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

“Экспериментально“ проверено, что действие равномерно распределенной матрицы поворота $\mathbb{S}$ на $\dot{\mathbf{r}}$ не приводит к изменению распределения $\dot{\mathbf{r}}$. Это интуитивно понятно, но строгого доказательства пока нет. Используем этот “экспериментальный“ факт для получения точных распределений для переменных J_x, J_y и p_R:

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

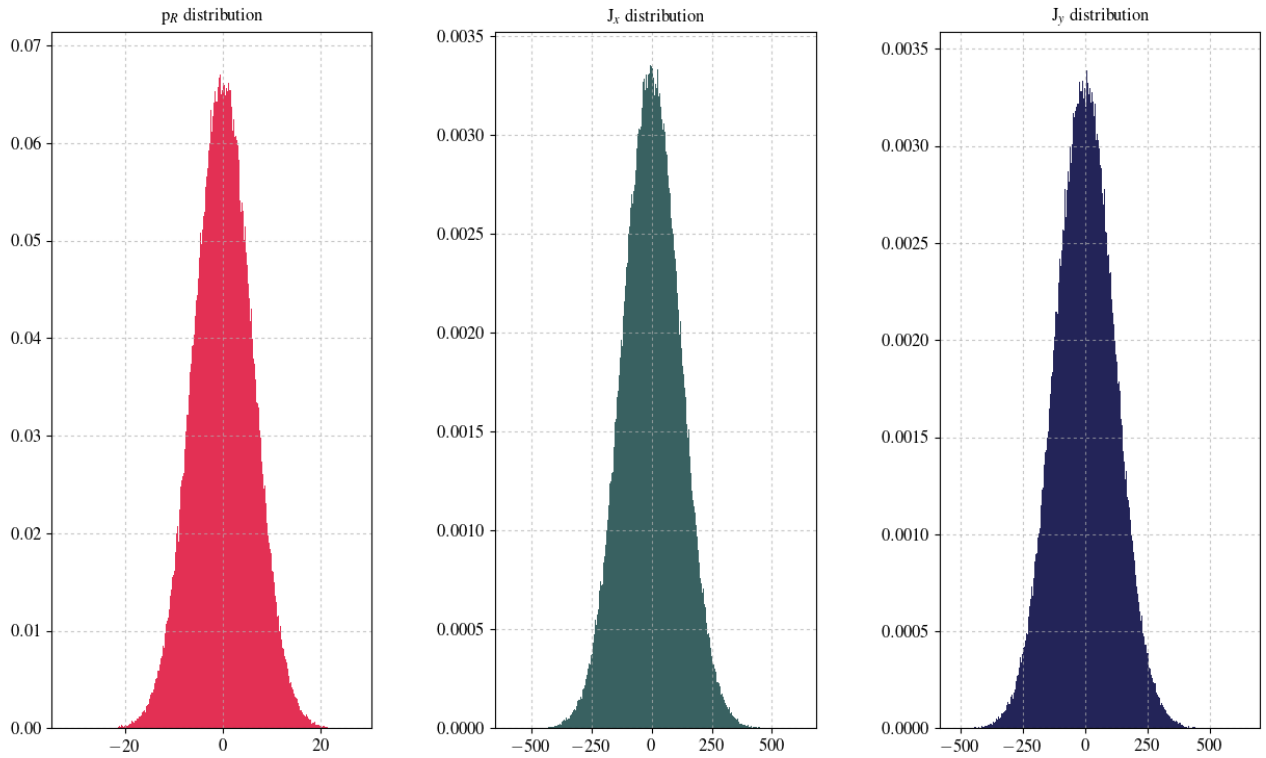

Рис. 1: Распределения переменных p_R, J_x, J_y для двух атомов с массами m_{Ar} и m_{CO_2} при $T = 300K$, 500.000 точек.

```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 $\bar{zp}$ и проходящей через центр координат O. Такое отражение описывается *Хаусхолдеровской матрицей*

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

где v – единичный вектор, параллельный $\bar{zp}$. 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.

```
1 #include <iostream>
2 #include <random>
3 #include <Eigen/Dense>
4
5 using namespace std;
6 using namespace Eigen;
7
8 //a Mersenne Twister pseudorandom generator of 32 bit numbers with a state
    size of 19937 bits
```

Algorithm 1 Scheme of [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: $\mathrm{v} = \begin{bmatrix} \cos\phi\sqrt{z} \\ \sin\phi\sqrt{z} \\ \sqrt{1-z} \end{bmatrix}$
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: $M \leftarrow \left(2vv^\top - 1\right) \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

```
9  random_device rd;
10 mt19937 eng( rd() );
11 uniform_real_distribution<double> distr( 0.0, 1.0);
12
13 void rotationMatrix( Matrix3d &m )
14 {
15   double theta = 2 * M_PI * distr( eng ); // a rotation about the pole
16   double phi = 2 * M_PI * distr( eng ); // a direction to deflect the pole
17   double z = distr( eng ); // the amount of pole deflection
18   double sz = sqrt( z );
19
20   // a vector to perform the reflection
21   Vector3d v ( cos(phi) * sz, sin(phi) * sz, sqrt(1 - z) );
22   // the Householder matrix
23   Matrix3d s = 2 * v * v.transpose() - Matrix<double, 3, 3>::Identity();
24
25   Matrix3d r;
26     r << cos(theta), sin(theta), 0,
27     -sin(theta), cos(theta), 0,
28     0, 0, 1;
29
30   m = s * r;
31 }
32
33 int main( int argc, char* argv[] )
34 {
35   int n = atoi( argv[1] );
36
37   // initial vector
38   Vector3d v ( 0.0, 0.0, 1.0 );
39   // resulting vector
40   Vector3d r;
41
42   for ( int i = 0; i < n; i++ )
43   {
44     // filling rotation matrix
45       Matrix3d m;
46       rotationMatrix( m );
47
48     // performing a random rotation of OZ-vector
```

```
49      r = m * v;
50
51      // displaying the components of resulting vector
52      cout << r(0) << " " << r(1) << " " << r(2) << endl;
53    }
54
55      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.

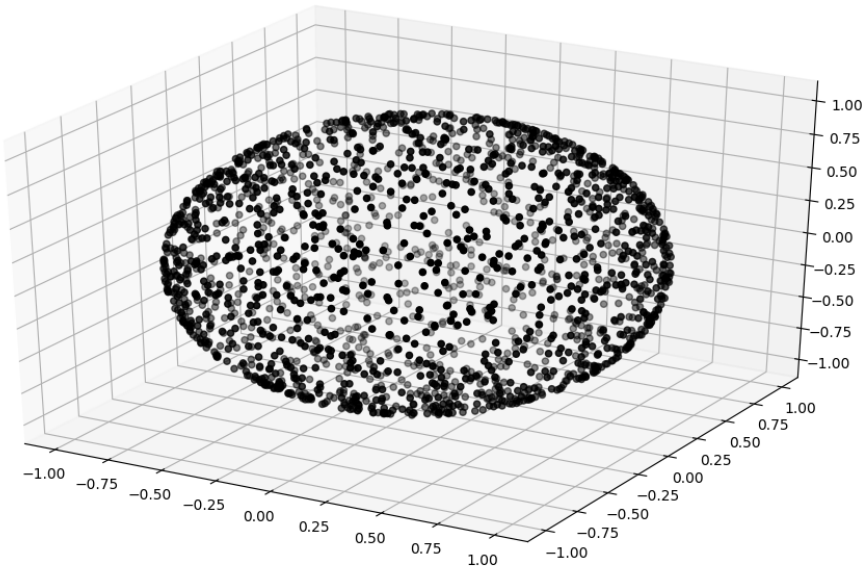

Рис. 2: Пример равномерно распределенных на сфере точек, полученных в результате приведенной выше программы. 2000 точек.

MCMC-sampling

Предположим мы генерируем последовательность случайных величин, $\{X_0, X_1, X_2, \dots\}$, такую что в каждый момент $t \geq 0$ следующее состояние X_{t+1} выбирается исходя из распределения $P(X_{t+1}|X_t)$, 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*.

Рассмотрим алгоритм Метрополиса-Гастингса, позволяющий получать последовательность

точек – элементов Марковской цепи – распределенную согласно заданной плотности вероятности $\pi(\cdot)$.

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 случайной точки (эта величина выбирается определенным образом на основе распределения; я же выбирал ее совершенно случайным образом, но так, чтобы она не оказалась в какой-то физически маловероятной области). Следующий за ним главный цикл алгоритма состоит из трех частей: (1) Получать 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 Гауссса:

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

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.

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.

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. D. Kirk. Graphics Gems III. III. 4. Fast random rotation matrices.

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 $\mathbf{r}$:

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

$$\dot{\mathbf{r}}_x \sim \int\limits_{-\infty}^{+\infty} f_1(x) f_2(z-x) dx = \frac{\sqrt{m_1 m_2}}{2\pi kT} \int\limits_{-\infty}^{+\infty} \exp\left(-\frac{m_1 x^2}{2kT}\right) \exp\left(-\frac{m_2 (z-x)^2}{2kT}\right) dx \tag{6}$$

Отдельно рассмотрим получившийся интеграл:

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

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

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

$$= \sqrt{\frac{2\pi kT}{m_1+m_2}} \exp\left(-\frac{m_1 m_2}{2(m_1+m_2)kT} z^2\right) \tag{7}$$

Подставляя значение интеграла (7) в выражение для плотности распределения $\dot{\mathbf{r}}_x$ (6), 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 $\mu = \dfrac{m_1 m_2}{m_1+m_2}$.