对空腔内气体振动发声的研究

10316108 王福毅

（东南大学，南京 211102）

摘 要： 作者综合运用计算物理的多种方法，模拟空腔中的气体振动产生的声波。最后发现在空腔内部形成了驻波，而驻波的频率的分布受到气体相对分子质量和温度的影响，相对分子质量越大，温度越低，驻波中低频占比越高，反之亦然。其中频率分布的变化对相对分子质量更为敏感。在常温下，甚至可以忽略温度变化对声波频率分布的影响。

关键词： 空腔；气体振动；蒙特卡罗方法；分子动力学；傅里叶变换

A study about the vibration of gas in the cavity

Fuyi Wang

(Southeast University, Nanjing ,211102)

Abstract: The author synthetically used many methods of computational physics to simulate the sound waves generated by gas vibration in the cavity. The standing wave formed inside the cavity and the frequency distribution of the standing wave is influenced by the gas's relative molecular mass and temperature. The larger the relative molecular mass and the lower the temperature, the ratio of the low frequency in the standing wave is higher, vice versa. The variation of frequency distribution is more sensitive to relative molecular mass. Under normal temperature, the influence of the temperature change on the distribution of acoustic frequency can be neglected.

key words: Cavity; Gas vibration; Monte Carlo method; Molecular dynamics; Fourier transform

1 研究背景

声音是一种压力波，当演奏乐器、拍打一扇门或者敲击桌面时，他们的振动会引起介质（空气）有节奏的振动，使周围的空气产生疏密变化，形成疏密相间的纵波，这就产生了声波，这种现象会一直延续到振动消失为止。而如果声源本身是一个充满气体的空腔，如各种管弦乐器和人的发声器官，人们一般认为产生的声音频率与空腔的大小有关，例如人们认为乐器中气柱越短声音频率越高，认为体型宽厚的人说话声音似乎更加低沉或者带有鼻音。但是，除了空腔的长度，可能还有一些其他因素会影响声音的频率，例如温度和气体的相对分子质量。因此，我想利用计算物理的方法模拟充满气体的空腔产生声音的过程，并研究气体分子质量，温度这 2 个因素对声音频率的影响。进而分析在气体平均分子质量和温度发生改变时，对人和管弦乐器产生的声音有着怎样的影响。

2 问题分析

研究时考虑一个充满气体的长方体空腔，如图所示：

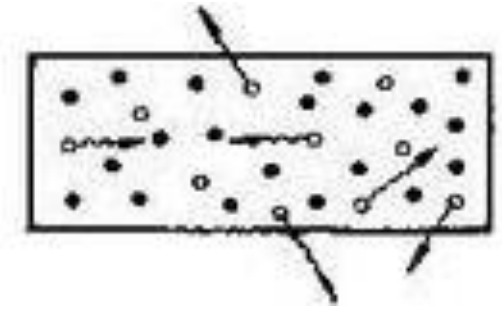

图 1 长方体空腔

考虑到声波在到达空腔壁时会发生反射，因此猜想空腔内产生的声波是驻波。若声波为驻波，则具有如下性质：

1. 频率是分立而不连续的；
2. 随着空腔长度的缩短或增加，频率响应的提高或减小；

若空腔壁中的声波的确为驻波，则猜想与空腔的长度对声音造成影响的原因不同，温度和相对分质量对声波造成的影响的原因是改变了气体的平均速率。气体分子速率越大，振动的也越快，因此频率应该提高。但是由于频率只能取分立的值，且气体分子速率分布较广，故同时存在多个驻波。因此当气体分子运动速率提高或减少时，对应为声波中的高频或低频占比增加。由于假定气体的速率分布服从麦克斯韦速率分布（忽略重力）：

$$f(v)=4\pi\left(\frac{m}{2\pi kT}\right)^{\frac{3}{2}}e^{-\frac{mv^2}{2kT}}v^2 \tag{1}$$

因此计算得气体的平均速率为：

$$\bar{v}=\int_0^{\infty}vf(x)\,dv=\sqrt{\frac{8kT}{\pi m}}=\sqrt{\frac{8RT}{\pi M}} \tag{2}$$

温度越高，气体平均速率越大，因此气体高频占比增加；相对分子质量越小，气体平均速率越小，因此气体低频占比增加。

为了验证猜想，因此需要利用计算物理的知识对这一过程进行模拟。

3 物理模型的建立

3.1 模型假设

设空腔为一长方体，有固定的长宽高。空腔外壁绝热，气体分子与壁之间为弹性碰撞。

为了方便计算，选取单原子气体，且只考虑气体的平动自由度，假设气体分子的速度方向没有最优取向。将气体分子视为具有电偶极矩的小球。同时考虑了分子间作用力之后，分子间作用势的计算方法参考列纳德-琼斯（Lennard-Jones）半经验公式.

$$\Phi(\mathrm{r})=\Phi_0\left[\left(\frac{r_0}{r}\right)^{12}-2\left(\frac{r_0}{r}\right)^{6}\right] \tag{3}$$

这里认为分子间作用力平衡位置为 10 个单位长度，最远作用距离为 20 个单位距离。同时认为Φ_0与相对分子质量成正比。

假设声音由敲击空腔壁产生，在敲击的瞬间空腔壁发生形变，从而推动了在空腔壁附近的气体分子，由于气体分子的质量很小，与空腔壁相比几乎为 0，因此对空墙壁的瞬时速度不产生影响，故认为每个被撞击气体分子增加相同的速度，而与气体分子的分子质量无关。

假定气体的速率分布满足麦克斯韦速率分布：

$$f(v)=4\pi\left(\frac{m}{2\pi kT}\right)^{\frac{3}{2}}e^{-\frac{mv^2}{2kT}}v^2 \tag{4}$$

3.2 利用蒙特卡罗算法初始化气体

这里并非使用蒙特卡罗进行数值计算，而是利用蒙特卡罗算法在给定温度下逆推出气体的速率分布。由假设，气体的速率分布满足麦克斯韦速率分布：

$$f(v)=4\pi\left(\frac{m}{2\pi kT}\right)^{\frac{3}{2}}e^{-\frac{mv^2}{2kT}}v^2 \tag{5}$$

生成速率之后，在球坐标中随机等可能地选取φ与$\cos(\theta)$的值，从而得出气体在 xyz 三个方向上的速度分量。完成初始化后，敲击空墙壁，同时等待 50 个单位时间，使其内部产生的声波稳定形成驻波。

3.3 利用分子动力学模拟声波的产生

在初始化完成之后，在 t=0 时使得在某一空腔壁（z=0）附近的气体分子在垂直于空腔壁的方向获得额外的速度。之后每一个气体分子先判断自己周围 20 个单位距离内是否有分子，如果有，则计算分子间作用力对速度的影响。这里用梯形法模拟分子的运动，即相当于二阶荣格-库塔方法，之所以选用二阶是因为这里选用的列纳德-琼斯势不含时间变量，继续求导无意义。在撞击空腔壁之后，对应方向速度分量大小不变，方向反向。

3.4 变量的选取

首先空腔的长度显然可以作为变量因此不必讨论，而在建模过程中发现，模型中相相对分子质量与温度总以比值的形式出现，因此不需要单独考虑，而是令 $\alpha = \frac{m_0}{T}$（其中m_0为相对分子质量，T 为温度），在固定其他条件时把α作为变量来研究声波的变化。

3.5 声音的测量与分析方法

这里通过测量空墙壁瞬时气压的变化间接测量声音。在记录下结果之后利用傅里叶变换分析声音的频谱。首先是固定其他条件只改变空腔的长度，判读是否形成了驻波。若空腔中形成了驻波，则在不同的空腔长度下，都只产生了特定频率的声波，而且的不同的频率等间距出现。且随着空腔长度的增加，高频部分减少，低频部分增加。在确认形成了驻波后，分别改变相对分子质量与温度并保持其他条件不变，观察频谱的变化。为了突出驻波的频率，对得到的频谱作锐化处理。具体锐化方法如下：

将所有的频谱强度取绝对值再除以强度绝对值的最大值，然后将得到的数平方再乘以原先的强度最大值，就得到了锐化后的频谱。这里利用的是$f(x) = x^2$的下凸性。

4 频谱分析的结果

4.1 固定所有变量

这一步的目的是判断初始化时的随机性是否会对结果产生较大的影响。固定所有变量，运行十次，得到结果如下：

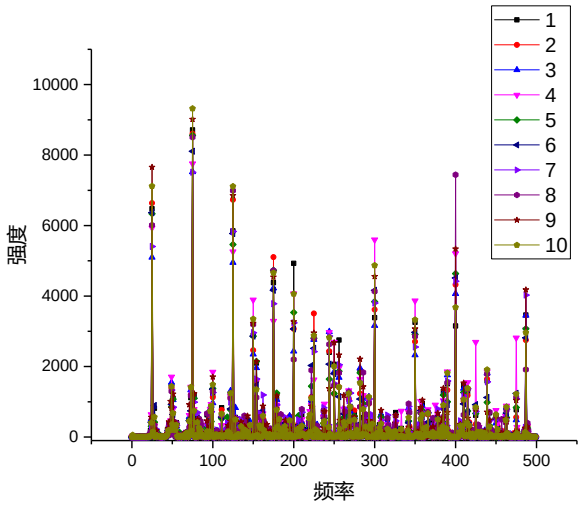

图 2 固定所有变量运行程序十次得到的结果

可以看出尽管峰的强度有一定的差异，10 次模拟结果特征峰的位置完全重合。而且如果忽略高频部分，峰强度的差异也在可接受范围之内。因此可以认为初始化气体的涨落造成的影响不大。

4.2 以空腔长度为变量

取空腔长度分别为 100,130,170 和 200 个单位长度，得到的结果如下图所示：

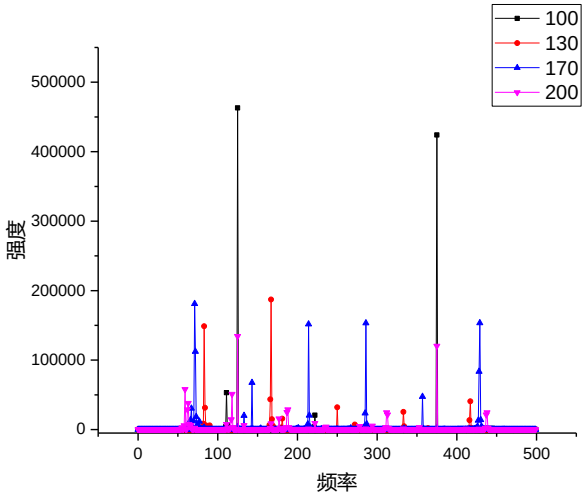

图 3 以空腔长度为变量得到的结果

从图中可以看出，在不同的空腔长度下，都只产生了特定频率的声波，而且的不同的频率等间距出现。且随着空腔长度的增加，高频占比减少，低频占比增加。反映出空腔中的确产生了驻波，与猜想相同。因

此进一步验证声波频率与气体分子质量与温度间的关系。

4.3 以气体相对分子质量为变量

从 10 个单位相对分子质量开始，每次增加 5 个单位的相对分子质量，连续测量 10 次。以频率为 x 轴，相对分子质量为 y 轴，强度为 z 轴，得到三维图如下：

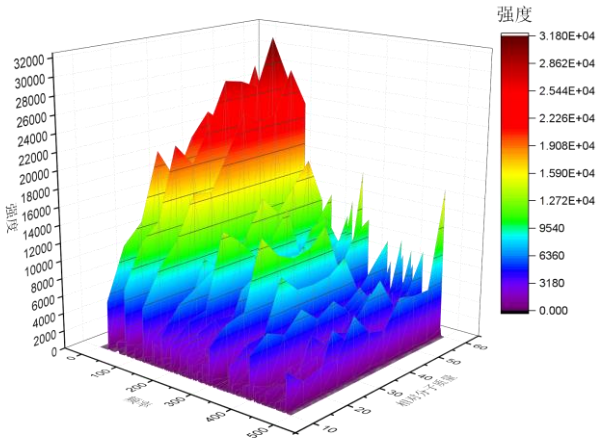

图 4 以气体相对分子质量为变量得到的结果

从图中可以看出，随着相对分子质量的逐渐提高，声波的低频部分显著增加，这说明改变气体的相对分子质量，可以显著改变声波的频率。

4.4 以温度为变量

从 100k 开始，每次增加 50k，连续测量 10 次。以频率为 x 轴，温度为 y 轴，强度为 z 轴，得到三维图如下：

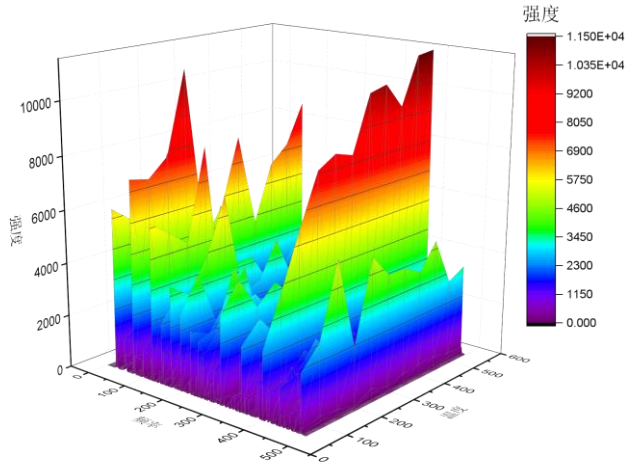

图 5 以温度为变量得到的结果

从图中可以看出，随着温度的升高，高频部分的强度呈现增加趋势，但低频部分的变化趋势并不明显。但是高频部分的增加是比较快的，可以体现出温度升高频率升高的趋势，但是改变温度对声音频率的影响明显小于分子质量对声音频率的影响。

5 得出结论

为什么模拟的结果中，温度对声波频率的影响明显小于相对分子质量呢？

由平均分子运动速率公式

$$\bar{v}=\int_0^{\infty} v f(x)\,dv=\sqrt{\frac{8kT}{\pi m}}=\sqrt{\frac{8RT}{\pi M}} \tag{6}$$

因为摩尔质量与相对分子质量相同，取：

$$\bar{v}=\sqrt{\frac{8RT}{\pi M}} \tag{7}$$

可以通过比较平均速度关于温度与相对分子质量的变化率来研究声波频率对两因素的敏感性。分别对 T 与 M 求导，得：

$$\frac{d\bar{v}}{dT}=\sqrt{\frac{2R}{\pi MT}} \tag{8}$$

$$\frac{d\bar{v}}{dM}=-\sqrt{\frac{2RT}{\pi M^3}} \tag{9}$$

忽略常数，比较$\sqrt{\frac{1}{MT}}$与$\sqrt{\frac{T}{M^3}}$的大小。取 T=300，M=29，计算得：

$$\sqrt{\frac{1}{MT}}\approx 0.0107 \tag{10}$$

$$\sqrt{\frac{T}{M^3}}\approx 0.1109 \tag{11}$$

则可以得出，在正常情况下：

$$\frac{d\bar{v}}{dT}:\frac{d\bar{v}}{dM}=\sqrt{\frac{1}{MT}}:\sqrt{\frac{T}{M^3}}=0.0107:0.1109\approx 0.0965 \tag{12}$$

故实验结果与理论是相符的，温度对声波的影响的确不如相对分子质量。

同时，若认为常温为 300k，且温度下限为 270k，上限为 310k，同时空气的平均相对分子质量比较稳定，可以被视为常数，则：

$$\frac{\bar{v}_{T=310k}}{\bar{v}_{T=270k}}=\sqrt{\frac{310}{270}}\approx 1.07 \tag{13}$$

因此，在日常生活中，温度对气体分子平均速率是很小的，甚至可以忽略。也就是说，在日常生活中，可以认为温度对空腔中气体振动产生的声波的振动频率几乎没有影响。

最终可以得出如下结论：

1. 空腔中产生的声波为驻波；
2. 气体相对分子质量越小，产生的声波中高频的部分强度越大，反之亦然；
3. 气体温度越高，产生的声波中高频的部分强度越大，反之亦然；
4. 产生的声波对相对分子质量的变化比较敏感，温度的变化对其影响较小；
5. 在常温且温度波动在 30 度以内时，若气体相对分子质量不变时，温度对产生声波的频率造成的影响很小。

6 总结

由以上结论可以看出，对声音频率产生影响的主要是气体的相对分子质量。而空气的平均相对分子质量比较稳定。因此在日常生活中，频率发生的改变不容易被察觉。因此管乐器的频率基本只和空腔内气柱的长度有关，人说话的声音在健康和发育完全的情况下也是基本不发生变化的。只有在极端情况下，例如人为地把管乐器充满某种气体或是人直接吸入某种气体（相对分子质量与空气差别较大），这时声音频率的改变才会十分显著。

参考文献：

[1] 汪志诚. 热力学·统计物理. 北京：高等教育出版社，2012.6-8.

附录

代码如下:

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include <ctime>
#include <fftw3.h>

using namespace std;

const int len=500;//容器长宽高
const int wid=500;
const int hei=100;
const double R2=20;//最远作用距离
const double r=10;//分子半径
const double k=1.38;//玻尔兹曼常数
const double dt=0.01;//单位微分时间
const double f0=20;//分子间作用力系数
const int N=25000;//粒子数
double T=100;
double m0=10;
double Alpha=m0/T;//组合系数
char name[20]="dataaverA.dat";
double t=0;
double px=0;
const int JC=11;//实验次数

struct molecule//分子结构体
{
    double x;//实际位置
    double y;
    double z;
    double vx;//速率
    double vy;
    double vz;
    int n;//分子编号
    int near[8];//用于存储与自己相邻的分子的编号
    int location[3];//空间坐标
};
```

```
double distant(molecule a,molecule b)//计算 2 个分子之间距离的函数
{
    return pow((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*(a.z-b.z),0.5);
}

void zero(molecule b[])//清零
{
    for(int i=0;i<N;i++)
    {
            b[i].vx=0;
            b[i].vy=0;
            b[i].vz=0;
    }
}

int scan(molecule a[],int m,int room[][wid][hei])//扫描，记录自己周围的气体分子
{

    int out=0;
    for(int i=a[m].location[0]+6;i<hei&&i<(a[m].location[0]+R2);i++)
        for(int j=a[m].location[1]+6;j<wid&&j<(a[m].location[1]+R2)&&j*j<R2*R2-i*i;j++)
            for(int
k=a[m].location[2]+6;k<len&&k<(a[m].location[2]+R2)&&k*k<R2*R2-i*i-j*j;k++)
            {
                if(room[k][j][i]>=0&&room[k][j][i]<N)
                {
                    a[m].near[out]=room[k][j][i];
                    out++;
                }
            }

    a[m].n=out;
    return out;
}

void impact(molecule a[],molecule b[],int m)//相互作用函数
{

        b[m].vx+=a[m].vx;
        b[m].vy+=a[m].vy;
        b[m].vz+=a[m].vz;
    for(int i=0;i<a[m].n;i++)
    {
        const double rr=distant(a[m],a[a[m].near[i]]);
```

```
            double f=f0*(-pow(r/rr,7)+pow(r/rr,13));//列纳德-琼斯（Lennard-Jones）半经验公式关于距离的导数
            double temp=dt*f/rr;
            b[m].vx+=(a[m].x-a[a[m].near[i]].x)*temp;
            b[m].vy+=(a[m].y-a[a[m].near[i]].y)*temp;
            b[m].vz+=(a[m].z-a[a[m].near[i]].z)*temp;
            b[a[m].near[i]].vx+=(a[m].x-a[a[m].near[i]].x)*temp;
            b[a[m].near[i]].vy+=(a[m].y-a[a[m].near[i]].y)*temp;
            b[a[m].near[i]].vz+=(a[m].z-a[a[m].near[i]].z)*temp;
        }
    }


    int move(molecule a[],molecule b[],int m,int room1[][wid][hei],int room2[][wid][hei])//分子动力学模拟
    {

        b[m].x=a[m].x+(a[m].vx+b[m].vx)*0.5*dt;
        b[m].y=a[m].y+(a[m].vy+b[m].vy)*0.5*dt;
        b[m].z=a[m].z+(a[m].vz+b[m].vz)*0.5*dt;
        int xx=b[m].x;
        int yy=b[m].y;
        int zz=b[m].z;

        if(zz<=0)//边界条件（反弹）
        {
            b[m].vz*=-1;//速度反向
            b[m].z=1.0001;//位置紧贴容器壁
            zz=b[m].z;
        }
        if(zz>=hei)
        {
            px+=a[m].vz;//测量瞬时压强，间接测量声波
            b[m].vz*=-1;
            b[m].z=hei-1.99;
            zz=b[m].z;
        }
        if(xx>=len-1)
        {
            b[m].vx*=-1;
            b[m].x=len-1.999;
            xx=b[m].x;
        }
        if(xx<=0)
```

```
        {
            b[m].vx*=-1;
            b[m].x=1.0001;
            xx=b[m].x;
        }
        if(yy>=wid-1)
        {
            b[m].vy*=-1;
            b[m].y=wid-1.999;
            yy=b[m].y;
        }
        if(yy<=0)
        {
            b[m].vy*=-1;
            b[m].y=1.0001;
            yy=b[m].y;
        }
        room2[xx][yy][zz]=m;
        b[m].location[0]=xx;
        b[m].location[1]=yy;
        b[m].location[2]=zz;

        return 1;
}

void zeroroom(int room[][wid][hei])//清空空间
{
        for(int i=0;i<len;i++)
        {
            for(int j=0;j<wid;j++)
            {
                for(int k=0;k<hei;k++)
                {
                    room[i][j][k]=-1;
                }
            }
        }
}

double w(const double x,const double Alpha)//分布函数
{
        return x*x*exp(-Alpha*x*x/k/2);
}
```

```
double RandomExp(const double Alpha)
{

    double x=20000.0*rand()/RAND_MAX-10000.0*RAND_MAX;//范围尽量大，因为真实情况下光速才是速率上限
    double wx=w(x,Alpha);
    for(int i=10;i>0;i--)//减弱相关度
    {
        const double dx=40000.0*rand()/RAND_MAX-20000.0*RAND_MAX;

        double newx=x+dx;
        while(newx>10000.0)//边界条件
                newx-=20000.0;
        while(newx<-10000.0)
            newx+=20000.0;

        const double wnewx=w(newx,Alpha);

        if(wnewx*RAND_MAX>wx*rand())
        {
            x=newx;
            wx=wnewx;
        }

    }

    return x;
}

void initialize(molecule a[],molecule b[],int room1[][wid][hei],int room2[][wid][hei],const double Alpha)//利用蒙特卡罗进行对气体的初始化，使其满足速率分布
{
    zeroroom(room2);
    cout<<"start"<<endl;
    for(int i=0;i<N;i++)
    {
        b[i].vx=0;
        b[i].vy=0;
        b[i].vz=0;
        a[i].n=b[i].n=0;
        double v=RandomExp(Alpha);
        double c=2.0*rand()/RAND_MAX-1;
        double s=pow(1-c*c,0.5);
        double fei=2.0*M_PI*rand()/RAND_MAX-M_PI;
```

```
        a[i].vx=v*s*cos(fei);
        a[i].vy=v*s*sin(fei);
        a[i].vz=v*c;
        if(i<2500)a[i].vz+=500;//敲击产生声音
    }
    int n=0;
    for(int z=0;z<hei;z++)
        for(int y=0;y<wid;y++)
            for(int x=0;x<len;x++)
            {
                if(x&&y&&z&&(z%5==0)&&(y%5==0)&&(x%5==0)&&n<N)//初始化位置
                {
                    a[n].x=x;
                    a[n].y=y;
                    a[n].z=z;
                    a[n].location[0]=x;
                    a[n].location[1]=y;
                    a[n].location[2]=z;
                    room1[x][y][z]=n;
                    n++;
                }
                else
                {
                    room1[x][y][z]=-1;
                }
            }
    int coun=0;
    while(coun++<50)
    {
        for(int i=0;i<N;i++)
        {
            scan(a,i,room1);
        }
        for(int i=0;i<N;i++)
        {
            impact(a,b,i);
        }
        for(int i=0;i<N;i++)
        {
            move(a,b,i,room1,room2);
        }
        molecule *temp=a;
        int (*roomtemp)[wid][hei]=room1;
        a=b;
```

```
            b=temp;
            room1=room2;
            room2=roomtemp;
            zero(b);
            zeroroom(room2);
            px=0;
            t+=dt;
        }
        cout<<"Initialization completed!"<<endl;
}

void FFTW(double f[],double g[][2],const int n)//快速傅里叶变换
{
        fftw_plan p=fftw_plan_dft_r2c_1d(n,f,g,0);

        fftw_execute(p);
        fftw_destroy_plan(p);
}

/*double measureT(molecule a[])//温度测量函数
{
        double T=0;
        for(int i=0;i<N;i++)
        {
            if(a[i].mode==1)
                {
                    T+=a[i].vx*a[i].vx+a[i].vy*a[i].vy+a[i].vz*a[i].vz;
                }
        }
        return T*Alpha/nn/3;
}*/
void ZeroArray(double c[][2],const int n)//清空
{
        for(int i=0;i<n;i++)
        {
            c[i][0]=0;
            c[i][1]=0;
        }
}

int main()
{
        ofstream out1(name);
```

```
double (*result)[JC]=new double[500][JC];//实验数据存储
srand((unsigned)time(NULL));
for(int jishu=0;jishu<JC;jishu++)
{
Alpha=m0/T;//组合系数
double f[1000];
double g[1000][2];
double f1[1000];
double g1[1000][2];
ZeroArray(g,1000);
ZeroArray(g1,1000);
molecule *a,*b;
a=new molecule[N];
b=new molecule[N];
int (*room1)[wid][hei]=new int[len][wid][hei];//容器空间
int (*room2)[wid][hei]=new int[len][wid][hei];

initialize(a,b,room1,room2,Alpha);
int count=0;
while(count++<1000)//运动 1000 个单位时间
{
    for(int i=0;i<N;i++)
    {
        scan(a,i,room1);
    }
    for(int i=0;i<N;i++)
    {
        impact(a,b,i);
    }
    for(int i=0;i<N;i++)
    {
        move(a,b,i,room1,room2);
    }
    f[count]=px*m0/1000;
    f1[count]=0;

    t+=dt;
    molecule *temp=a;
    int (*roomtemp)[wid][hei]=room1;
    a=b;
    b=temp;
    room1=room2;
    room2=roomtemp;
    zero(b);
```

```
            zeroroom(room2);
            px=0;
        }


        //T+=50;//变化
        m0+=5;
        delete []a;
        delete []b;
        delete []room1;
        delete []room2;
        FFTW(f1,g1,1000);
        FFTW(f,g,1000);

        for(int i=0;i<500;i++)//后 500 毫无意义全为 0，故只输出前 500
        {
            result[i][jishu]=g[i][1];
        }
        cout<<"完成"<<jishu<<"次！"<<endl;
    }

        cout<<"全部完成，开始输出！"<<endl;
        ofstream out2("datasumA.dat");
        double sum1[JC]={0},sum2[JC]={0},max[JC]={0};
        for(int i=0;i<500;i++)
        {
            for(int j=0;j<JC;j++)
            {
                if(fabs(result[i][j])>max[j])
                    max[j]=fabs(result[i][j]);//找出绝对值最大的峰
            }
        }

        for(int i=0;i<500;i++)
        {
            //out2<<i;
            for(int j=0;j<JC;j++)
            {
                result[i][j]/=max[j];
                result[i][j]*=result[i][j];//利用 y=x^2 锐化
                out2<<i<<"\t"<<j*5+10<<"\t"<<max[j]*result[i][j]<<endl;//输出处理后的频谱
                sum1[j]+=max[j]*result[i][j];
                sum2[j]+=i*max[j]*result[i][j];
                if(i==499)
```

```
                    //out1<<j*50+100<<"\t"<<sum2[j]/sum1[j]<<endl;//输出平均频率
                    out1<<j*5+10<<"\t"<<sum2[j]/sum1[j]<<endl;
                }
                out2<<endl;

            }

            out1.close();
            out2.close();
            return 0;
}
```