Содержание

Начальные распределения для задачи двух тел

Точные формулы для двухатомной системы

Рассмотрим вектор, соединяющий центры атомов. Обозначим $\mathbf{r}$ его координаты в лабораторной системе координат, $\mathbf{R}$ – в молекулярной системе координат. Производные $\mathbf{r}$ и $\mathbf{R}$ связаны при помощи матрицы эйлеровых углов $\mathbb{S}$ и угловой скорости $\mathbf{\Omega}$:

$$\dot{\mathbf{r}} = \mathbb{S}^{-1}\left(\dot{\mathbf{R}} + [\mathbf{\Omega} \times \mathbf{R}]\right). \tag{1}$$

Пусть атомы в молекулярной системе координат расположены на оси Z, в таком случае правая часть выражения (1) превращается в

$$\begin{aligned}\dot{\mathbf{r}} &= \mathbb{S}^{-1}\left\{\begin{bmatrix}0\\0\\\dot{R}\end{bmatrix} + \begin{bmatrix}\Omega_y R\\-\Omega_x R\\0\end{bmatrix}\right\}\\ \mathbb{S}\dot{\mathbf{r}} &= \begin{bmatrix}\Omega_y R\\-\Omega_x R\\\dot{R}\end{bmatrix}.\end{aligned} \tag{2}$$

Лагранжиан в молекулярной системе координат имеет следующий вид:

$$\mathcal{L} = \frac{1}{2}\mu\dot{R}^2 + \frac{1}{2}\mathbf{\Omega}^\top \begin{bmatrix}\mu R^2 & 0 & 0\\0 & \mu R^2 & 0\\0 & 0 & 0\end{bmatrix}\mathbf{\Omega} - U$$

Используя теорему Донкина, находим связь гамильтоновых переменных $\mathbf{J}$ и $\mathbf{p} = [p_R]$ с лагранжевыми переменными $\mathbf{\Omega}$ и $\mathbf{q} = [R]$:

$$\begin{aligned}\mathbf{J} &= \frac{\partial\mathcal{L}}{\partial\mathbf{\Omega}} = \mathbb{I}\,\mathbf{\Omega}\\ \mathbf{p} &= \frac{\partial\mathcal{L}}{\partial\dot{\mathbf{q}}} = \mathbb{a}\,\dot{\mathbf{q}}\end{aligned} \quad\Longrightarrow\quad \begin{aligned}J_x &= \mu R^2\,\Omega_x\\ J_y &= \mu R^2\,\Omega_y\\ p_R &= \mu\dot{R}\end{aligned} \tag{3}$$

Выкладка в приложении A показывает, что каждая компонента $\dot{\mathbf{r}}$ имеет нормальное распределение $\dot{\mathbf{r}} \sim \mathcal{N}\left(\mu = 0, \sigma^2 = \dfrac{kT}{\mu}\right)$.

Выкладка в приложении B показывает, что действие равномерно распределенной матрицы поворота $\mathbb{S}$ на $\dot{\mathbf{r}}$ не приводит к изменению распределения $\dot{\mathbf{r}}$. Воспользуемся этим фактом для получения точных распределений для переменных J_x, J_y и p_R:

$$\mathbb{S}\dot{\mathbf{r}} \sim \dot{\mathbf{r}} \sim \begin{bmatrix}\Omega_y R\\-\Omega_x R\\\dot{R}\end{bmatrix} \quad\Longrightarrow\quad \begin{aligned}\Omega_x R &\sim \mathcal{N}\left(0, \frac{kT}{\mu}\right)\\ \Omega_y R &\sim \mathcal{N}\left(0, \frac{kT}{\mu}\right)\\ \dot{R} &\sim \mathcal{N}\left(0, \frac{kT}{\mu}\right)\end{aligned} \quad\Longrightarrow\quad \begin{aligned}J_x &\sim \mu\,\Omega_x R^2 \sim \mathcal{N}\left(0, kT\mu R^2\right)\\ J_y &\sim \mu\,\Omega_y R^2 \sim \mathcal{N}\left(0, kT\mu R^2\right)\\ p_R &\sim \mu\dot{R} \sim \mathcal{N}\left(0, kT\mu\right)\end{aligned} \tag{4}$$

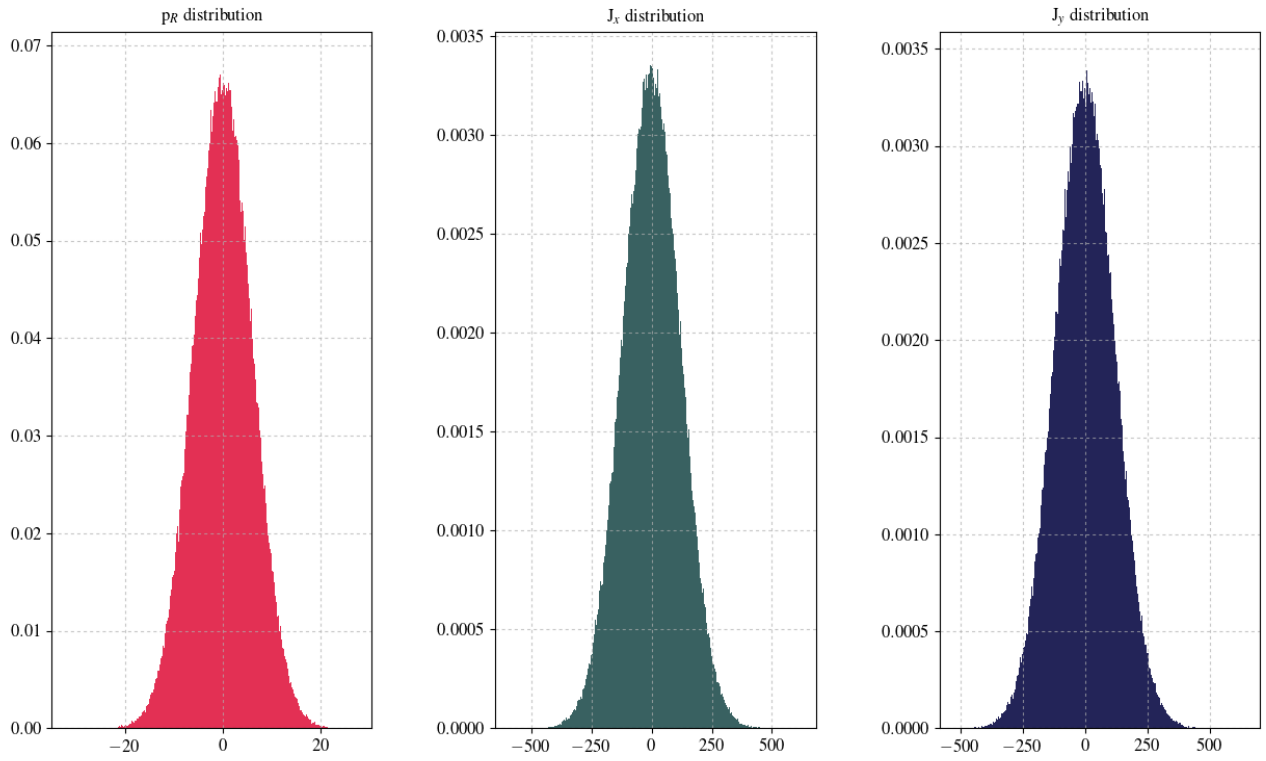

Рис. 1: Распределения переменных p_R, J_x, J_y для двух атомов с массами m_{Ar} и m_{CO_2} при $T = 300K$, 500.000 точек, $R = 20$ бор.

```cpp
#include <iostream>
#include <random>

using namespace std;

// boltzamnn constant
const double BOLTZCONST = 1.38064e-23;
// dalton to kg
const double DALTON = 1.660539e-27;
// atomic length unit to m
const double ALU = 5.29177e-11;

// reduced mass of ar and co2 = m(ar) * m(co2) / (m(ar) + m(co2)) in kg
const double MU = 20.952 * DALTON;

// planck constant
const double HBAR = 1.0545718e-34;

const double temperature = 300;

// distance between atoms
const double RDIST = 20.0;

// a Mersenne Twister pseudo-random generator of 32-bit numbers with a state size of 19937 bits
static thread_local mt19937 generator;
```

```
26
27 double nextGaussian( const double &mean, const double &sigma )
28 {
29     normal_distribution<double> d( mean, sigma );
30     return d( generator );
31 }
32
33 int main( int argc, char* argv[] )
34 {
35   int n = atoi( argv[1] );
36
37   for ( int i = 0; i < n; i++ )
38   {
39     double jx = nextGaussian( 0, RDIST * ALU * sqrt(BOLTZCONST * temperature * MU )
       ) / HBAR;
40     double jy = nextGaussian( 0, RDIST * ALU * sqrt(BOLTZCONST * temperature * MU )
       ) / HBAR;
41     double pR = nextGaussian( 0, sqrt(BOLTZCONST * temperature * MU)) / HBAR * ALU;
42
43     cout << jx << "␣␣" << jy << "␣␣" << pR << endl;
44   }
45
46   return 0;
47 }
```

Пример программы на C++ для генерации значений J_x, J_y и p_R по точным распределениям (4).

Равномерно распределенные матрицы поворота

Следующий алгоритм к получению равномерно распределенных матриц поворота состоит из двух шагов:

1. равномерно распределенный поворот вокруг оси OZ
2. поворот, приводящий к равномерному на сфере положению северного полюса

Первый шаг осуществить легко; пусть случайная величина x_1 равномерно распределена на отрезке $[0, 1]$, тогда матрица R осуществляет равномерно распределенный поворот вокруг оси OZ

$$R = \begin{bmatrix} \cos(2\pi x_1) & \sin(2\pi x_1) & 0 \\ -\sin(2\pi x_1) & \cos(2\pi x_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Второй шаг может быть выполнен при помощи *преобразования Хаусхолдера* (Householder transform). Точка $z = (0, 0, 1)$ может быть перенесена в любую точку сферу при помощи отражения относительно плоскости, перпендикулярной вектору $\overline{zp}$ и проходящей через центр координат O. Такое отражение описывается *Хаусхолдеровской матрицей*

$$H = 1 - 2vv^{\top},$$

где v – единичный вектор, параллельный $\overline{zp}$. Взяв комбинацию хаусхолдеровского отражения и инверсии мы получим поворот, т.к. детерминант такого преобразования будет равен $\det(\cdot) =$

$+1$ (матрица преобразования будет равна $-H$). Таким образом, искомоая матрица поворота равна

$$M = -HR$$

Матрица поворота M будет равномерно распределена внутри $SO(3)$, если H равномерно преобразует Северный полюс в любую точку на сфере, а R описывает равномерный поворот вокруг OZ. Оператор H будет удовлетворять поставленному условию, если мы возьмем

$$v = \begin{bmatrix} \cos(2\pi x_2)\sqrt{x_3} \\ \sin(2\pi x_2)\sqrt{x_3} \\ \sqrt{1-x_3} \end{bmatrix},$$

где x_2, x_3 – равномерно распределены на $[0, 1]$. В таком случае матрица H принимает следующий вид

$$H = 1 - 2vv^\top = \begin{bmatrix} 1 - 2\cos^2(2\pi x_2)\, x_3 & -2\sin(2\pi x_2)\cos(2\pi x_2)\, x_3 & -2\cos(2\pi x_2)\sqrt{x_3(1-x_3)} \\ -2\sin(2\pi x_2)\cos(2\pi x_2)\, x_3 & 1 - 2\sin^2(2\pi x_2)\, x_3 & -2\sin(2\pi x_2)\sqrt{x_3(1-x_3)} \\ -2\cos(2\pi x_2)\sqrt{x_3(1-x_3)} & -2\sin(2\pi x_2)\sqrt{x_3(1-x_3)} & 2x_3 - 1 \end{bmatrix}$$

Действие H на вектор z приводит к вектору p, компоненты которого равны

$$p = Hz = \left(1 - 2vv^\top\right)\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -2\cos(2\pi x_2)\sqrt{x_3(1-x_3)} \\ -2\sin(2\pi x_2)\sqrt{x_3(1-x_3)} \\ 2x_3 - 1 \end{bmatrix}$$

Заметим, что если положить $\sin\phi = -2\sqrt{x_3(1-x_3)}$, то тогда $\cos\phi = 2x_3 - 1$. Действительно,

$$\sin^2\phi + \cos^2\phi = \left[-2\sqrt{x_3(1-x_3)}\right]^2 + [2x_3 - 1]^2 = 1$$

То есть, вектор p может быть представлен в следующей форме

$$p = \begin{bmatrix} \cos(2\pi x_2)\sin\varphi \\ \sin(2\pi x_2)\sin\varphi \\ \cos\varphi \end{bmatrix} = \begin{bmatrix} \cos(2\pi x_2)\sqrt{z} \\ \sin(2\pi x_2)\sqrt{z} \\ \sqrt{1-z} \end{bmatrix}.$$

Следовательно p равномерно распределен на сфере, т.к. азимутальный угол и косинус полярного угла $\cos\varphi = 2x_3 - 1$ распределены равномерно на $[-1, 1]$. Для упрощения компонент вектора переобозначим $\sqrt{z} = \sqrt{x_3(1-x_3)}$, $\sqrt{1-z} = 2x_3 - 1$. Итак, схема алгоритма представлена ниже.

Algorithm 1 Generation of random rotation matrices [3]

1: x_1, x_2, $x_3 \leftarrow$ 3 random variables uniformly distributed over $[0, 1]$
2: Pick a rotation about the pole: $\theta \leftarrow 2\pi x_1$
3: Pick a direction to deflect the pole: $\phi \leftarrow 2\pi x_2$
4: Pick the amount of pole deflection: $z \leftarrow x_3$.
5: Construct a vector to perform the reflection: v $= \begin{bmatrix} \cos\phi\sqrt{z} \\ \sin\phi\sqrt{z} \\ \sqrt{1-z} \end{bmatrix}$
6: Construct the rotation matrix by combining two simple rotations: first rotate about the Z-axis, then rotate the Z-axis to a random orientation: $M \leftarrow \left(2vv^\top - 1\right)\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

```cpp
#include <iostream>
#include <random>
#include <Eigen/Dense>

using namespace std;
using namespace Eigen;

//a Mersenne Twister pseudorandom generator of 32 bit numbers with a state size of 19937 bits
random_device rd;
mt19937 eng( rd() );
uniform_real_distribution<double> distr( 0.0, 1.0);

void rotationMatrix( Matrix3d &m )
{
  double theta = 2 * M_PI * distr( eng ); // a rotation about the pole
  double phi = 2 * M_PI * distr( eng ); // a direction to deflect the pole
  double z = distr( eng ); // the amount of pole deflection
  double sz = sqrt( z );

  // a vector to perform the reflection
  Vector3d v ( cos(phi) * sz, sin(phi) * sz, sqrt(1 - z) );
  // the Householder matrix
  Matrix3d s = 2 * v * v.transpose() - Matrix<double, 3, 3>::Identity();

  Matrix3d r;
    r << cos(theta), sin(theta), 0,
    -sin(theta), cos(theta), 0,
    0, 0, 1;

  m = s * r;
}

int main( int argc, char* argv[] )
{
  int n = atoi( argv[1] );

  // initial vector
  Vector3d v ( 0.0, 0.0, 1.0 );
  // resulting vector
  Vector3d r;

  for ( int i = 0; i < n; i++ )
  {
    // filling rotation matrix
      Matrix3d m;
      rotationMatrix( m );

    // performing a random rotation of OZ-vector
    r = m * v;

    // displaying the components of resulting vector
    cout << r(0) << " " << r(1) << " " << r(2) << endl;
  }

    return 0;
```

```
56 }
```

Пример программы на C++ с применением библиотеки линейной алгебры *Eigen*. Программа принимает на вход количество рассчитываемых векторов n. Внутри главного цикла генерируется по описанному алгоритму случайная матрица поворота и применяется для поворота вектора $v = [0, 0, 1]$. На выходе получаем n равномерно распределенных на сфере векторов.

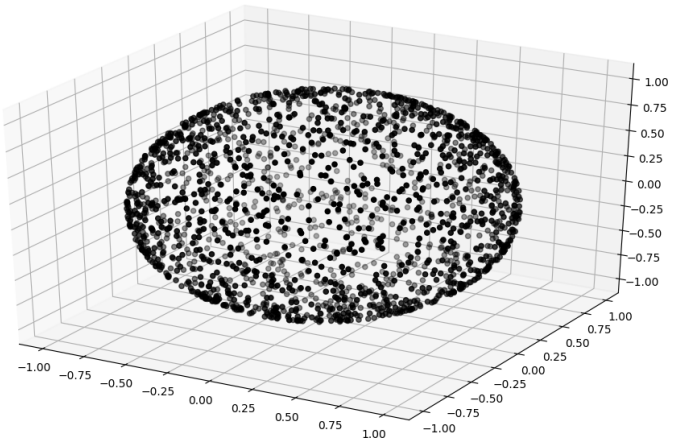

Рис. 2: Пример равномерно распределенных на сфере точек, полученных в результате приведенной выше программы. 2000 точек.

MCMC-sampling

Как известно, в каноническом ансамбле плотность вероятности в фазовом пространстве определяется гамильтонианом [4]

$$\rho\left(\mathbf{q}, \mathbf{p}, \mathbf{J}\right) \propto \exp\left(-\frac{H\left(\mathbf{q}, \mathbf{p}, \mathbf{J}\right)}{kT}\right)$$

Таким образом, задача получения распределений $\mathbf{q}, \mathbf{p}, \mathbf{J}$ может быть рассмотрена как задача сэмплинга точек в фазовом пространстве согласно известной плотности вероятности. Попытаемся решить эту задачу при помощи метода *Markov Chain Monte Carlo* (MCMC); суть метода заключается в построении Марковской цепи в фазовом пространстве, стационарное распределение которой совпадает с целевым распределением. Фактически, в нашем случае мы будем рассматривать не полное фазовое пространство, а его сечение – $R = const$.

Предположим мы генерируем последовательность случайных величин, $\{X_0, X_1, X_2, \dots\}$, так что в каждый момент $t \geq 0$ следующее состояние X_{t+1} выбирается исходя из распределения $P\left(X_{t+1}|X_t\right)$, которое зависит от текущего состояния X_t, но не от предыдущего набора состояний $\{X_0, X_1, X_2 ... X_{t-1}\}$. То есть, состояние X_{t+1} определяется исключительно предыдущим X_t. Такая последовательность состояний называется *цепью Маркова*.

Перейдем к рассмотрению алгоритма Метрополиса-Гастингса, представляющего собой один из простейших алгоритмов для построения Марковской цепи с заданным стационарным распределением $\pi\left(\cdot\right)$.

Algorithm 2 Scheme of Metropolis-Hastings algorithm from [1]

Initialize $x^{(0)} \sim q(x)$
for iteration $i = 1, 2, \ldots$ **do**
  Propose: $x^{cand} \sim q\left(x^{(i)}|x^{(i-1)}\right)$
  Acceptance probability:
    $\alpha\left(x^{cand}|x^{(i-1)}\right) = \min\left\{1, \frac{q\left(x^{(i-1)}|x^{cand}\right)\pi\left(x^{(cand)}\right)}{q\left(x^{cand}|x^{(i-1)}\right)\pi\left(x^{(i-1)}\right)}\right\}$
  $u \sim$ Uniform(u; 0, 1)
  if $u < \alpha$ **then**
    Accept the proposal: $x^{(i)} \leftarrow x^{cand}$
  else
    Reject the proposal: $X^{(i)} \leftarrow x^{(i-1)}$
  end if
end for

Первым шагом алгоритма является выбор стартовой точки цепи, определенных методик, насколько я понимаю, здесь нет; в целом важно не задавать ее в какой-то "нефизичной"области фазового пространства (в той, в которой очень маловероятно застать рассматриваемую систему). Следующий за ним главный цикл алгоритма состоит из трех частей: (1) получить следующую точку ("кандидата") x^{cand} исходя из вспомогательного распределения $q\left(x^{(i)}|x^{(i-1)}\right)$; (2) рассчитать вероятность перехода в новую точку $\alpha\left(x^{cand}|x^{(i-1)}\right)$, основываясь на распределении q и функции распределения π; (3) принять новую точку с вероятностью α.

Обратим внимание на то, что точка, полученная исходя из вспомогательного распределения $q(\cdot)$, принимается не всегда, а лишь с вероятностью $\alpha(\cdot)$. Рассматривают вспомогательные распределения двух классов – симметричные и асимметричные. Симметричным называется распределение, удовлетворяющее следующему соотношению

$$q\left(x^{(i)}|x^{(i-1)}\right) = q\left(x^{(i-1)}|x^{(i)}\right)$$

К часто используемым симметричным распределениям относятся гауссово и равномерное распределения. В качестве примера рассмотрим вспомогательное распределение Гаусссса:

$$x^{cand} = x^{(i-1)} + Normal(0, \sigma)$$

Понятно, что $Normal(x^{cand} - x^{(i-1)}; 0, \sigma) = Normal(x^{(i-1)} - x^{cand}; 0, \sigma)$, то есть Гауссово распределение в действительности задает симметричное вспомогательное распределение. Среднеквадратичное отклонение σ является параметром модели. Значение этого параметра будет определять динамику Марковской цепи в рассматриваемом пространстве.

В случае симметричных вспомогательных распределений выражение для вероятности выбора новой точки $\alpha(\cdot)$ существенно упрощается:

$$\alpha\left(x^{cand}|x^{(i-1)}\right) = \min\left\{1, \frac{\pi\left(x^{cand}\right)}{\pi\left(x^{(i-1)}\right)}\right\}$$

Заметим, что если плотность вероятности (точнее говоря, величина, пропорциональная плотности вероятности) в новой точке $\pi\left(x^{cand}\right)$ больше, чем плотность вероятности в текущей $\pi\left(x^{(i-1)}\right)$, то их отношение будет больше 1, а значит вероятность перехода в новую точку будет равна 1: $\alpha\left(x^{cand}|x^{(i-1)}\right) = 1$. Другими словами, если новая точка выбрана таким образом,

что плотность вероятности в ней больше, чем в текущей, то в нее осуществляется переход. Устройство алгоритма таково, что Марковская цепь "склонна"посещать те точки пространства, в которых моделируемая плотность вероятности выше. Однако, если новая точка была выбрана таким образом, что плотность вероятности в ней меньше, чем в текущей, то тогда вероятность перейти в нее будет определяться отношением плотностей вероятности:

$$\alpha\left(x^{cand}|x^{(i-1)}\right) = \frac{\pi\left(x^{cand}\right)}{\pi\left(x^{(i-1)}\right)}$$

То есть, если вероятность в новой точке будет мала по сравнению с текущей, то и переход в нее будет маловероятен. Заметим, что при подсчете вероятности нам нужно знать лишь отношение плотностей вероятности, а не ее абсолютное значение; поэтому в качестве $\pi(\cdot)$ можно брать не плотность вероятности, а величину ей пропорциональную.

Вид вероятности перехода в новую точку из текущей определяется *условием детальноого баланса* [2]. Последнее гарантирует, что полученная Марковская цепь в действительности будет удовлетворять заданной плотности вероятности.

Представим в качестве примера основу кода для реализации Марковской цепи для двухатомной системы.

```
1  #include <iostream>
2  #include <random>
3  #include <ctime>
4  #include <cmath>
5
6  #include <iomanip> // std::atoi
7  #include <algorithm> // std::min
8
9  #include <Eigen/Dense>
10
11 using namespace std;
12 using namespace Eigen;
13
14 // boltzmann constant
15 const double BOLTZCONST = 1.38064e-23;
16 // unified atomic mass units to kg
17 const double DALTON = 1.660539e-27;
18 // atomic length unit
19 const double ALU = 5.29177e-11;
20 // atomic mass unit
21 const long double AMU = 9.1093826e-31;
22 // hartree to joules
23 const double HTOJ = 4.35974417e-18;
24
25 // reduced mass of ar and co2 = m(ar) * m(co2) / ( m(ar) + m(co2) ) in kg
26 const double MUKG = ( 40.0 * 44.0 ) / ( 40.0 + 44.0 ) * DALTON;
27 const long double MUAMU = MUKG / AMU;
28
29 // temperature in K
30 const double temperature = 300;
31
32 static mt19937 generator;
33
34 // distance between two atoms in ALU
35 const double RDIST = 20.0;
```

```
36
37 static double nextDouble( const double &min = 0.0, const double &max = 1.0 )
38 {
39     uniform_real_distribution<double> distribution( min, max );
40     return distribution( generator );
41 }
42
43 // fills given vector V with random gaussian variables with given mean and sigma
44 static void nextGaussianVec( Vector3d &v, Vector3d mean, const double sigma )
45 {
46   for ( int i = 0; i < 3; i++ )
47   {
48     normal_distribution<double> d( mean(i), sigma );
49     v(i) = d( generator );
50   }
51 }
52
53 // x = [jx, jy, pR]
54 // the target distribution function we wish to sample
55 double target( Vector3d x )
56 {
57     double jx = x(0);
58     double jy = x(1);
59     double pR = x(2);
60     double h = pow(pR, 2) / ( 2 * MUAMU ) + ( pow(jx, 2) + pow(jy, 2) ) / (2 *
          MUAMU * pow(RDIST, 2)); //  + potential( RDIST );
61     return exp( - h * HTOJ / ( BOLTZCONST * temperature ));
62 }
63
64 // step of Metropolis-Hastings algroithm
65 Vector3d metro_step( Vector3d x, double alpha )
66 {
67     Vector3d prop;
68
69     // generating random gaussian vector
70     nextGaussianVec( prop, x, alpha );
71
72     if ( nextDouble() < min( 1.0, target(prop) / target(x) ))
73     {
74         x = prop;
75     }
76
77     return x;
78 }
79
80 int main( int argc, char* argv[] )
81 {
82     const int nsteps = atoi( argv[1] );
83     const int burnin = atoi( argv[2] );
84     const double alpha = atof( argv[3] );
85     const bool show_vecs = atoi( argv[4] );
86
87     Vector3d x ( 10.0, 10.0, 10.0 );
88     Vector3d xnew;
89     int moves = 0;
90
```

```
91      for ( int i = 0; i < nsteps + burnin; i++ )
92      {
93          xnew = metro_step( x, alpha );
94
95          if ( xnew != x )
96          {
97              moves++;
98
99              if ( i > burnin && show_vecs == true )
100             {
101                 cout << x(0) << " " << x(1) << " " << x(2) << endl;
102             }
103         }
104
105         x = xnew;
106     }
107
108     cerr << "total steps: " << nsteps + burnin << "; moves: " << moves << "; "
        percent: " << (double) moves / ( nsteps + burnin ) * 100 << "%" << endl;
109
110     return 0;
111 }
```

В приведенной программе использована переменная *burnin*, в которой задается количество шагов, которое цепь Маркова строится "вхолостую"(без сохранения шагов). Это делается для того, чтобы избежать влияния особенности выбора начального положения цепи. Следующие за *burnin* шаги выводятся в консоль. Сравнение распределений, полученных по алгоритму MH и по точным формулам, представлено на рис. 3.

Parallel Metropolis-Hastings

...

Начальные распределения для CO_2-Ar

Точные формулы для распределений

Из классической механики известно, что при отделении центра масс в системе «атом + линейная молекула» образуются две эффективные частицы, где первая частица движется на расстоянии l и имеет массу $\mu_1 = \frac{m_1 m_2}{m_1+m_2}$, а вторая, с массой $\mu_2 = \frac{(m_1+m_2)m_3}{m_1+m_2+m_3}$ движется на расстоянии R, где R – расстояние между центром масс линейной молекулы и атомом, l – длина линейной молекулы.

Тогда для связи векторов в ЛСК и МСК у нас получатся слеующие выражения:

$$\begin{gathered} \mathbb{S}\,\dot{\mathbf{r}}_1 = \left\{ \begin{pmatrix} l\cos(\Theta)\dot{\Theta} \\ 0 \\ -l\sin(\Theta)\dot{\Theta} \end{pmatrix} + \begin{pmatrix} l\Omega_y\cos(\Theta) \\ -l\Omega_x\cos(\Theta) + l\Omega_z\sin(\Theta) \\ -l\Omega_y\sin(\Theta) \end{pmatrix} \right\} \\ \mathbb{S}\,\dot{\mathbf{r}}_2 = \left\{ \begin{pmatrix} 0 \\ 0 \\ \dot{R} \end{pmatrix} + \begin{pmatrix} \Omega_y R \\ -\Omega_x R \\ 0 \end{pmatrix} \right\} \end{gathered} \tag{6}$$

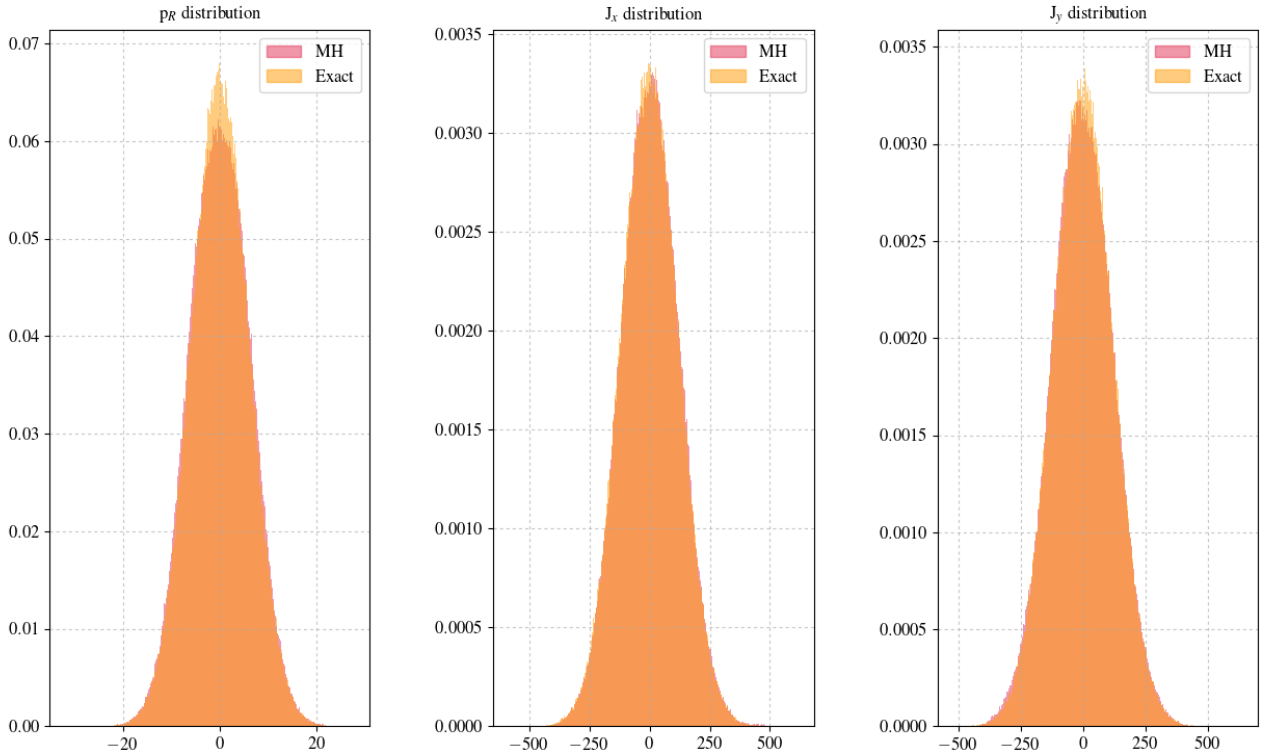

Рис. 3: Распределения переменных p_R, J_x, J_y для двух атомов с массами m_{Ar} и m_{CO_2} при $T = 300K$, полученные методом MH и по точным формулам, 500.000 точек, $R = 20$ бор.

причем $\mathbb{S}$ – ортогональная матрица поворота, например, матрица углов Эйлера. Для упрощения дальнейших выкладок тильдой пометим компоненты векторов, получающиеся при действии матрицы $\mathbb{S}$ на вектора скорости в лабораторной системе координат.

$$\begin{aligned}
\mathbb{S}\,\dot{\mathbf{r}}_1 = \mathbb{S}\begin{pmatrix}\dot{x}_1\\ \dot{y}_1\\ \dot{z}_1\end{pmatrix} &= \begin{pmatrix}S_{11}\dot{x}_1 + S_{12}\dot{y}_1 + S_{13}\dot{z}_1\\ S_{21}\dot{x}_1 + S_{22}\dot{y}_1 + S_{23}\dot{z}_1\\ S_{31}\dot{x}_1 + S_{32}\dot{y}_1 + S_{33}\dot{z}_1\end{pmatrix} = \begin{pmatrix}\tilde{r}_{11}\\ \tilde{r}_{12}\\ \tilde{r}_{13}\end{pmatrix}\\
\mathbb{S}\,\dot{\mathbf{r}}_2 = \mathbb{S}\begin{pmatrix}\dot{x}_2\\ \dot{y}_2\\ \dot{z}_2\end{pmatrix} &= \begin{pmatrix}S_{11}\dot{x}_2 + S_{12}\dot{y}_2 + S_{13}\dot{z}_2\\ S_{21}\dot{x}_2 + S_{22}\dot{y}_2 + S_{23}\dot{z}_2\\ S_{31}\dot{x}_2 + S_{32}\dot{y}_2 + S_{33}\dot{z}_2\end{pmatrix} = \begin{pmatrix}\tilde{r}_{21}\\ \tilde{r}_{22}\\ \tilde{r}_{23}\end{pmatrix}
\end{aligned} \tag{7}$$

$$\mathcal{L} = \frac{\mu_1}{2}\dot{\mathbf{r}}_1^2 + \frac{\mu_2}{2}\dot{\mathbf{r}}_2^2 - U(R,\Theta) \tag{8}$$

Запишем (6) в качестве системы уравнений, с учетом (7).

$$\begin{cases}
\tilde{r}_{11} = l\dot{\Theta}\cos\Theta + l\Omega_y\cos\Theta\\
\tilde{r}_{12} = -l\Omega_x\cos\Theta + l\Omega_z\sin\Theta\\
\tilde{r}_{13} = -l\dot{\Theta}\sin\Theta - l\Omega_y\sin\Theta\\
\tilde{r}_{21} = \Omega_y R\\
\tilde{r}_{22} = -\Omega_x R\\
\tilde{r}_{21} = \dot{R}
\end{cases} \tag{9}$$

Предполагая, что в (9) 1-е и 3-е уравнения эквивалентны (в конце документа рассмотрен случай отсутствия этого предположения), найдем выражения для скоростей в МСК, решая данную систему линейных уравнений:

$$\begin{cases} \dot{R} = \tilde{r}_{23} \\ \dot{\Theta} = -\dfrac{1}{l\sin\Theta}\left(\tilde{r}_{13} + \dfrac{1}{R}\tilde{r}_{21}\, l\sin\Theta\right) \\ \Omega_x = -\dfrac{1}{R}\tilde{r}_{22} \\ \Omega_y = \dfrac{1}{R}\tilde{r}_{21} \\ \Omega_z = \dfrac{1}{\sin\Theta}\left(\tilde{r}_{12} - \dfrac{1}{R}\tilde{r}_{22} l\cos\Theta\right) \end{cases} \tag{10}$$

Мы не будем приводить здесь вывод, однако несложно показать, что гамильтоновы переменные связаны с лагранжевыми переменными для рассматриваемой системы следующим образом:

$$\begin{cases} p_R = \mu_2\dot{R} \\ p_\Theta = \Omega_y l^2\mu_1 + l^2\mu_1\dot{\Theta} \\ J_x = -\mu_1 l^2\Omega_z\sin\Theta\cos\Theta + \Omega_x\left(\mu_1 l^2\cos^2\Theta + \mu_2 R^2\right) \\ J_y = \mu_1 l^2\dot{\Theta} + \Omega_y\left(\mu_2 R^2 + \mu_1 l^2\right) \\ J_z = -l^2\mu_1\sin\Theta\left(\Omega_x\cos\Theta - \Omega_z\sin\Theta\right) \end{cases} \tag{11}$$

Теперь подставим выражения для скоростей в выражения для импульсов.

$$\begin{cases} p_R = \mu_2\tilde{r}_{23} \\ p_\Theta = -\dfrac{\mu_1 l}{\sin\Theta}\tilde{r}_{13} \\ J_x = -\mu_2 R\tilde{r}_{22} - \mu_1 l\tilde{r}_{12}\cos\Theta \\ J_y = \mu_2 R\tilde{r}_{21} - \dfrac{\mu_1 l}{\sin\Theta}\tilde{r}_{13} \\ J_z = \mu_1 l\tilde{r}_{12}\sin\Theta \end{cases} \tag{12}$$

Рассмотрим вопрос о распределении компонент векторов $\dot{\mathbf{r}}_1$ и $\dot{\mathbf{r}}_2$. Представим вектор $\dot{\mathbf{r}}_1$ в лабораторной системе в сферических координатах $\begin{bmatrix} l, & \phi, & \theta \end{bmatrix}$:

$$\dot{\mathbf{r}}_1 = \frac{d}{dt}\begin{bmatrix} l\sin\theta\cos\phi \\ l\sin\theta\sin\phi \\ l\cos\theta \end{bmatrix} = \begin{bmatrix} l\dot{\theta}\cos\theta\cos\phi - l\dot{\phi}\cos\theta\sin\phi \\ l\dot{\theta}\cos\theta\sin\phi + l\dot{\phi}\sin\theta\cos\phi \\ -l\dot{\theta}\sin\theta \end{bmatrix}$$

Нетрудно видеть, что вектора $\dot{\boldsymbol{\theta}}$ и $\dot{\boldsymbol{\phi}}\sin\phi$ направлены перпендикулярно оси вращения линейной молекулы, а значит, являются угловыми скоростями вращения линейной молекулы, поэтому распределены как $\mathcal{N}(0, kT/I)$, где I – момент инерции линейной молекулы.

Ради интереса отметим, что будет, если в системе (9) не считать 1-е и 3-е уравнения эквивалентными. Для этого умножим первое уравнение на $\sin\Theta$, а второе на $\cos\Theta$, после этого получим

$$\sin\Theta\,\tilde{r}_{11} + \cos\Theta\,\tilde{r}_{13} = 0$$

$$\cot\Theta = -\frac{\tilde{r}_{13}}{\tilde{r}_{11}}$$

Если $\tilde{r}_{11}$ и $\tilde{r}_{13}$ нормально распределенные случайные величины, то $\cot\Theta$ имеет распределение Коши.

Заметим, что алгоритм МН для получения начальных условий может быть применен как только имеется процедура расчета гамильтониана в произвольной точке фазового пространства. Распределения для Θ, p_R, p_T, J_x, J_y, J_z, полученные по точным формулам и при помощи алгоритма МН, представлены на рис. 4. Распределение угла Θ (как угла между двумя равномерно на сфере распределенными векторами) имеет плотность вероятности $f(x) = \frac{1}{2}\sin x$ (на рисунке представлено пунктирной линией), это показано в приложении D.

Веса траекторий при расчете спектра поглощения

Рассмотрим случайный вектор начальных условий $\boldsymbol{\xi} = (\Theta, p_R, p_\Theta, J_X, J_Y, J_Z)$. Очевидно, что отношение частот появления начальных условий, описываемых вектором $\boldsymbol{\xi}$ при двух разных температурах T_1 и T_2 будет определяться соотношением:

$$\lambda = \frac{P(\boldsymbol{\xi}, T_1)}{P(\boldsymbol{\xi}, T_2)} = \frac{\exp\left(-H\left(\boldsymbol{\xi}\right)/kT_1\right)}{\exp\left(-H\left(\boldsymbol{\xi}\right)/kT_2\right)}$$

Спектральная функция вычисляется по траекториям следующим образом:

$$J(\omega) = \frac{1}{N}\sum_{i=1}^{N}\hat{A}\left(\mu_i(t)\right) \tag{13}$$

где N – число траекторий, а $\hat{A}$ – определенного вида операция, производимая для дипольного момента на каждой траектории (квадрат модуля преобразования Фурье). Тогда вполне очевидно, что для пересчета спектральной плотности на другую температуру мы должны домножить каждый член суммы в числителе (13) на соответствующий коэффициент пересчета λ_i, не забыв учесть нормировку в знаменателе.

$$J(\omega, T_2) = \frac{\sum_{i=1}^{N}\lambda_i\hat{A}\left(\mu_i(t)\right)}{\sum_{i=1}^{N}(\lambda_i \cdot 1)} \tag{14}$$

Для наглядности продемонстируем вышесказанное на простом примере. Пусть есть две траектории, причем коэффициент λ для первой оказался равен 0.5, а для второй – 2

$$J(\omega, T_1) = \frac{\hat{A}(\mu_1(t)) + \hat{A}(\mu_2(t))}{2} = \frac{2\hat{A}(\mu_1(t)) + 2\hat{A}(\mu_2(t))}{4} = \frac{\hat{A}(\mu_1(t)) + \hat{A}(\mu_1(t)) + \hat{A}(\mu_2(t)) + \hat{A}(\mu_2(t))}{4}$$

Теперь если для первой траектории коэффициент равен 0.5, то это означает, что при температуре T_2 начальное условие, соответствующее данной траектории будет встречаться в 2 раза

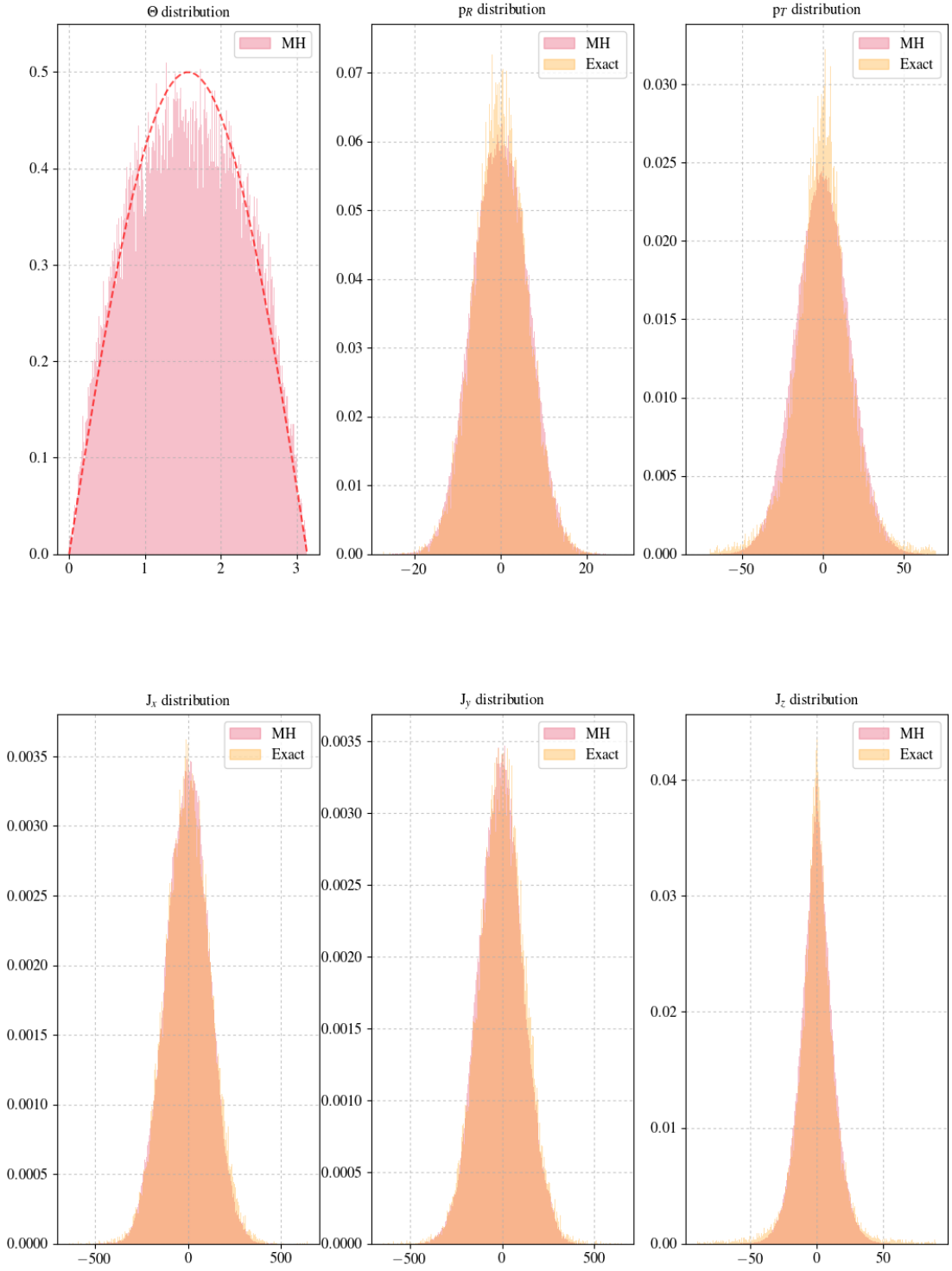

Рис. 4: Распределения переменных Θ, p_R, p_T (**первый ряд**), J_x, J_y, J_z (**второй ряд**) для CO_2-Ar при $T = 300K$, полученные методом MH и по точным формулам, 500.000 точек, R = 20 бор.

меньше, чем при температуре T_1, а это означает, что в сумме (13) мы в 2 раза реже будем писать преобразование от первой траектории, то есть таких членов в сумме будет в два раза меньше. Аналогичные рассуждения применяются и для второй траектории. В итоге:

$$J(\omega, T_2) = \frac{\hat{A}(\mu_1(t)) + 4\hat{A}(\mu_2(t))}{5} = \frac{0.5\hat{A}(\mu_1(t)) + 2\hat{A}(\mu_2(t))}{2.5}$$

что соответствует формуле (14).

Литература

1. Yildirim I. Bayesian Inference: Metropolis-Hastings Sampling. MIT Online Library
2. Gilks, W.R., Richardson, S., & Spiegelhalter, D.J. (1996). *Markov Chain Monte Carlo in Practice*. London: Chapman and Hall.
3. Kirk D. Graphics Gems III. III. 4. Fast random rotation matrices.
4. Н. А. Смирнова. Методы статистической термодинамики в физической химии. (гл. 4)
5. Marsaglia G. (1972). "Choosing a Point from the Surface of a Sphere". *Annals of Mathematical Statistics*. **42** (2): 645-645.

Appendices

Приложение А. Распределения в лабораторной системе координат

Воспользуемся следующими двумя выводами из теории вероятностей:

1. Пусть случайная величина ξ распределена с плотностью $f_\xi(x)$. Тогда случайная величина $\eta = a\xi + b$ распределена с плотностью

$$f_\eta(x) = \frac{1}{|a|} f_\xi\left(\frac{x-b}{a}\right)$$

2. Если две независимые случайные величины X и Y распределены с плотностями $X \sim f_1(x)$ и $Y \sim f_2(x)$ соответственно, то случайна величина $Z = X + Y$ распределена с плотностью

$$g(z) = \int\limits_{-\infty}^{+\infty} f_1(x) f_2(z-x) dx$$

Т.к. вектор $\mathbf{r} = \mathbf{r}_1 - \mathbf{r}_2$ равен разнице радиус-векторов двух атомов $\mathbf{r}_1$ и $\mathbf{r}_2$ в лабораторной системе координат соответственно, то $\dot{\mathbf{r}} = \dot{\mathbf{r}}_1 - \dot{\mathbf{r}}_2$. Используя п.1 и п.2 получим распределение для компонент $\mathbf{r}$:

$$\begin{cases} \dot{\mathbf{r}}_{1x} \sim f_1(x) = \sqrt{\dfrac{m_1}{2\pi kT}} \exp\left(-\dfrac{m_1 x^2}{2kT}\right) \\ -\dot{\mathbf{r}}_{2x} \sim f_2(x) = \sqrt{\dfrac{m_2}{2\pi kT}} \exp\left(-\dfrac{m_2 x^2}{2kT}\right) \end{cases}$$

$$\dot{\mathbf{r}}_x \sim \int\limits_{-\infty}^{+\infty} f_1(x) f_2(z-x) dx = \frac{\sqrt{m_1 m_2}}{2\pi kT} \int\limits_{-\infty}^{+\infty} \exp\left(-\frac{m_1 x^2}{2kT}\right) \exp\left(-\frac{m_2 (z-x)^2}{2kT}\right) dx \qquad (15)$$

Отдельно рассмотрим получившийся интеграл:

$$\int\limits_{-\infty}^{+\infty} \exp\left(-\frac{m_1 x^2}{2kT} - \frac{m_2 (z-x)^2}{2kT}\right) dx = \int\limits_{-\infty}^{+\infty} \exp\left(\frac{-(m_1+m_2)x^2 - m_2 z^2 + 2m_2 zx}{2kT}\right) dx =$$

$$= \int\limits_{-\infty}^{+\infty} \exp\left(-\frac{\left(\sqrt{m_1+m_2}x - \frac{m_2}{\sqrt{m_1+m_2}}z\right)^2}{2kT}\right) \exp\left(-\frac{m_2 z^2 - \frac{m_2^2}{m_1+m_2} z^2}{2kT}\right) dx =$$

$$= \left[y = \frac{\sqrt{m_1+m_2}x - \frac{m_2}{\sqrt{m_1+m_2}}z}{\sqrt{2kT}}\right] = \sqrt{\frac{2kT}{m_1+m_2}} \exp\left(-\frac{m_1 m_2}{2(m_1+m_2)kT} z^2\right) \int\limits_{-\infty}^{+\infty} \exp\left(-y^2\right) dy =$$

$$= \sqrt{\frac{2\pi kT}{m_1+m_2}} \exp\left(-\frac{m_1 m_2}{2(m_1+m_2)kT} z^2\right) \qquad (16)$$

Подставляя значение интеграла (16) в выражение для плотности распределения $\dot{\mathbf{r}}_x$ (15), получаем

$$\dot{\mathbf{r}}_x \sim \frac{1}{\sqrt{2\pi kT}}\sqrt{\frac{m_1 m_2}{m_1+m_2}}\exp\left(-\frac{m_1 m_2}{2\left(m_1+m_2\right)kT}z^2\right) = \sqrt{\frac{\mu}{2\pi kT}}\exp\left(-\frac{\mu z^2}{2kT}\right),$$

где через μ была обозначена приведенная масса двухатомной системы $\mu = \dfrac{m_1 m_2}{m_1+m_2}$.

Приложение B. О действии равномерно распределенной матрицы поворота на гауссов вектор

Разобъем задачу на две части – фиксируем некоторое значение равномерно распределенной матрицы $\mathbb{S}$ и рассмотрим ее действие на случайный гауссов вектор $\mathbf{X}$ (такой подход обоснован формулой полной вероятности). Обозначим случайную величину, являющуюся результатом действия $\mathbb{S}$ на $\mathbf{X}$ за $\mathbf{Y}$.

$$\mathbf{Y} = \mathbb{S}\mathbf{X}$$

В силу линейности мат. ожидания:

$$\mathbb{E}\left[\mathbf{Y}\right] = \mathbb{S}\mathbb{E}\left[\mathbf{X}\right] = \mathbf{0}$$

Матрица ковариации исходного вектора $\mathbf{X}$ кратна единичной матрице за счет того, что компоненты вектора независимы и дисперсии компонент равны

$$Cov\left(\mathbf{X}\right) = \begin{bmatrix} Var\left(\mathbf{X}_1\right) & 0 & 0 \\ 0 & Var\left(\mathbf{X}_2\right) & 0 \\ 0 & 0 & Var\left(\mathbf{X}_3\right) \end{bmatrix} = \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix} = \sigma^2\mathbb{E}.$$

Вычислим ковариацию величины $\mathbf{Y}$ по определению:

$$Cov\left(\mathbf{Y}\right) = \mathbb{E}\left[\left(\mathbf{Y}-\bar{\mathbf{Y}}\right)\left(\mathbf{Y}-\bar{\mathbf{Y}}\right)\right] = \mathbb{E}\left[\mathbf{Y}\mathbf{Y}^\top\right] = \mathbb{S}\,Cov\left(\mathbf{X}\right)\mathbb{S}^\top = \sigma^2\mathbb{E}$$

Таким образом мы показали, что мат.ожидание и дисперсия компонент при действии фиксированной ортогональной матрицы не изменятся. Заметим, что ни мат. ожидание, ни ковариация $\mathbf{Y}$ не зависят от распределения $\mathbb{S}$, поэтому когда мы будем "интегрировать"по $\mathbb{S}$, чтобы учесть наличие распределение по ней, то на итоговой результат для параметров $\mathbf{Y}$ это не повлияет.

Приложение C. О равномерном на сфере распределении

Пусть у нас есть сфера радиуса R с введенной на ней сферической системой координат. Пусть θ – полярный угол, а ϕ – азимутальный. Тогда элемент поверхности сферы

$$dS = R\sin\theta\,d\theta\,d\phi = -R\,d\cos\theta\,d\phi$$

Переходя к малым, но конечным интервалам

$$\Delta S = -R\Delta\left(\cos\theta\right)\Delta\phi$$

Чтобы равномерно распределить точки на сфере, нужно разбить сферу на равные участки поверхности и расставить в каждый элемент по точке. Для этого независимо от значений углов необходимо, чтобы интервалы $\Delta\phi$ и $\Delta\left(\cos\theta\right)$ были равными. Следовательно, равномерно распределив $\cos\theta$ и ϕ и пересчитав с их помощью координаты точки $[x, y, z]$, получим равномерное на сфере распределение.

Приложение D. Угол между равномерно на сфере распределенными векторами

Пусть $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n) \in \mathcal{N}\left(\mathbf{0}, \mathbb{E}\right)$ – нормальный случайный вектор с нулевым средним и единичной ковариационной матрицей. Тогда вектор

$$\mathbf{e} = \frac{\boldsymbol{\xi}}{\sqrt{\xi_1^2 + \cdots + \xi_n^2}}$$

имеет равномерное распределение на n-мерной сфере единичного радиуса (на этом основан алгоритм Marsaglia [5] эффективной генерации точек равномерно распределенных на n-сфере). Заметим также, что n-мерное распределение случайного вектора $\mathbf{e}$ инвариантно относительно поворотов в n-мерном пространстве (это несложно показать, основываясь на подходе, описанном в приложении B).

Рассмотрим два нормальных независимых вектора $\boldsymbol{\xi}$ и $\boldsymbol{\eta}$. Построим, основываясь на определенных векторах, равномерные на сфере вектора $\boldsymbol{e}_1$ и $\boldsymbol{e}_2$. Угол между ними равен скалярному произведению

$$\cos\alpha = \boldsymbol{e}_1 \cdot \boldsymbol{e}_2 = \frac{\xi_1\eta_1 + \cdots + \xi_n\eta_n}{\sqrt{\xi_1^2 + \cdots + \xi_n^2}\sqrt{\eta_1^2 + \cdots + \xi_n^2}}$$

Распределение косинуса будем искать по определению. Пусть $x \in (-1, 1)$, тогда

$$\mathbb{P}\left(\cos\alpha < x\right) = \int_{\mathbb{R}^n} \mathbb{P}\left(\frac{\mathbf{c}\cdot\boldsymbol{\eta}}{||\mathbf{c}||\sqrt{\eta_1^2 + \cdots + \eta_n^2}} < x\right) f_\xi(\mathbf{c})d\mathbf{c} =$$

$$= \int_{\mathbb{R}^n} \mathbb{P}\left(\frac{\eta_1}{\sqrt{\eta_1^2 + \cdots + \eta_n^2}} < x\right) \delta\left(\mathbf{c} = \begin{bmatrix} 1 \\ 0 \\ \cdots \\ 0 \end{bmatrix}\right) f_\xi(\mathbf{c})d\mathbf{c}$$

Здесь сначала была использована независимость векторов $\boldsymbol{\eta}$ и $\boldsymbol{\xi}$, а затем изотропность распределения вектора $\boldsymbol{\eta}$, в силу которой произвольный вектор $\mathbf{c}$ заменен на конкретный вектор $(1, 0, \ldots, 0)$. Дельта-функционал в интеграле справа означает, что распределение по вектору $\mathbf{c}$ было сведено к дельта-распределению. Итак, это выкладка приводит к следующему

$$\mathbb{P}\left(\cos\alpha < x\right) = \mathbb{P}\left(\frac{\eta_1}{\sqrt{\eta_1^2 + \cdots + \eta_n^2}} < x\right),$$

то есть, угол между произвольными векторами на сфере распределен так же, как угол между одним случайным вектором и каким-нибудь фиксированным направлением, в данном случае $(1, 0, \ldots, 0)$.

Однако

$$\phi_1 = \arccos \frac{\eta_1}{\sqrt{\eta_1^2 + \cdots + \eta_n^2}}$$

есть угол-координата в n-мерной сферической системе координат, плотность распределения которой равна

$$f(x) = \frac{\sin^{n-2} x}{\int\limits_0^\pi \sin^{n-2} x dx}, x \in (0, \pi)$$

что при $n = 2$ дает $f(x) = \frac{1}{\pi}$ (равномерное распределение), а при $n = 3$: $f(x) = \frac{1}{2} \sin(x)$.