Санкт-Петербургский государственный политехнический университет

Институт Прикладной математики и механики
Кафедра Теоретической механики

А.В.Смирнов

МОЛЕКУЛА УГЛЕКИСЛОГО ГАЗА

Курсовой проект

Направление подготовки бакалавров: 010800 Механика и математическое моделирование

Группа 23604/1

Руководитель проекта: Панченко А.Ю.

Допущен к защите:

«\_\_» 20\_\_ г.

Санкт-Петербург

2015

#

Оглавление

[**Глава 1 Молекулярная динамика** 3](#_Toc421271937)

[1.2 Парные потенциалы 5](#_Toc421271938)

[1.2.1 Потенциал Морзе. 5](#_Toc421271939)

[1.2.2 Потенциал Леннард-Джонса. 6](#_Toc421271940)

[1.2.3 Сравнение потенциалов Морзе и Леннард-Джонса 7](#_Toc421271941)

[1.2.4 Графики сравнения потенциалов и сил. 7](#_Toc421271942)

[1.2.5 Вывод 9](#_Toc421271943)

[1.2 Молекула углекислого газа 9](#_Toc421271944)

[**Глава 2 Написание программы** 10](#_Toc421271945)

[2.1 Требования к программе 10](#_Toc421271946)

[2.2 Код программы. 11](#_Toc421271947)

[2.2.1 Переменные. 11](#_Toc421271948)

[2.2.2 Функция создания частиц 12](#_Toc421271949)

[2.2.3 Функция физики 14](#_Toc421271950)

[2.2.4 Функция power 18](#_Toc421271951)

[2.3 Выбор оптимальных параметров 19](#_Toc421271952)

[**Итоги работы** 20](#_Toc421271953)

[**Список Литературы** 21](#_Toc421271954)

**Введение и формулировка задачи**

Моделирование молекул, даже самых простых - сложная задача. Для их моделирования необходимо использовать многочастичные потенциалы, но их программирование - тоже очень сложная задача. Встает вопрос о том, можно ли найти более простой путь моделирования простейших молекул.

Для моделирования хорошо подходят парные потенциалы, ибо они имеют простой вид и легко программируются. Но как их применить к моделированию молекул? Моя работа и посвящена решению данной проблемы.

Поэтому, задачу, поставленную перед мои проектом можно сформулировать так - смоделировать с помощью парного потенциала молекулу углекислого газа ( 2D модель ) и рассмотреть ее простейшую динамику молекулы.

# **Глава 1 Молекулярная динамика**

* 1. **Метод классической молекулярной динамики**

Метод молекулярной динамики (метод МД) — метод, в котором временная эволюция системы взаимодействующих атомов или частиц отслеживается интегрированием их уравнений движения

Основные положения:

* Для описания движения атомов или частиц применяется [классическая механика](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%BA%D0%B0). Закон движения частиц находят при помощи аналитической механики.
* Силы межатомного взаимодействия можно представить в форме классических потенциальных сил (как градиент потенциальной энергии системы).
* Точное знание траекторий движения частиц системы на больших промежутках времени не является необходимым для получения результатов макроскопического (термодинамического) характера.
* Наборы конфигураций, получаемые в ходе расчетов методом молекулярной динамики, распределены в соответствии с некоторой статистической функцией распределения, например отвечающей микроканоническому распределению.

Метод молекулярной динамики применим, если длина волны Де Бройля атома (или частицы) много меньше, чем межатомное расстояние.

Также классическая молекулярная динамика не применима для моделирования систем, состоящих из легких атомов, таких как гелий или водород. Кроме того, при низких температурах квантовые эффекты становятся определяющими и для рассмотрения таких систем необходимо использовать квантов- химические методы. Необходимо, чтобы времена на которых рассматривается поведение системы были больше, чем время релаксации исследуемых физических величин.

Метод молекулярной динамики, изначально разработанный в теоретической физике, получил большое распространение в химии и, начиная с 1970х годов, в биохимии и биофизике. Он играет важную роль в определении структуры белка и уточнении его свойств, если взаимодействие между объектами может быть описано силовым полем.

## 1.2 Парные потенциалы

В своей работе я использовал два потенциала: Леннард-Джонса и Морзе. О них и пойдет речь ниже.

### 1.2.1 [Потенциал Морзе.](http://tm.spbstu.ru/%D0%9F%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB_%D0%9C%D0%BE%D1%80%D0%B7%D0%B5)

Парный силовой потенциал взаимодействия. Определяется формулой:



где

* *D* — энергия связи,
* *a* — длина связи,
* *α* — параметр, характеризующий ширину потенциальной ямы.

Потенциал имеет один безразмерный параметр *αa*. При *αa*=6 взаимодействия Морзе и Леннард-Джонса близки. При увеличении *αa* ширина потенциальной ямы для взаимодействия Морзе уменьшается, взаимодействие становится более жестким и хрупким.

Уменьшение *αa* приводит к противоположным изменениям — потенциальная яма расширяется, жесткость падает.

 Сила, соответствующая потенциалу Морзе, вычисляется по формуле:



Или в векторной форме:



### 1.2.2 [Потенциал Леннард-Джонса](http://tm.spbstu.ru/%D0%9F%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB_%D0%9B%D0%B5%D0%BD%D0%BD%D0%B0%D1%80%D0%B4-%D0%94%D0%B6%D0%BE%D0%BD%D1%81%D0%B0).

Парный силовой потенциал взаимодействия. Определяется формулой:



где

* *r* — расстояние между частицами,
* *D* — энергия связи,
* *a* — длина связи.

Потенциал является частным случаем [потенциала Ми](http://tm.spbstu.ru/%D0%9F%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB_%D0%9C%D0%B8) и не имеет безразмерных параметров.

Сила взаимодействия, соответствующая потенциалу Леннард-Джонса, вычисляется по формуле



Для потенциала Леннард-Джонса жесткость связи, критическая длина связи и прочность связи, соответственно, равны


Векторная сила взаимодействия определяется формулой



Данное выражение содержит лишь четные степени межатомного расстояния *r*, что позволяет при численных расчетах методом динамики частиц не использовать операцию извлечения корня.

### 1.2.3 Сравнение потенциалов Морзе и Леннард-Джонса

Чтобы определиться с потенциалом, рассмотрим каждый с функциональной точки зрения.

У обоих потенциалов есть два слагаемых, одно отвечает за притяжение, а другое за притяжение.

В потенциале Морзе содержится экспонента с отрицательным показателем – одна из самых быстро убывающих функций. Напомню, что показатель имеет вид $-2α(r-a)$ для слагаемого, отвечающего за отталкивание, и $-α(r-a)$ для слагаемого, отвечающего за притяжение.

Преимущества:

* имеет параметр $α$, с помощью которого можно подбирать ширину потенциальной ямы
* при больших $r$ сила взаимодействия становится очень мала( во много раз меньше чем у потенциала Леннард- Джонса )

Потенциал Леннард Джонса в свою очередь содержит степенную функцию вида

$$\left(\frac{a}{r}\right)^{n}$$

$$ $$

Где n = 6 для слагаемого, отвечающего за притяжение, и n = 12 для слагаемого, отвечающего за отталкивания.

Преимущества:

* не требуется операция извлечения квадратного корня, так как при программировании степени четные
* Более плавное убывание и возрастание по сравнению с потенциалом Морзе

###  1.2.4 Графики сравнения потенциалов и сил.

* *αa*= 6
* *αa* = 4



* *αa*= 8



### 1.2.5 Вывод

Из данных графиков можно сделать 1 вывод – потенциал Морзе более гибкий, поэтому больше подходит для моих нужд, ибо надо описывать взаимодействия между тремя частицами, и для этого потребуется 3 вида потенциала:

1. Для взаимодействия между кислородом и углеродом ( оно одинаковое для каждого кислорода в молекуле )
2. Для взаимодействия между кислородами в молекуле углекислого газа (назовем его стабилизирующим )
3. Для взаимодействия между частицами из разным молекул

Поэтому в дальнейшем я буду использовать только потенциал Морзе, а название буду опускать.

## 1.2 Молекула углекислого газа

Углекислый газ ( диоксид углерода ) - газ без запаха и цвета. Молекула углекислого газа имеет линейное строение и ковалентные полярные связи, хотя сама молекула не является полярной. Дипольный момент = 0.

# **Глава 2 Написание программы**

## 2.1 Требования к программе

Программа должна выполнять следующие функции:

* Визуализация поля и стенок.
* Визуализация разных частиц.
* Система должна приходить в покой со временем.
* Правдоподобно описывать взаимодействие внутри молекулы.
* Молекула углекислого газа должна быть стабильной.
* Между разными частицами должно быть разное взаимодействие.
* Добавление и создание частиц по клавише мыши.
* Возможность управление частицами.
* Программа написана на языке Java Script, а визуализация происходит на canvas.
* Разные частицы рисуются разным цветом. Атом углерода рисуется коралловым, атом кислорода – синим и атом азота – золотым.
* В покой система приходит благодаря введению трению между стенками, частицами и средой.
* Описание функции physics будет дано ниже.
* Молекула становится после стабильной после введения потенциала между частицами кислорода.
* Взаимодействие меняется из-за изменения длины связи в формуле потенциала.
* Описание функций создания и удаления частиц ниже.
* Управление осуществляется зажатием левой кнопки мыши по частице.

## 2.2 Код программы.

Здесь будет кратко описываться исходный код основой части программы.

### 2.2.1 Переменные.

var Pi = 3.1415926; // число "пи"

 var m0 = 1; // масштаб массы

 var T0 = 1; // масштаб времени (период колебаний исходной системы)

 var a0 = 1; // масштаб расстояния (диаметр шара)

 var g0 = a0 / T0 / T0; // масштаб ускорения (ускорение, при котором за T0 будет пройдено расстояние a0)

 var k0 = 2 \* Pi / T0; // масштаб частоты

 var C0 = m0 \* k0 \* k0; // масштаб жесткости

 var B0 = 2 \* m0 \* k0; // масштаб вязкости

 // \*\*\* Задание физических параметров \*\*\*

 var Ny = 5; // число шаров, помещающихся по вертикали в окно (задает размер шара относительно размера окна)

 var m = 1 \* m0; // масса

 var Cwall = 10 \* C0; // жесткость стен

 var B = 0.003 \* B0; // вязкость среды

 var Bwall = 0.03 \* B0; // вязкость на стенках

 var Cball = 0.1 \* Cwall; // жесткость между частицами

 var mg = 0//0.25 \* m \* g0; // сила тяжести

 var r = 0.1 \* a0; // радиус частицы в расчетных координатах

 var K = 0.85; // сила взаимодействия ограничивается значением, реализующимся при r/a = K

 var a = 5 \* r; // равновесное расстояние между частицами

 var a\_1 = 2.1 \* r;

 var a\_2 = 2.1 \* a ;

 var aCut = 2 \* r ; // радиус обрезания

 var alfa = 5 ;

 // \*\*\* Задание вычислительных параметров \*\*\*

 var fps = 50; // frames per second - число кадров в секунду (качечтво отображения)

 var spf = 100; // steps per frame - число шагов интегрирования между кадрами (скорость расчета)

 var dt = 0.045 \* T0 / fps; // шаг интегрирования (качество расчета)

 // Выполнение программы

 var scale = canvas.height / Ny / a0 ; // масштабный коэффициент для перехода от расчетных к экранным координатам

 var r2 = r \* r ; // \_\_\_в целях оптимизации\_\_\_

 var aCut2 = aCut \* aCut ; // \_\_\_в целях оптимизации\_\_\_

 var a2 = a \* a ; // \_\_\_в целях оптимизации\_\_\_

 var a22 = a\_2 \* a\_2 ;

 var a11 = a\_1 \* a\_1 ;

 var D = a2 \* Cball / 72 ; // энергия связи между частицами

 var LJCoeff = 12 \* D / a2 ; // коэффициент для расчета потенциала Л-Дж

 var a1 = alfa / a ; // параметр для потенциала Морзе

 var MorzCoeff = 2 \* a1 \* D ; // коэффициент для расчета потенциала Морзе

 var Ka = K \* r ; // \_\_\_в целях оптимизации\_\_\_

 var K2a2 = Ka\*Ka ; // \_\_\_в целях оптимизации\_\_\_

 var w = canvas.width / scale ; // ширина окна в расчетных координатах

 var h = canvas.height / scale ; // высота окна в расчетных координатах

### 2.2.2 Функция создания частиц

var balls = []; // массив шаров

 var C =[] ; //массив углеродов

 var O1 = [] ; // массивы кислородов

 var O2 = [] ;

 var addNewBall = function(x, y) {

 // проверка - не пересекается ли новый шар со стенами или уже существующими шарами

 if (x - r < 0 || x + r > w || y - r < 0 || y + r > h) return null;

 for (var i = 0; i < balls.length; i++) {

 var rx = balls[i].x - x;

 var ry = balls[i].y - y;

 var rLen2 = rx \* rx + ry \* ry;

 if (rLen2 < 4 \* r2) return null;

 }

 var b = [];

 b.x = x; b.y = y; // расчетные координаты шара

 b.fx = 0; b.fy = mg; // сила, действующая на шар

 b.vx = 0; b.vy = 0; // скорость

 balls[balls.length] = b; // добавить элемент в конец массива

 return b;

 };

 var addNewC = function(x, y) {

 // проверка - не пересекается ли новый шар со стенами или уже существующими шарами

 if (x - r < 0 || x + r > w || y - r < 0 || y + r > h) return null;

 for (var i = 0; i < balls.length; i++) {

 var rx = balls[i].x - x;

 var ry = balls[i].y - y;

 var rLen2 = rx \* rx + ry \* ry;

 if (rLen2 < 4 \* r2) return null;

 }

 var b = [];

 b.x = x; b.y = y; // расчетные координаты шара

 b.fx = 0; b.fy = mg; // сила, действующая на шар

 b.vx = 0; b.vy = 0; // скорость

 C[C.length] = b; // добавить элемент в конец массива

 return b;

 };

 var addNewO1 = function(x, y) {

 // проверка - не пересекается ли новый шар со стенами или уже существующими шарами

 if (x - r < 0 || x + r > w || y - r < 0 || y + r > h) return null;

 for (var i = 0; i < balls.length; i++) {

 var rx = balls[i].x - x;

 var ry = balls[i].y - y;

 var rLen2 = rx \* rx + ry \* ry;

 if (rLen2 < 4 \* r2) return null;

 }

 var b = [];

 b.x = x; b.y = y; // расчетные координаты шара

 b.fx = 0; b.fy = mg; // сила, действующая на шар

 b.vx = 0; b.vy = 0; // скорость

 O1[O1.length] = b; // добавить элемент в конец массива

 return b;

 };

 var addNewO2 = function(x, y) {

 // проверка - не пересекается ли новый шар со стенами или уже существующими шарами

 if (x - r < 0 || x + r > w || y - r < 0 || y + r > h) return null;

 for (var i = 0; i < balls.length; i++) {

 var rx = balls[i].x - x;

 var ry = balls[i].y - y;

 var rLen2 = rx \* rx + ry \* ry;

 if (rLen2 < 4 \* r2) return null;

 }

 var b = [];

 b.x = x; b.y = y; // расчетные координаты шара

 b.fx = 0; b.fy = mg; // сила, действующая на шар

 b.vx = 0; b.vy = 0; // скорость

 O2[O2.length] = b; // добавить элемент в конец массива

 return b;

 };

### 2.2.3 Функция физики

function physics() { // то, что происходит каждый шаг времени

 for (var s = 1; s <= spf; s++) {

 // пересчет сил идет отдельным массивом, т.к. далее будут добавляться силы взаимодействия между шарами

 for (var i0 = 0; i0 < balls.length; i0++) {

 balls[i0].fx = - B \* balls[i0].vx;

 balls[i0].fy = mg - B \* balls[i0].vy;

 }

 for (var i0 = 0; i0 < C.length; i0++) {

 C[i0].fx = - B \* C[i0].vx;

 C[i0].fy = mg - B \* C[i0].vy;

 }

 for (var i0 = 0; i0 < O1.length; i0++) {

 O1[i0].fx = - B \* O1[i0].vx;

 O1[i0].fy = mg - B \* O1[i0].vy;

 }

 for (var i0 = 0; i0 < O2.length; i0++) {

 O2[i0].fx = - B \* O2[i0].vx;

 O2[i0].fy = mg - B \* O2[i0].vy;

 }

 for (var i = 0; i < balls.length; i++) { // расчет для массива Balls

 // расчет взаимодействия производится со всеми следующими шарами в массиве,

 // чтобы не считать каждое взаимодействие дважды

 var b = balls[i];

 for (var j = i + 1; j < balls.length; j++) {

 var b2 = balls[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( r2 > aCut2 ) continue ;

 powers(b,b2,a\_1,0.3) ; // описание этой функции ниже

 }

 for (var j = 0; j < C.length; j++) {

 var b2 = C[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( r2 > aCut2 ) continue ;

 powers(b,b2,a\_1,0.3) ;

 }

 for (var j = 0; j < O1.length; j++) {

 var b2 = O1[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( r2 > aCut2 ) continue ;

 powers(b,b2,a\_1,0.3) ;

 }

 for (var j = 0; j < O2.length; j++) {

 var b2 = O2[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( r2 > aCut2 ) continue ;

 powers( b, b2, a\_1 ,0.3) ;

 }

 if (b == dNd) continue; // если шар схвачен курсором - его вз. со стенами и перемещение не считаем

 if (b.y + r > h) { b.fy += -Cwall \* (b.y + r - h) - Bwall \* b.vy; }

 if (b.y - r < 0) { b.fy += -Cwall \* (b.y - r) - Bwall \* b.vy;}

 if (b.x + r > w) { b.fx += -Cwall \* (b.x + r - w) - Bwall \* b.vx; }

 if (b.x - r < 0) { b.fx += -Cwall \* (b.x - r) - Bwall \* b.vx; }

 b.vx += b.fx / m \* dt; b.vy += b.fy / m \* dt;

 b.x += b.vx \* dt; b.y += b.vy \* dt;

 }

 for (var i = 0; i < C.length; i++) { // расчет для массива C

 // расчет взаимодействия производится со всеми следующими шарами в массиве,

 // чтобы не считать каждое взаимодействие дважды

 var b = C[i];

 for (var j = i + 1; j < C.length; j++) {

 var b2 = C[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( r2 > aCut2 ) continue ;

 powers(b,b2,a\_1, 0.3) ;

 }

 for (var j = 0; j < O1.length; j++) {

 var b2 = O1[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( j == i ) { powers(b,b2,a, 6) ;}

 else {

 if ( r2 > aCut2 ) continue ;

 powers(b,b2,a\_1, 0.3) ;

 }

 }

 for (var j = 0; j < O2.length; j++) {

 var b2 = O2[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( j == i ) { powers(b,b2,a,6) ;}

 else {

 if ( r2 > aCut2 ) continue ;

 powers(b,b2,a\_1, 0.3) ;

 }

 }

 if (b == dNd) continue; // если шар схвачен курсором - его вз. со стенами и перемещение не считаем

 if (b.y + r > h) { b.fy += -Cwall \* (b.y + r - h) - Bwall \* b.vy; }

 if (b.y - r < 0) { b.fy += -Cwall \* (b.y - r) - Bwall \* b.vy;}

 if (b.x + r > w) { b.fx += -Cwall \* (b.x + r - w) - Bwall \* b.vx; }

 if (b.x - r < 0) { b.fx += -Cwall \* (b.x - r) - Bwall \* b.vx; }

 b.vx += b.fx / m \* dt; b.vy += b.fy / m \* dt;

 b.x += b.vx \* dt; b.y += b.vy \* dt;

 }

 for (var i = 0; i < C.length; i++) { // расчет для массива O1

 // расчет взаимодействия производится со всеми следующими шарами в массиве,

 // чтобы не считать каждое взаимодействие дважды

 var b = O1[i];

 for (var j = i + 1; j < O1.length; j++) {

 var b2 = O1[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( r2 > aCut2 ) continue ;

 powers(b,b2,a\_1,0.3) ;

 }

 for (var j = 0; j < O2.length; j++) {

 var b2 = O2[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( i == j ) { powers(b,b2,a\_2, 3) ;}

 else {

 if ( r2 > aCut2 ) continue ;

 else powers(b,b2,a\_1 , 0.3) ;

 }

 }

 if (b == dNd) continue; // если шар схвачен курсором - его вз. со стенами и перемещение не считаем

 if (b.y + r > h) { b.fy += -Cwall \* (b.y + r - h) - Bwall \* b.vy; }

 if (b.y - r < 0) { b.fy += -Cwall \* (b.y - r) - Bwall \* b.vy;}

 if (b.x + r > w) { b.fx += -Cwall \* (b.x + r - w) - Bwall \* b.vx; }

 if (b.x - r < 0) { b.fx += -Cwall \* (b.x - r) - Bwall \* b.vx; }

 b.vx += b.fx / m \* dt; b.vy += b.fy / m \* dt;

 b.x += b.vx \* dt; b.y += b.vy \* dt;

 }

 for (var i = 0; i < C.length; i++) { // расчет для массива O2

 // расчет взаимодействия производится со всеми следующими шарами в массиве,

 // чтобы не считать каждое взаимодействие дважды

 var b = O2[i];

 for (var j = i + 1; j < O2.length; j++) {

 var b2 = O2[j];

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 if ( r2 > aCut2) continue ;

 powers(b,b2,a\_1, 0.3) ;

 }

 if (b == dNd) continue; // если шар схвачен курсором - его вз. со стенами и перемещение не считаем

 if (b.y + r > h) { b.fy += -Cwall \* (b.y + r - h) - Bwall \* b.vy; }

 if (b.y - r < 0) { b.fy += -Cwall \* (b.y - r) - Bwall \* b.vy;}

 if (b.x + r > w) { b.fx += -Cwall \* (b.x + r - w) - Bwall \* b.vx; }

 if (b.x - r < 0) { b.fx += -Cwall \* (b.x - r) - Bwall \* b.vx; }

 b.vx += b.fx / m \* dt; b.vy += b.fy / m \* dt;

 b.x += b.vx \* dt; b.y += b.vy \* dt;

 }

 }

 }

### 2.2.4 Функция power

function powers(b ,b2 , hkk , jk ) {

 var rx = b.x - b2.x; var ry = b.y - b2.y; // вектор смотрит на первый шар (b)

 var r2 = rx \* rx + ry \* ry; // квадрат расстояния между шарами

 var rLen = (Math.sqrt(r2));

 if (r2 < K2a2) {

 if (rLen > 0.00001) { // проверка, чтобы избежать деления на 0

 rx = rx / rLen \* Ka;

 ry = ry / rLen \* Ka;

 }

 r2 = K2a2;

 rLen = Ka; // корень K2a2

 }

 // сила взаимодействия

 var u = Math.exp( -a1 \* ( rLen - hkk )) ;

 var F = jk \* MorzCoeff \* u\* ( u - 1 ) /rLen ;

 var Fx = F \* rx; var Fy = F \* ry; // подсчет сил для каждой координаты

 b.fx += Fx; b.fy += Fy;

 b2.fx -= Fx; b2.fy -= Fy;

 }

## 2.3 Выбор оптимальных параметров

При моделировании молекулы углекислого газа перед нами встает три вопроса:

* Какую взять длину связи для каждого потенциала.
* Какое взять соотношение между коэффициентами в потенциале.
* Определить оптимальную величину параметра $αa$.

Длину связи между кислородом и углеродом возьмем константную и равную $a$.

Тогда длина связи между кислородами в молекуле углекислого газа может быть больше $2a$, меньше $2a$, либо равна $2a$.

Если взять длину связи меньше $2a$, то молекула будет стремиться к треугольной. Если взять длину связи равную $2a$, то молекула получается неустойчивой и легко разрушается. Самым оптимальным вариантом является случай, когда длина связи немного больше $2a$. Из-за этого появляется напряжение в молекуле, и оно не дает ей легко разрушиться.

Опыты в программе показали, что оптимальным соотношением между коэффициентом для потенциала между кислородом и углеродом и коэффициентом потенциала между кислородами является соотношение равное 2.

Оптимальная ширина ямы (а именно на это влияет параметр) достигается при *αa* = 4.

# **Итоги работы**

Разработан алгоритм, с помощью которого можно смоделировать молекулу углекислого газа при помощи парных потенциалов. Данная модель имеет свои особенности:

* За конечное время при небольшой вязкости среды частицы собираются в молекулы углекислого газа.
* Поведение молекулы углекислого газа в системе при небольших скоростях получается реалистичным. Это означает, что можно изучать динамику системы при малых скоростях молекул.
* При больших скоростях молекулы углекислого газа разрушаются. После разрушения они собираются заново, образуя те же самые молекулы.

В будущем планируется ввести степени насыщенности связи и начать моделировать более сложные молекулы.

# **Список Литературы**

1. **Ландау Л.Д., Лифшиц Е.М.,** Квантовая Механика
2. [**Глинка Н.Л.**](http://www.alleng.ru/d/chem/chem26.htm)[**,**](http://www.alleng.ru/d/engl/engl67.htm)Общая химия
3. **Ландау Л.Д., Лифшиц Е.М.,** Механика
4. **Беринский Е.И., Кривцов А.М., Кузькин В.А.,** Современные проблемы механики. Механические свойства ковалентных кристалов