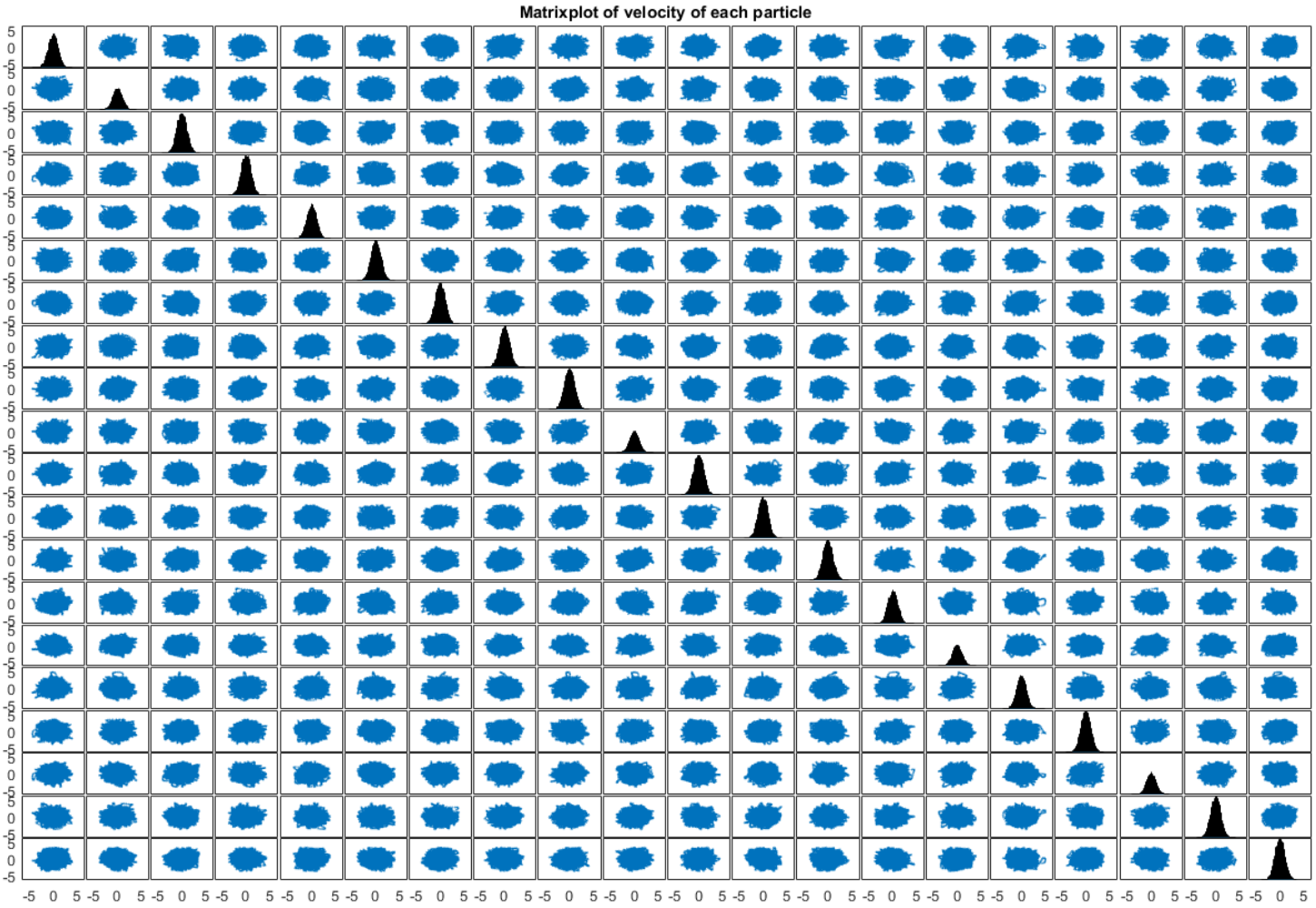


图 4.8 每个粒子所有速度分量的矩阵图

这里需要说明图 4.8 是按照下面的方式做出来的，20 个变量排列方式分别如下：

$$\left(v_x^1,v_y^1;v_x^2,v_y^2;v_x^3,v_y^3;v_x^4,v_y^4;v_x^5,v_y^5;v_x^6,v_y^6;v_x^7,v_y^7;v_x^8,v_y^8;v_x^9,v_y^9;v_x^{10},v_y^{10}\right)$$

然后随机变量和自己时间做条形图，可以展示出这个随机变量自己的统计分布规律，不同变量之间两两作图，可以展示出变量两两之间的规律比如相关性。分析这个矩阵图，可以看出速度变量自己呈现的钟形的正态分布，然后不同变量之间做的的散点图基本上是一个圆形区域分布，所以基本上可以排除里面存在的线性相关性。因此无论是不同的粒子的速度还是不同的速度分量，基本上是统计独立的，其数学表达如下：

$$\rho\left(v_x^1,v_y^1;v_x^2,v_y^2;...;v_x^{10},v_y^{10}\right)=\rho\left(v_x^1\right)\rho\left(v_y^1\right)\rho\left(v_x^2\right)\rho\left(v_y^2\right)...\rho\left(v_x^{10}\right)\rho\left(v_y^{10}\right)$$

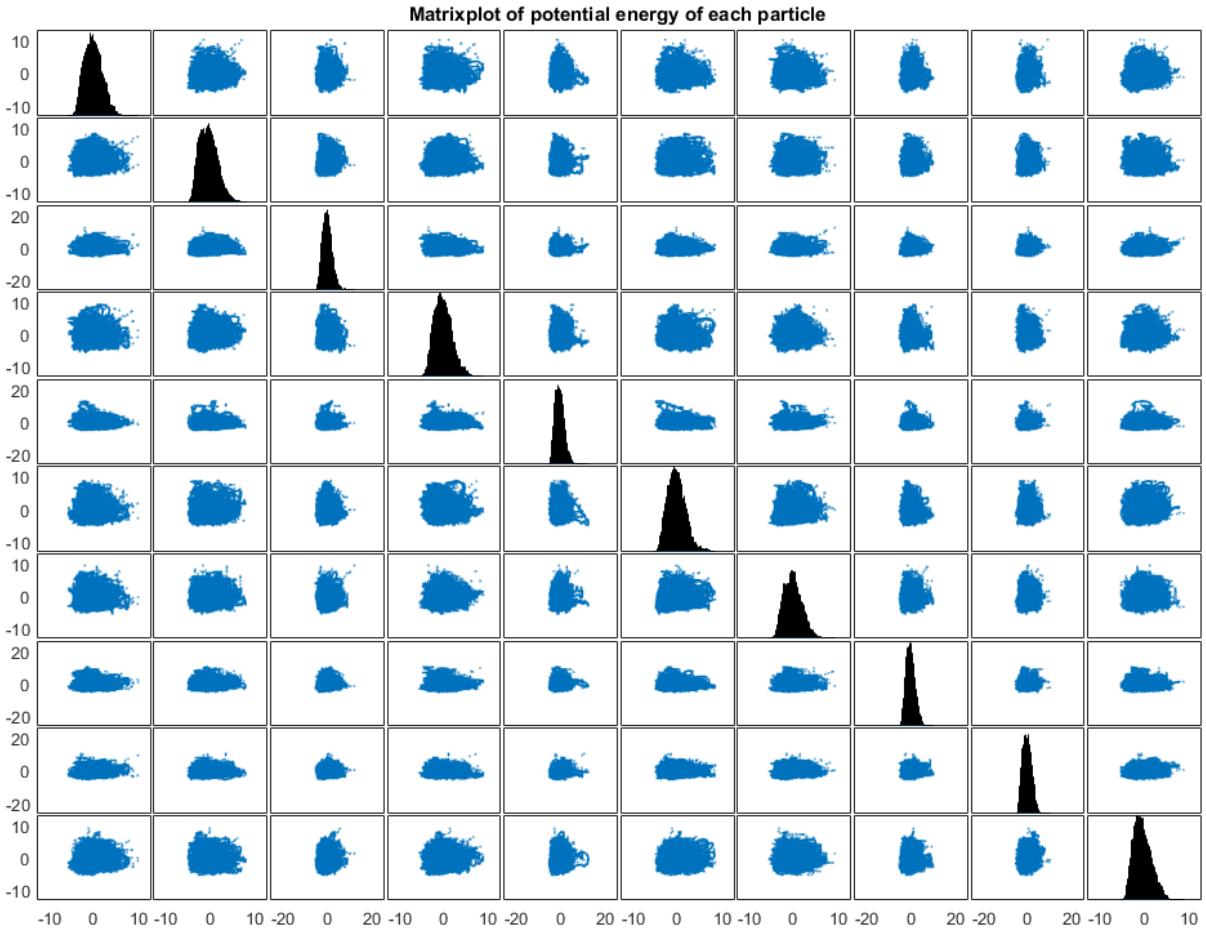


图 4.9 不同粒子势能的矩阵图

粒子势能的矩阵图中的随机变量如下

$$\left(V^1,V^2,V^3,V^4,V^5,V^6,V^7,V^8,V^9,V^{10}\right)$$

从每个随机变量自己的统计图（图 4.9 对角线）看出势能似乎不存在形如 $a\exp\left(-\frac{V}{T}\right)$ 的吉布斯分布，因此我们回答了题目中的一个问题。另外可以看出不同粒子之间的势能分布也不存在显著的统计相关性。

对于不同的粒子有些统计规律是一致的，所以这里我们只挑选出粒子 1 进行更详细的分析：

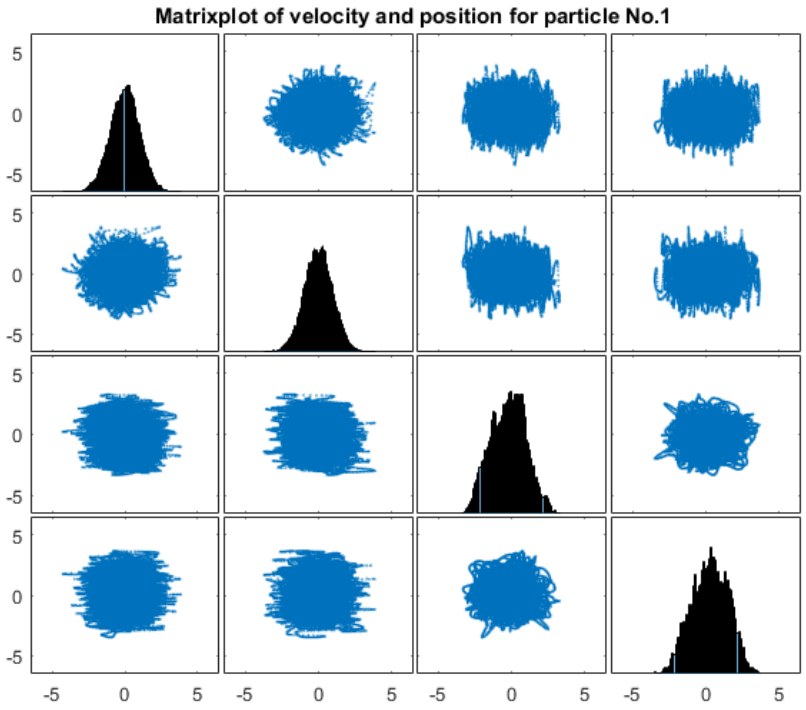


图 4.10 粒子机械能样本的箱线图

在图 4.10 中我们画出粒子 1 的速度和位置随机变量的统计箱线图，变量顺序如下：

$$\left(v_x, v_y, x, y\right)$$

从图中可以看出粒子速度分布具有类似正态的特点，同时通过看不同变量之间做的散点图可以知道速度和位置是统计独立的，而且不同的速度不同的位置之间也是独立的，也就是这四个随机变量两两统计独立。由于每个粒子都是这样的规律，粒子之间也是这样的，因此我们又回答了一个问题：达到统计平衡后 $\rho(u)$ 和 $\rho(r)$ 是独立的也就是有下面的数学表达式：

$$\rho(u,r)=\rho(u)\rho(r)$$

额外的，我们对系综的能量也做一个箱线图，分析其统计规律，随机变量顺序如下：

$$(T,V,E)$$

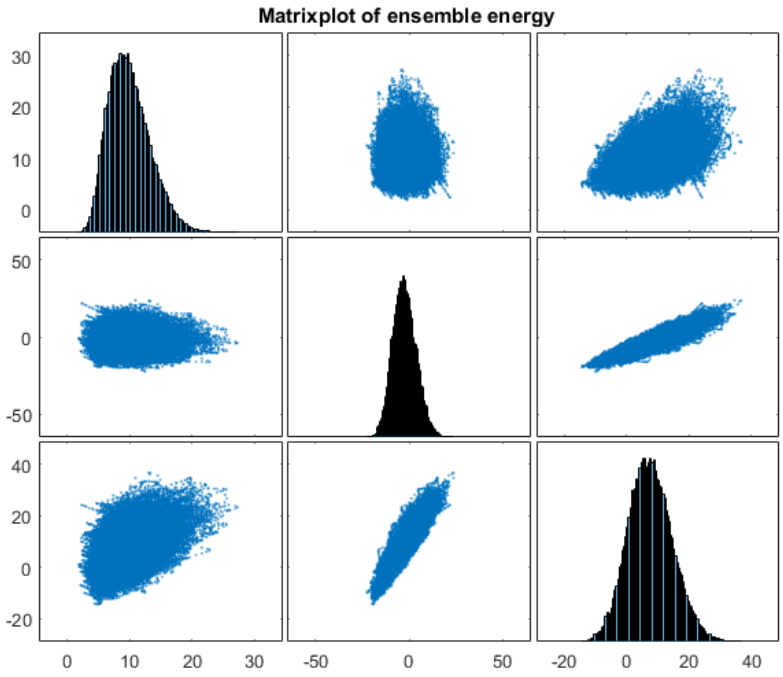


图 4.11 系综动能、势能、总能矩阵图

我们看到系综的势能和动能分布并不像吉布斯分布，同时系综势能和动能之间并不存在明显的统计相关性，但是势能和总能，动能和总能之间是显著线性相关的，这是因为有下面的表达式：

$$E=T+V$$

我们看到矩阵图能很好的反映一些随机变量自己和之间的统计规律。另外分析图 4.11 我们可以得到下面的数学表述：

$$\rho(E)=\rho(T,V)=\rho(T)\rho(V)$$

（5）分布估计

这里的分布估计是指的是对速度分布的估计，前面已经说明了，不同粒子不同速度分量之间的分布是基本相同的，并且之间是独立的，所以不失一般性，这里只分析粒子 1 的 X 方向的速度分布。由统计物理的知识我们知道，对于粒子构成的正则系综的速度分布是麦克斯韦玻尔兹曼分布：

$$f(v)=f(v_x)f(v_y)=\frac{m}{2\pi T}\exp\left(-\frac{m\left(v_x^{\,2}+v_y^{\,2}\right)}{2T}\right)$$

我们只考察 X 方向的分量：

$$f(v_x)=\left(\frac{m}{2\pi T}\right)^{\frac{1}{2}}\exp\left(-\frac{mv_x^{\,2}}{2T}\right)$$

有上面结果的原因是和各态遍历以及能量均分原理有关的。考虑到系综的温度为 1，质量也是单位 1 所以上面的公式又可简化成：

$$f(v)=f(v_x)f(v_y)=\frac{1}{2\pi}\exp\left(-\frac{v_x^{\,2}+v_y^{\,2}}{2}\right)$$

$$f(v_x)=\left(\frac{1}{2\pi}\right)^{\frac{1}{2}}\exp\left(-\frac{v_x^{\,2}}{2}\right)$$

上面 X 方向的速度分布就是需要我们进行验证的，因此我们得到了一个正态分布参数估计的问题：

$$f(v_x)=\left(\frac{1}{2\pi\sigma}\right)^{\frac{1}{2}}\exp\left[-\frac{(v_x-\mu)^2}{2\sigma}\right]$$

希望求出参数$(\hat{\mu},\hat{\sigma})$的估计值，利用 MATLAB® 的统计工具箱可以很容易做到。结果如下：

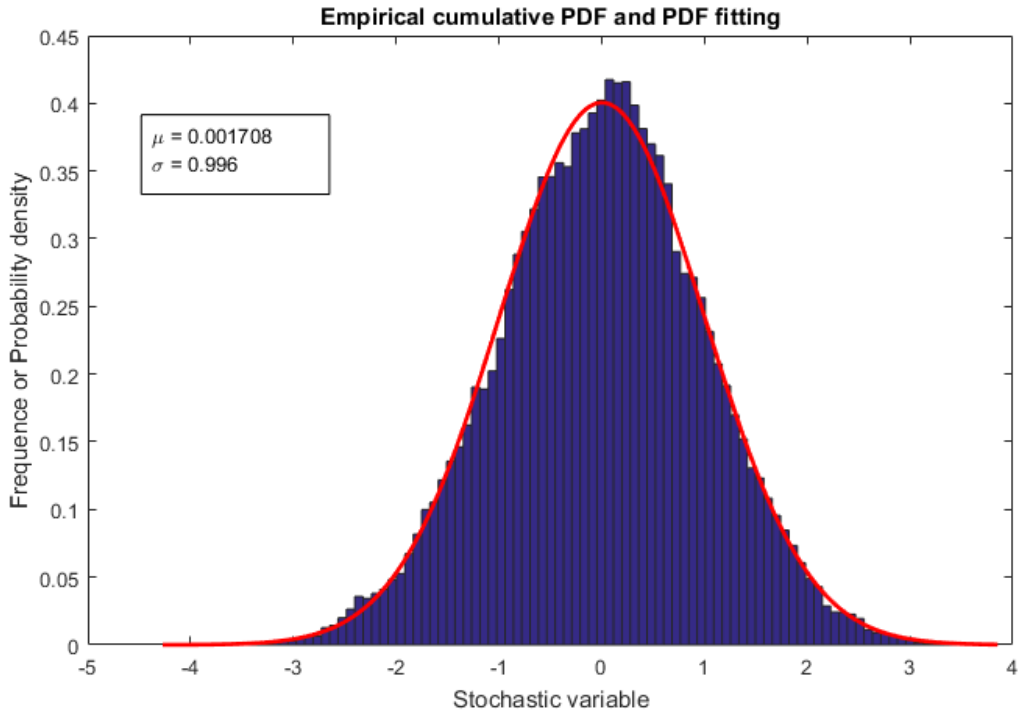


图 4.12 粒子 1 X 方向速度分布的估计

基于概率论的统计物理给出这个分布的参数值为 $\mu=0,\sigma=1$，而基于确定性微分方程理论的分子动力学模拟结果给出这个分布的参数估计值为 $\hat{\mu}=0.00176\cancel{68}\quad 0$，这个估计和理论值相比的误差是很小的。所以这里我们回答了题目中的另一个问题，系统达到平衡后速度确实是服从麦克斯韦分布的，而且不同分量和不同粒子的速度之间是独立的，动能均分到每一个粒子和每一个方向（系综的总的动量自由度）。

为了更清楚的展示出这些结论我们使用统计分析中常用的正态图的方法分析粒子 1 两个速度分量样本。

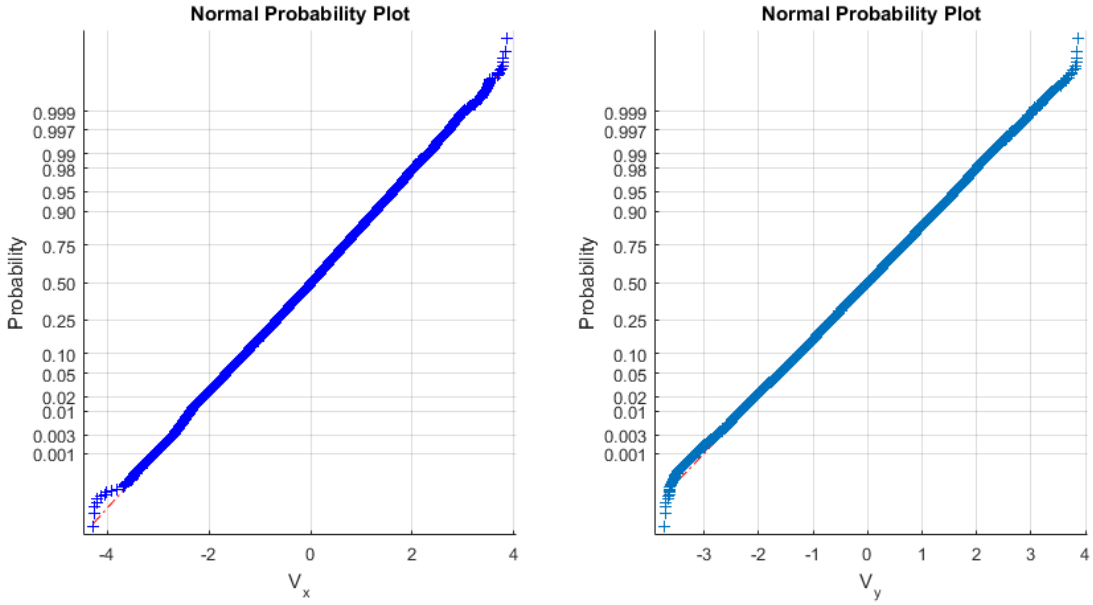


图 4.13 粒子 1 能速度分量样本的正态图

看到数据散点基本上都是位于正态图中一条直线上，因此粒子 1 不同速度分量样本上是同分布于麦克斯韦分布的，而且具有非常好的正态性。再次验证了上面的结论。

（6）系综压强

运用维里定理求出粒子系统的压强，对于维里定理的叙述已经在前面完成了，下面我们直接利用前面的结论。

对于内力的维里函数为：

$$W^{internal}(r_i)=\sum_i r_i\cdot f_i^{internal}$$

因此我们在计算机模拟的过程中要计算和记录这个量，因此这个量也是一个时间序列，它的样本经验分布密度如下：

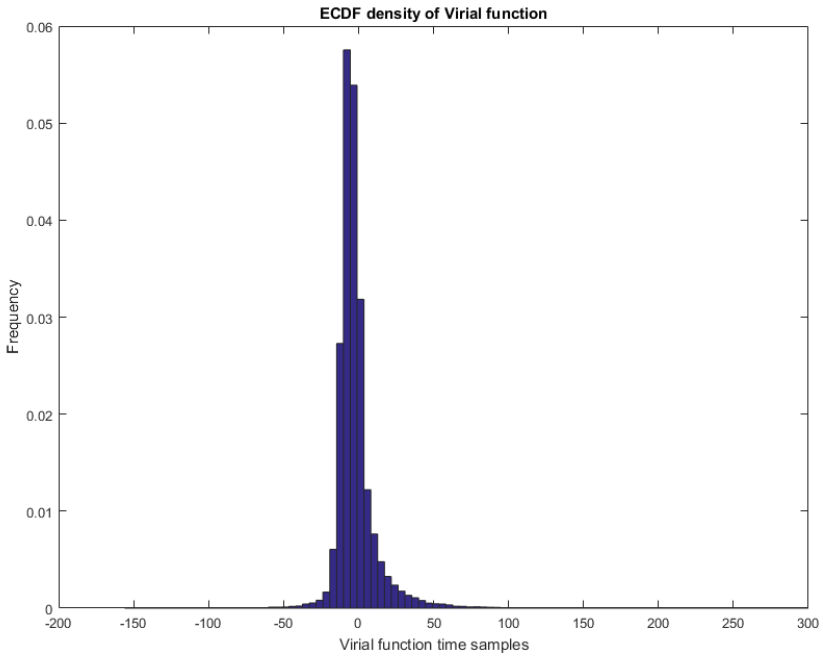


图 4.14 维里函数经验分布密度

而压强由维里函数的系综平均给出，这里的系综平均我们做时间平均。

$$p=\frac{Nk_BT}{V}+\frac{1}{2V}\left\langle\sum_i r_i\cdot f_i^{internal}\right\rangle$$

系统粒子数目 $N=10$，系统温度 $T=1$，由于归一化 $k_B=1$。系统体积：

$V=\pi r^2=16\pi$

求得：$pV(r)=9.0390$

最后系统压强：$p=\frac{9.0390}{16\pi}\approx 0.1790$

程序附录：

说明：由于部分计算程序按照高计算量的大问题编写，所以程序较长，只列出程序的顶部的注释。

1.分子动力学模拟 C++程序

程序名：MD.cpp　作者：黄裕函

```
#include <iostream>
#include <cmath>
#include <cstdio>
#include <fstream>

using namespace std;

const int N = 10;           //粒子个数
const int DIM = 2;              //问题维数
const int MAX_STEP = 10000000; //最大迭代步数
const double Ms = 0.2;          //Nose-Hover 热库参数
const double Temperature =1.0; //Nose-Hover 热库温度
const bool isNoseHover = true; //Nose-Hover 热库开关

const double PI = 3.1415926;

double* pos;            //存储位置
double*   pre_pos;          //存储上一步位置
double* velocity;       //存储速度（暂未使用）
double* force;              //存储受力
double* potential;
double* kinetic_energy;
double max_force;       //存储最大受力
double eta = -1.0;
double time_step;//时间步长
double time=0.0; //统计当前物理时间
int step;

void initialize(){
    //此处设置初始位置
    //初始位置按照圆周均匀分布
    for (int i = 0; i < 10; i++) {
        pos[i*DIM + 0] = cos(2*PI * ((double)i) / N);
        pos[i*DIM + 1] = sin(2*PI * ((double)i) / N);
    }
```

```
    for(int i=0;i<DIM*N; i++)
        pre_pos[i] = pos[i];

    for(int i=0;i<DIM*N; i++)
        *(velocity+i) = 0.0;
}

void compute_force(){
    /*
      totalForce 两个粒子 ij 之间的相互作用力合力,吸引为负
      dist 两个粒子之间的距离
      */
    for(int i=0; i<N*DIM; i++){
        force[i] = 0.0;              //将每个粒子的受力清零
        max_force = 0.0;          //存储最大合力（用于变步长计算）
    }
    double  r, x, y, dist, dx, dy, totalForce;
    for(int i=0;i<N;i++){
        x = pos[i*DIM + 0];
        y = pos[i*DIM + 1];
        r = x*x + y*y;
        force[i*DIM + 0] += -x;      //外势力作用
        force[i*DIM + 1] += -y;      //外势力作用
        for(int j=i+1;j<N;j++){
            dx = pos[i*DIM + 0] - pos[j*DIM + 0];
            dy = pos[i*DIM + 1] - pos[j*DIM + 1];
            dist = sqrt(dx*dx+dy*dy);
            totalForce = (48*pow(dist,-13.0)-24*pow(dist,-7.0));

            //printf("i=%d,j=%d, dist ij = %f, force = %f\n",i,j,dist,
totalForce);
            force[i*DIM + 0] += totalForce*dx/dist;
            force[i*DIM + 1] += totalForce*dy/dist;
            force[j*DIM + 0] -= totalForce*dx/dist;
            force[j*DIM + 1] -= totalForce*dy/dist;
        }
        if(isNoseHover){             //对 Nose-Hover 热库
            //eta = 1.0;
            force[i*DIM + 0] += -eta*velocity[i*DIM + 0];
            force[i*DIM + 1] += -eta*velocity[i*DIM + 1];
            //if(eta                >                50                )
printf("step=%d,time=%f,fx=%f,fy=%f,x=%6f,y=%6f,vx=%f,vy=%f,eta=%f
\n",
            //
```

```
    step,time,force[i*DIM+0],force[i*DIM+1],pos[i*DIM],pos[i*DIM+1]
,velocity[i*DIM+0],velocity[i*DIM+1],eta);
        }
        //if(abs(force[i*DIM    +    0]>max_force))    max_force    =
abs(force[i*DIM + 0]);
        //if(abs(force[i*DIM    +    1]>max_force))    max_force    =
abs(force[i*DIM + 1]);
    }

    for(int i=0; i<N; i++){

    //printf("i=%d,fx=%f,fy=%f,vx=%f,vy=%f\n",i,force[i*DIM+0],forc
e[i*DIM+1],velocity[i*DIM+0],velocity[i*DIM+1]);
    }
}

// Verlet 算法
void compute_pos(double step = 1e-5){
    for(int i=0;i<N*DIM;i++){
        pre_pos[i]  =  2*pos[i]-pre_pos[i]+step*step*force[i];    //
Verlet 算法
        velocity[i] = - (pos[i]-pre_pos[i]) / step;
    }
    swap(pre_pos,pos);
}

void compute_energy(){
    double  dist, dx, dy;
    for(int i=0;i<N;i++){
        kinetic_energy[i] = 0.5*velocity[i*DIM+0]*velocity[i*DIM+0]
                          +0.5*velocity[i*DIM+1]*velocity[i*DIM+1];
        potential[i]                                                =
0.5*(pos[i*DIM+0]*pos[i*DIM+0]+pos[i*DIM+1]*pos[i*DIM+1]);
        for(int j=0;j<N;j++){
            if (i==j) continue;
            dx = pos[i*DIM + 0] - pos[j*DIM + 0];
            dy = pos[i*DIM + 1] - pos[j*DIM + 1];
            dist = sqrt(dx*dx+dy*dy);
            potential[i] += 4*(pow(dist,-12)-pow(dist,-6));
        }
    }
}

//打印所有粒子的位置信息
```

```
void print_pos(int step = 0){
    for(int i=0; i<N; i++){

    printf("step:%d,i=%d,x=%f,y=%f\n",step,i,pos[i*DIM+0],pos[i*DIM
+1]);
    }
    cout << endl;
}

//Nose-Hover 热库
//a = f/m -> a=f/m - v*eta
//此部分计算 eta
void compute_eta(double timeStep){
    double total_velocity_square = 0.0;
    for(int i=0; i<N; i++){
        total_velocity_square += velocity[i*DIM+0]*velocity[i*DIM+0]
                              +velocity[i*DIM+1]*velocity[i*DIM+1];
    }
    eta += timeStep*(total_velocity_square - N*DIM*Temperature)/Ms;
    //此处有疑问，公式中的 h 等于自由度?
}

int main(){
    pos        = new double[N*DIM];
    pre_pos       = new double[N*DIM];
    velocity   = new double[N*DIM];
    force      = new double[N*DIM];
    potential    = new double[N];
    kinetic_energy = new double[N];
    initialize();
    print_pos(0);
    system("pause");
    ofstream out("result.txt");

    for(step=0;step<MAX_STEP;step++){
        time_step = 1e-5;//1.0/max_force;      //计算应该使用的时间
步长（取为固定步长）
        if(isNoseHover){
            compute_eta(time_step);            //计算热库摩擦力
        }
        compute_force();                  //计算所有粒子受力
        compute_pos(time_step);              //向前推进一步
        time+=time_step;                  //计算此步物理时间
        if(step%10000 == 0)                  //每隔 10000 步存储一次数
```

```
据
        {
            compute_energy();
            out << time << " " ;
            for(int in=0; in<N; in++){
                out << pos[in*DIM +0] << " " << pos[in*DIM +1] << " "
                    << velocity[in*DIM +0] << " " << velocity[in*DIM
+1] << " "
                    << kinetic_energy[in] << " " << potential[in] << "
";
            }
            out << endl;
            if(step%1000000 == 0)               //每隔1000000显示一次
进度
                print_pos(step);
        }
    }
    out.close();
    return 0;
}
```

2.分子动力学模拟 Fortran 程序

程序名：TenBodyMD.f90　　作者：杨阳　　注：程序较长，略

```
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! Program name: TenBodyMD.f90
!
! Program Purpose:
!                        Solve ten-bodies probelm with Nose-Hoover
Thermostat in molecule simulation.
!  Version message:
!        Date          Programmer            Description of change
!                     ========                          ============
========================
!  1.  36/4/2016      Yang Yang :-)              Source Code
!
!
!
! A brief algorithm introduction:
! First time step is caculated using velocity-verlet algorithm and
other time step is
! caculated using verlet algorithm. Velocity is caculated using
centered differential scheme.
! $r(0 + \delta t) = 2r(0) - \delta t*v(0) + \frac{1}{2} \delta
```

```
\t^{2} a(0)$
!!$v(0 + \delta t) = v(t) + \frac{1}{2} \delta t [a(0) + a(0 + \delta
t)]$
! $r(t+\delta t) = 2r(t) - r(t - \delta t)+ \delta \t^{2}a(t)$
! $v(t) = [r(t + \delta t) - r(t - \delta t)] / 2\delta t$
! $v(N) = [r(N) - r(N - \delta t)] / \delta t$
! external field : $V(r) = \frac{1}{2} \r^{2}
! internal reaction : Leonard-Jones 6-12 potential  $V(r_{ij}) =
4[(\frac{1}{r_{ij}})^{12} - (\frac{1}{r_{ij}})^{6}]
! Nose-Hoover thermostat: $zeta(t_\delta t) = zeta(t) + frac{\Sigma
m_{i}v_{i}^2 - N_{f}T}{Q} \delta t$    ~ O(delta_t)
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

3.分子动力学数据后处理 MATLAB 程序

程序名：MDDataAnalysis.m　作者：杨阳　　注：程序较长，略

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Program name : MDDataAnalysis.m
%%%% Program Author: Yang Yang
%%%% Data : 30/4/2016
%%%% Version: V1.0
%%%% copyright: Author only!
%%%% input file format: result.txt
%%%% data column (
%%%% t, x1, y1, u1, v1, fx1, fy1, k1, p1,
%%%%    x2, y2, u2, v2, fx2, fy2, k2, p2,
%%%% ......
%%%%    x10, y10, u10, v10, k10, p10)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```