



Моделирование физического процесса "Абсолютно упругое взаимодействие"

Группа М3203
Кравченкова Елизавета

Преподаватель
Хуснутдинова Наира Рустемовна

Физические основы компьютерных и сетевых технологий
Университет ИТМО
Санкт-Петербург, Россия

7 января 2024 г.

Оглавление

1 Задание	3
1.1 Объект исследования	3
1.2 Метод экспериментального исследования	3
2 Цели и Задачи	4
3 Теория и рабочие формулы	5
3.1 Абсолютно упругий удар	5
3.2 Абсолютно упругий удар шара о неподвижную массивную стенку.	7
4 Ход работы	8
4.1 Поготовка	8
4.2 Основная часть	8
4.2.1 Задание констант	8
4.2.2 Создание модели	9
4.2.3 Реализация основных функций физической модели	10
4.2.4 Тело программы	12
4.3 Результат	13
5 Исследование	14
5.1 Рассмотрение требуемых случаев	14
5.1.1 $m_2 = m_1 \cdot 10$	14
5.1.2 $m_2 = m_1 \cdot 10^2$	15
5.1.3 $m_2 = m_1 \cdot 10^3$	15
5.1.4 $m_2 = m_1 \cdot 10^4$	16

5.1.5	$m_2 = m_1 \cdot 10^5$	16
5.1.6	$m_2 = m_1 \cdot 10^6$	17
5.2	Анализ	17
5.3	Проверка	24
5.3.1	$m_2 = m_1 \cdot 10$	24
5.3.2	$m_2 = m_1 \cdot 10^2$	24
5.3.3	$m_2 = m_1 \cdot 10^3$	24
5.3.4	$m_2 = m_1 \cdot 10^4$	25
5.3.5	$m_2 = m_1 \cdot 10^5$	25
5.3.6	$m_2 = m_1 \cdot 10^6$	25
6	Выводы	26

Задание

Тело массой m_2 , движущееся со скоростью v_0 , сталкивается с неподвижным телом массой m_1 . Масса $m_1 < m_2$. Сколько соударений N со стенкой, расположенной слева и телом массой m_2 , совершил тело массой m_1 , до полной остановки. Столкновения со стенкой и телами считать абсолютно упругими.

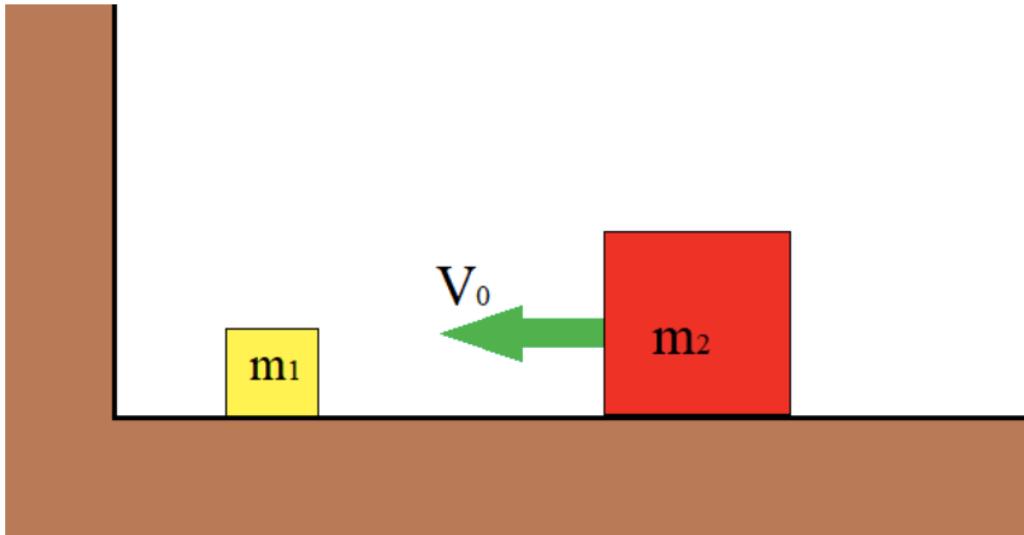


Рис. 1. Иллюстрация к задаче

Рассмотреть случаи когда $m_2 = m_1 \cdot 10^n$, где $n = 1, 2, 3, 4\dots$. Проанализировать полученные результаты.

1.1 Объект исследования

Исследование абсолютно упругого взаимодействия

1.2 Метод экспериментального исследования

Имитация физической модели с помощью языка программирования

Цели и Задачи

Цель: Выполнить численное моделирование абсолютно упругого взаимодействия двух тел разной массы.

Задачи:

1. Вывести необходимые для моделирования формулы скорости тел после их взаимодействия.
2. Имитация физической модели с помощью языка программирования c++.
3. Исследование полученных результатов.

Теория и рабочие формулы

В физике под ударом понимают такой тип взаимодействия движущихся тел, при котором временем соприкосновения можно пренебречь.

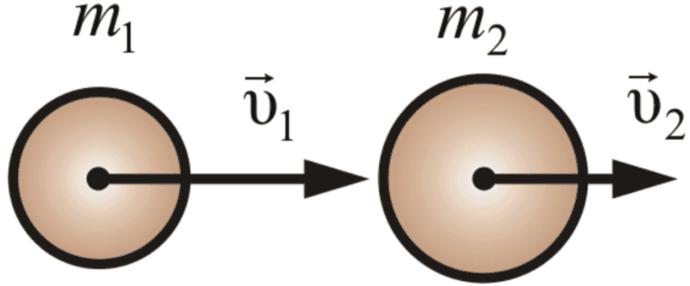


Рис. 2. Два шара, при $v_1 > v_2$ произойдет соударение

При ударе выполняется закон сохранения импульса и закон сохранения момента импульса, но обычно не выполняется закон сохранения механической энергии. Она переходит в нагрев, деформацию тел, колебания (в том числе акустические) и другие виды энергии. Но для удобства рассмотрения в физике применяются упрощённые модели. Поэтому используются предельные случаи:

1. абсолютно упругий удар, при котором полная кинетическая энергия тел сохраняется;
2. абсолютно неупругий, когда тела соединяются в единое целое, затрачивая энергию на неупругую деформацию.

Закон сохранения импульса(в проекциях на ось x)

$$m_1v_1 + m_2v_2 = m_1v'_1 + m_2v'_2 \quad (1)$$

3.1 Абсолютно упругий удар

Абсолютно упругий удар — это модель соударения, при которой полная кинетическая энергия системы сохраняется.

Закон сохранения механической энергии

$$\frac{m_1v_1^2}{2} + \frac{m_2v_2^2}{2} = \frac{m_1v'^2_1}{2} + \frac{m_2v'^2_2}{2} \quad (2)$$

В классической механике при этом пренебрегают деформациями тел. Соответственно, считается, что энергия на деформации не теряется, а взаимодействие распространяется по всему телу мгновенно.

Если взглянуть поближе, то при столкновении тел происходит их небольшая деформация. Её отчётливо можно заметить для теннисного мячика. Для бильярдного шара она очень мала, а в случае с заряженными частицами изменяет свою форму их электрическое поле. Эти случаи объединяет то, что деформация близка к упругой.

Поверхности сжимаются подобно пружинам, запасая энергию на доли секунды. Кинетическая энергия тел переходит в потенциальную энергию упругой деформации. Но потом поверхности снова выпрямляются, отталкивая тела. В результате энергия снова перетекает в кинетическую. Но эти переходы считаются моментальными. Однако, нельзя считать, что энергия сохранилась для каждого тела. Объекты взаимодействуют и совершают работу. Шар, налетев на другой, теряет скорость. Закон сохранения энергии выполняется лишь для системы, которая не получает приращения энергии от тел извне – закрытой системы.

Получаем, что математическая модель абсолютно упругого удара работает примерно следующим образом:

1. Есть в наличии два абсолютно твёрдых тела, которые сталкиваются.
2. В точке контакта происходят упругие деформации. Кинетическая энергия движущихся тел мгновенно и полностью переходит в энергию деформации.
3. В следующий момент деформированные тела принимают свою прежнюю форму, а энергия деформации полностью обратно переходит в кинетическую энергию.
4. Контакт тел прекращается, и они продолжают движение.

Для математического описания простейших абсолютно упругих ударов используется закон сохранения энергии (2) и закон сохранения импульса (1):

$$\begin{cases} m_1 v_1 + m_2 v_2 = m_1 v'_1 + m_2 v'_2 \\ \frac{m_1 v_1^2}{2} + \frac{m_2 v_2^2}{2} = \frac{m_1 v'_1^2}{2} + \frac{m_2 v'_2^2}{2} \end{cases}$$

Решив эту систему относительно v'_1 и v'_2 получим:

$$v'_1 = \frac{2m_2 v_2 + (m_1 - m_2)v_1}{m_1 + m_2} \quad (3)$$

$$v'_2 = \frac{2m_1 v_1 + (m_2 - m_1)v_2}{m_1 + m_2} \quad (4)$$

3.2 Абсолютно упругий удар шара о неподвижную массивную стенку.

Стенку можно рассматривать как неподвижный шар с $v_2 = 0$ и массой $m_2 \rightarrow \infty$

Найдем v'_1 из (3). Разделим числитель и знаменатель на m_2 , тогда $\frac{m_1}{m_2}$ стремится к нулю:

$$v'_1 = \frac{2m_2 v_2 + (m_1 - m_2)v_1}{m_1 + m_2} = \frac{2v_2 + (\frac{m_1}{m_2} - 1)v_1}{\frac{m_1}{m_2} + 1} = 2v_2 - v_1 = -v_1$$

Таким образом, шар изменит скорость на противоположную.

Ход работы

4.1 Поготовка

Для визуализации физической модели был выбран язык c++. Для работы с графическим интерфейсом использовалась библиотека SFML. Для ее установки пропишем в CMakeLists.txt

```
1 include(FetchContent)
2 FetchContent_Declare(SFML
3     GIT_REPOSITORY https://github.com/SFML/SFML.git
4     GIT_TAG 2.6.x)
5 FetchContent_MakeAvailable(SFML)
6
7 target_link_libraries({PROJECT NAME} PRIVATE sfml-graphics)
8 target_compile_features({PROJECT NAME} PRIVATE cxx_std_17)
9
10 if(WIN32)
11     add_custom_command(
12         TARGET {PROJECT NAME}
13         COMMENT "Copy OpenAL DLL"
14         PRE_BUILD COMMAND ${CMAKE_COMMAND} -E copy ${SFML_SOURCE_DIR}/extlibs/bin/
15             ${IF:$<EQUAL:$CMAKE_SIZEOF_VOID_P,8>,x64,x86}/openal32.dll ${TARGET_FILE_DIR
16             :{PROJECT NAME}}
17             VERBATIM)
18 endif()
19
20 install(TARGETS {PROJECT NAME})
```

4.2 Основная часть

4.2.1 Задание констант

Зададим константы для нашего приложения: размеры окна, ширина левой стены, высота нижней поверхности, основные цвета, а также ограничим наибольший размер кубика в 200 пикселей.

```
1 const int MaxSquareSize = 200;
2 const int LENGTH_SIZE = 1280;
3 const int WIGHT_SIZE = 720;
```

```

4 const int LeftWallCoord = 35;
5 const int DownWallCoord = 80;
6 const sf::Color BG = sf::Color(230, 230, 250); //Lavender
7 const sf::Color MediumSlateBlueColor = sf::Color(123, 104, 238); //MediumSlateBlue

```

4.2.2 Создание модели

Создадим класс Model, который будет состоять из всех элементов нашей системы: два тела и стены. Благодаря концепции ООП вся информация о ее состоянии будет закрыта от пользователя (Очень хочется привести тут аналогию с фразой "На тело не действуют внешние силы"). Наш класс полностью управляет всеми данными о наших телах: их массой, скоростью, цветом, количеством столкновений и тд. Вообщем наш класс обладает следующей структурой:

```

1 class Model : public sf::Drawable, public sf::Transformable {
2
3 private:
4     bool active = false;
5     int CollisionsCount = 0;
6     int WidthLine = WIGHT_SIZE - DownWallCoord;
7     sf::Font font;
8     sf::Color color1 = sf::Color(255, 20, 147); //Deep Pink
9     sf::Color color2 = sf::Color(123, 104, 238); //MediumSlateBlue
10    std::pair<float, float> _speed = std::pair<float, float>(-100, 0);
11    int _mass1 = 100;
12    int _mass2 = 1;
13    sf::RectangleShape shape1 = sf::RectangleShape(sf::Vector2f((float)
14        GetSquareSize(_mass1), (float)GetSquareSize(_mass1)));
15    sf::RectangleShape shape2 = sf::RectangleShape(sf::Vector2f((float)
16        GetSquareSize(_mass2), (float)GetSquareSize(_mass2)));
17    sf::RectangleShape wallLeft = sf::RectangleShape(sf::Vector2f(LeftWallCoord ,
18        WIGHT_SIZE));
19    sf::RectangleShape wallDown = sf::RectangleShape(sf::Vector2f(LENGTH_SIZE,
20        DownWallCoord));
21
22    std::pair<std::pair<float, float>, std::pair<float, float>> getSquareCoord()
23    const;
24    int GetSquareSize(int mass) const;
25    void Collision();
26
27 public:
28     Model();

```

```

23     virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const;
24     void Move();
25     int GetCollisions();
26     void setActive();
27     bool getActive();
28     void setSpeed(float v);
29     float GetSpeed1();
30     float GetSpeed2();
31     void setMass(int mass1, int mass2);
32     void setStartSpeed(float v);
33 };

```

Пара неочевидных пояснений:

Переменная **active** показывает запущена система или нет (начало вообще что-то происходит?).

Внутренний метод **GetSquareSize()** возвращает оптимальный размер тела в пикселях в зависимости от его массы.

Внутренний метод **getSquareCoord()** используется для начального расположения кубиков там где надо. Он расчитывает координаты на которых они должны стоять и возвращает их

Внутренний метод **Collision()** вызывается когда два кубика сталкиваются, этот метод пересчитывает скорости по полученным формулам

Публичный метод **Move()** двигает тела. При пересечении тел - пересчитывает скорости.

4.2.3 Реализация основных функций физической модели

Как только мы создали экземпляр нашей модельки и отрисовали ее, существует единственный открытый метод **Move()**, который может сдвинуть наши кубики:

```

1 void Model::Move()
2 {
3     if (shape2.getPosition().x > LENGTH_SIZE && _speed.second > _speed.first){
4         ++CollisionsCount;
5         Collision();
6     }
7     else if (shape2.getGlobalBounds().intersects(wallLeft.getGlobalBounds())){
8         ++CollisionsCount;
9         _speed.second = -_speed.second;
10        shape2.setPosition(std::max((float)LeftWallCoord, shape2.getPosition().x),
11                            shape2.getPosition().y);
11    }

```

```

12     else if (shape2.getGlobalBounds().intersects(shape1.getGlobalBounds()))
13     {
14         ++CollisionsCount;
15         Collision();
16     }
17     else if (shape1.getGlobalBounds().intersects(shape2.getGlobalBounds()))
18     {
19         ++CollisionsCount;
20         Collision();
21     }
22
23     shape2.setPosition(std::max(shape2.getPosition().x+_speed.second,(float)
24 LeftWallCoord),shape2.getPosition().y);
25     shape1.setPosition(std::max(shape1.getPosition().x+_speed.first,shape2.
26 getPosition().x),shape1.getPosition().y);
27
28     if (shape1.getPosition().x < (shape2.getPosition().x + (float)GetSquareSize(
29 _mass2))){
30         shape1.setPosition(shape2.getPosition().x + (float)GetSquareSize(_mass2),
31 shape1.getPosition().y);
32     }
33
34 }

```

Первый if обрабатывает ситуацию, когда оба кубика улетели за границы экрана, тогда если мы понимаем, что соударение произойдет (скорость одного больше скорости второго), то сразу увеличиваем количество соударений на один и вызываем метод пересчета скоростей

Второй if обрабатывает ситуацию, когда маленький кубик ударился о левую стену, тогда мы просто меняем его скорость на противоположную

Третий и четвертый if обрабатывают ситуацию когда два кубика столкнулись. Увеличиваем количество соударений на один и вызываем метод пересчета скоростей

В конце мы перемещаем кубики на нужную скорость. Учитываем ширину стены, размеры кубиков, чтобы все изобразилось корректно

Далее рассмотрим метод **Collision()**, пересчитывающий скорости:

```

1 void Collision(){
2     float speed1 = (2*(float)_mass2*_speed.second + (float)(_mass1-_mass2)*
3     _speed.first)/(float)(_mass1+_mass2);

```

```

3         float speed2 = (2*(float)_mass1*_speed.first + (float)(_mass2-_mass1)*
4             _speed.second)/(float)(_mass1+_mass2);
5         _speed = std::make_pair(speed1,speed2);

```

В целом тут все просто. Просто по формулам (3) и (4) пересчитываем скорости, предварительно приводим массу к нецелому числу для увеличения точности.

В целом это вся физика в этой задаче.

4.2.4 Тело программы

Рассмотрим как вообще происходит отрисовка программы

```

1 int main(){
2     ...
3     while (window.isOpen())
4     {
5         for (auto event = sf::Event{}; window.pollEvent(event););
6         {
7             if (event.type == sf::Event::Closed)
8             {
9                 window.close();
10            }
11             if (event.type == sf::Event::KeyPressed)
12             {
13                 ...
14             }
15             // Check if event was mouse event
16             if (event.type == sf::Event::MouseButtonPressed )
17             {
18                 const bool pressed = button1.tryActivate();
19             }
20
21             for (auto & TestField : TestFields){
22                 TestField.input(event);
23                 TestField.render(window);
24             }
25         }
26         if (model.getActive()){

```

```

27     model.Move();
28 }
29 Marks[CollisionsNumber].setString(std::to_string(model.GetCollisions()));
30 Marks[speed1].setString(std::to_string(abs(model.GetSpeed1())));
31 Marks[speed2].setString(std::to_string(abs(model.GetSpeed2())));
32
33 window.clear(BG);
34 window.draw(model);
35 window.draw(button1);
36
37 for (const auto & Mark : Marks){
38     window.draw(Mark);
39 }
40 for (auto & TestField : TestFields){
41     TestField.render(window);
42 }
43 window.display();
44 }
45
46 }

```

Пока программа открыта - работаем. Если Мы нажмем на крестик программы, то while закончится, это обрабатывает первый if внутри while.

Второй if внутри while обрабатывает считывание данных в текстовые поля

Третий if проверяет нажата ли кнопка старт, если да - активируем модельку.

В строках 26-28 если моделька активирована двигаем ее

Далее меняем данные в текстовых полях, перерисовываем кнопки и тд.

4.3 Результат

С полным кодом вы можете ознакомиться и скачать его тут

А пример работы тут

Исследование

5.1 Рассмотрение требуемых случаев

5.1.1 $m_2 = m_1 \cdot 10$

Количество соударений: 10



$$5.1.2 \quad m_2 = m_1 \cdot 10^2$$

Количество соударений: 31



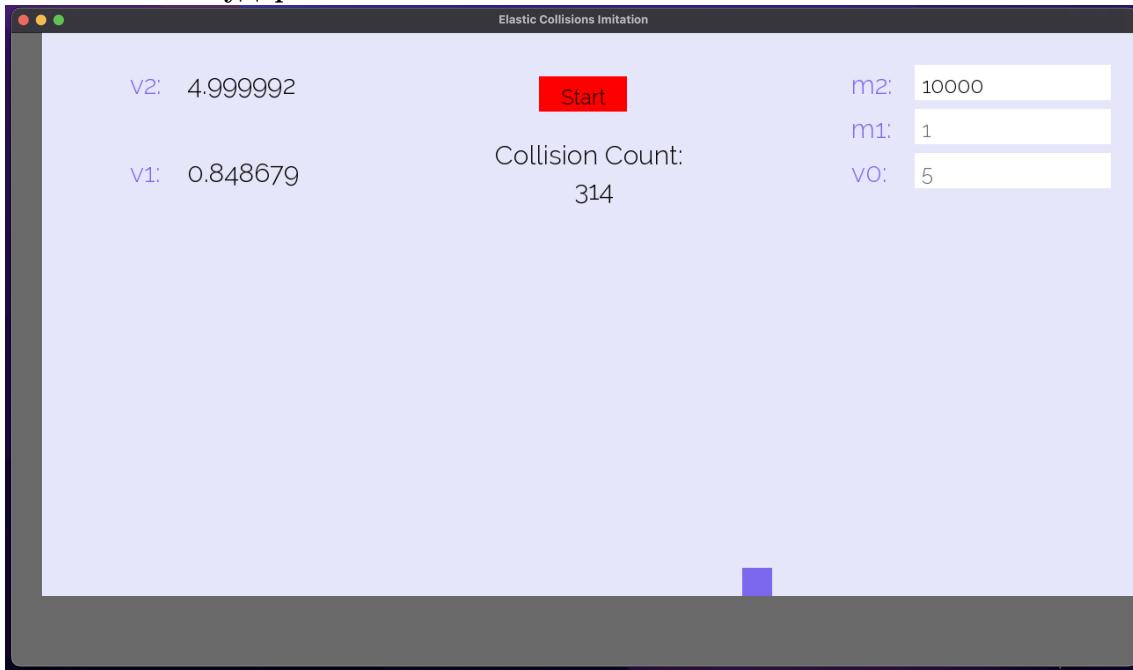
$$5.1.3 \quad m_2 = m_1 \cdot 10^3$$

Количество соударений: 99



5.1.4 $m_2 = m_1 \cdot 10^4$

Количество соударений: 314



5.1.5 $m_2 = m_1 \cdot 10^5$

Количество соударений: 993



5.1.6 $m_2 = m_1 \cdot 10^6$

Количество соударений: 3141



5.2 Анализ

Интересно, что при $m_2 = m_1 \cdot 10^{2k}$ число соударений очень похоже на число Пи. Также интересно, что число соударений не зависит от начальной скорости.

Рассмотрим закон сохранения энергии (2):

$$\frac{m_1 v_1^2}{2} + \frac{m_2 v_2^2}{2} = E$$

Это очень напоминает уравнение окружности, там тоже сумма квадратов является константой (корнем из радиуса)

Так и есть, это окружность радиуса $\sqrt{2E}$ в координатах $x = \sqrt{m_2}v_2$, $y = \sqrt{m_1}v_1$

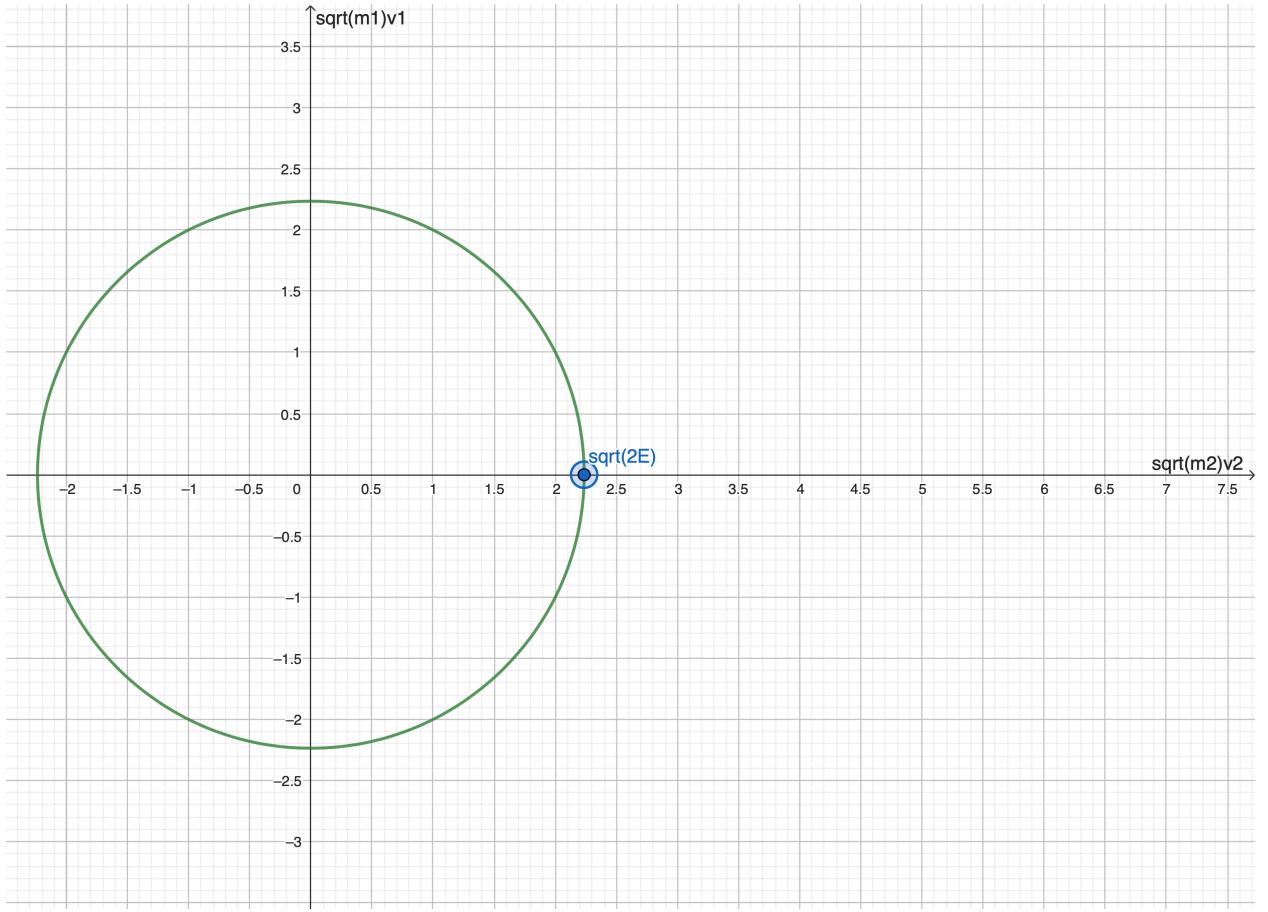


Рис. 5.1: Закон сохранения энергии в новых координатах

Выберем любую точку на графике. Это положение в данное время. Из закона сохранения импульса (3) выразим правило, по которому получим график всех точек с равным импульсом

$$m_1 v_1 + m_2 v_2 = P$$

$$\begin{aligned} \sqrt{m_1} v_1 &= -\frac{m_2 v_2}{\sqrt{m_1}} + \frac{P}{\sqrt{m_1}} \\ \sqrt{m_1} v_1 &= -\frac{\sqrt{m_2}}{\sqrt{m_1}} x + \frac{P}{\sqrt{m_1}} \\ y &= -\frac{\sqrt{m_2}}{\sqrt{m_1}} x + \frac{P}{\sqrt{m_1}} \end{aligned} \tag{6}$$

Тангенс угла наклона прямой $= -\frac{\sqrt{m_2}}{\sqrt{m_1}}$

Для примера взята точка начала Start ($v_1 = 0, v_2 < 0$). Провели прямую (6). На ней показаны все возможные точки(состояния), которые возможны после соударения с точки зрения закона сохранения импульса. Но нас интересует ситуация, когда еще и закон сохранения энергии сохраняется. А это только в точке пересечения В.

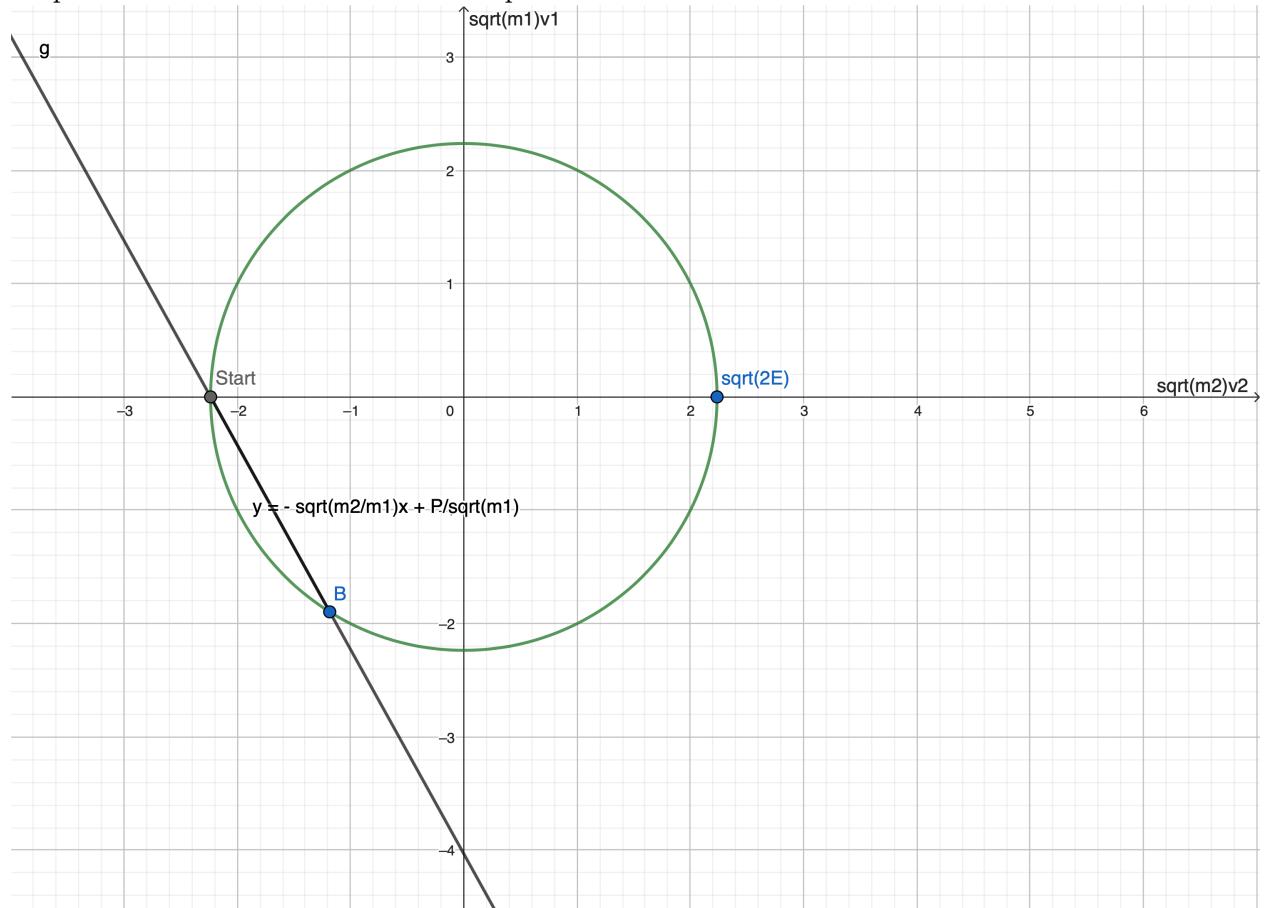


Рис. 5.2: Закон сохранения энергии и закон сохранения импульса в новых координатах

Что происходит дальше?

Дальше тело 1 ударяется о стенку и меняет свою скорость на противоположную. Происходит просто отражение относительно Ox :

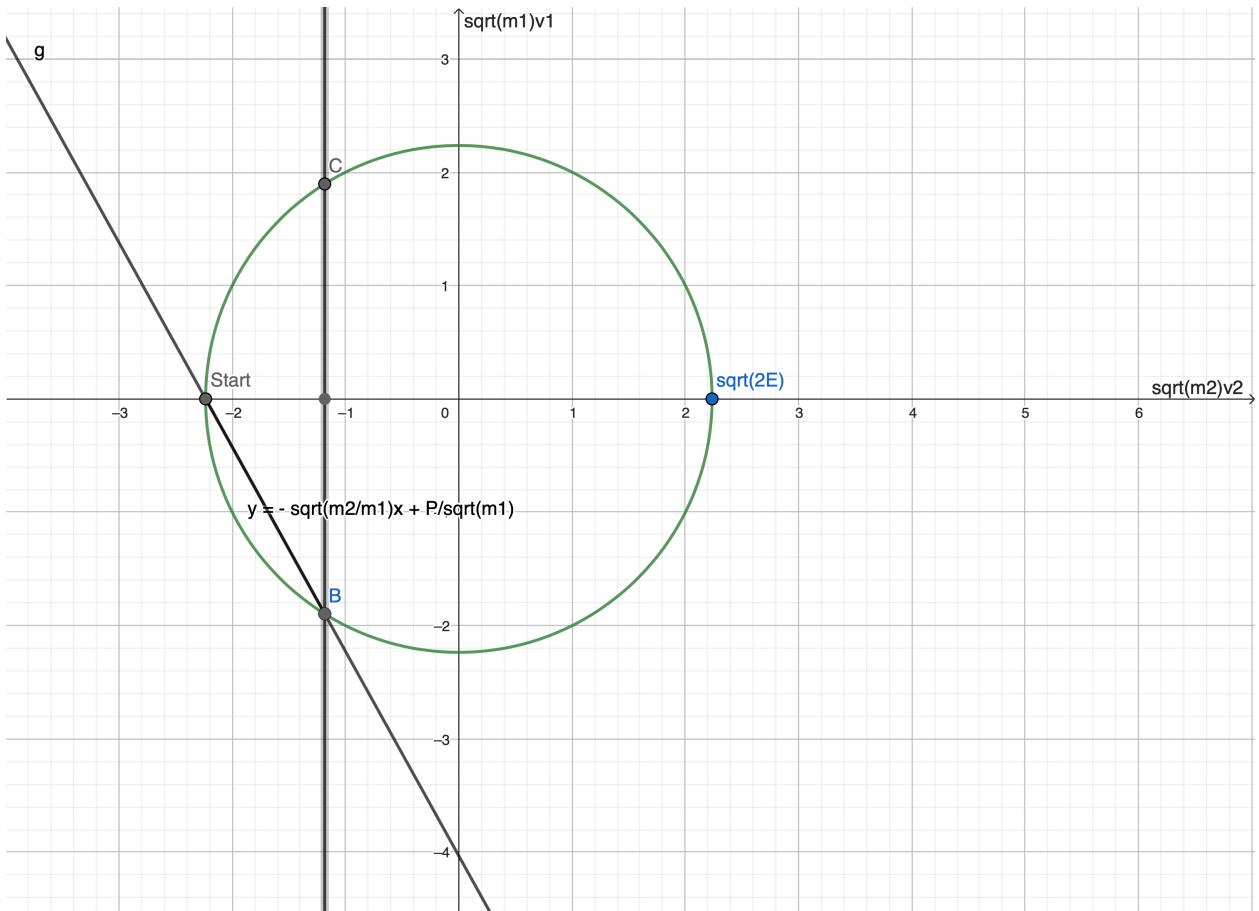


Рис. 5.3: Траектория тел в новых координатах после удара от стены маленького тела

Далее тела снова столкнутся друг с другом как вначале. По тому же правилу (6) изменятся скорости. И так далее.

До какого момента это будет происходить?

Ответ прост, все соударения закончатся в тот момент, когда $v_1 < v_2$ и $v_1 \geq 0$

Покажем это на графике:

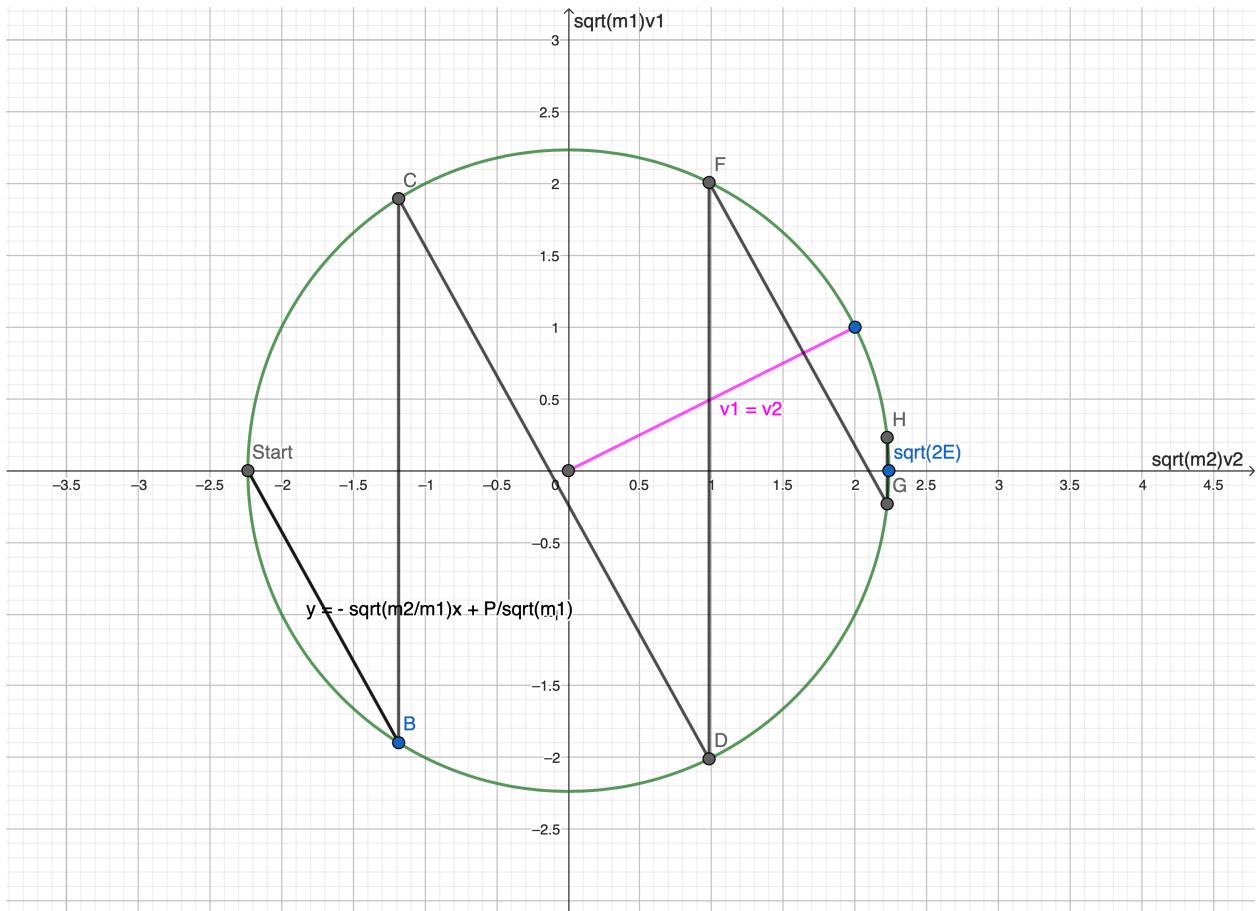


Рис. 5.4: Траектория тел в новых координатах от старта до конца соударений

Каждое новое соударение "отрезает" дугу окружности. Причем стоит отметить что эти дуги равны. Так как дуги окружности, заключенные между параллельными хордами равны.

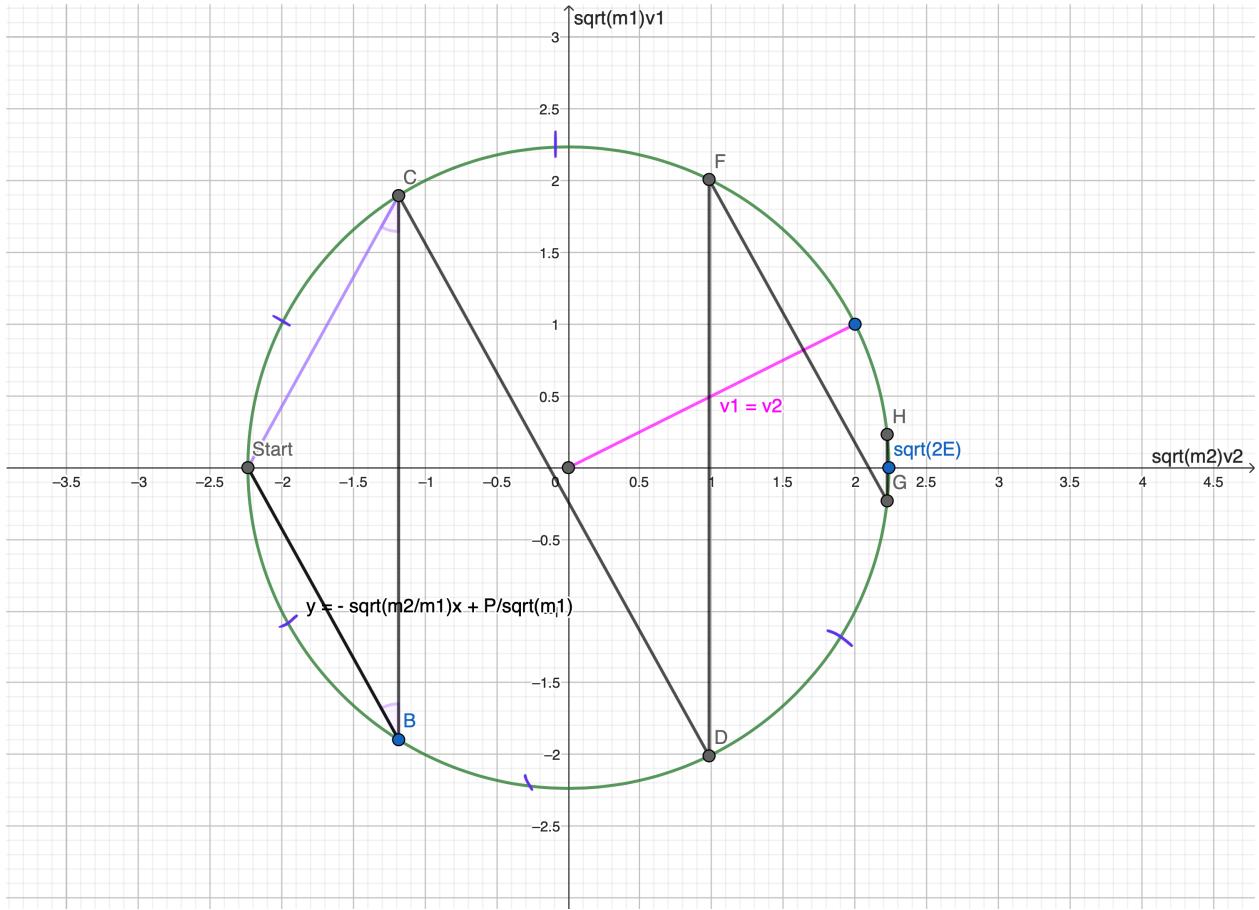


Рис. 5.5: Равенство дуг

Пусть каждая такая дуга равна 2α . И когда мы достигаем конца, остается небольшая часть окружности (обозначим 2γ), которую мы не прошли, но она точно меньше, чем 2α . Почему?
Докажем, что $2\gamma < 2\delta \leq 2\alpha$.

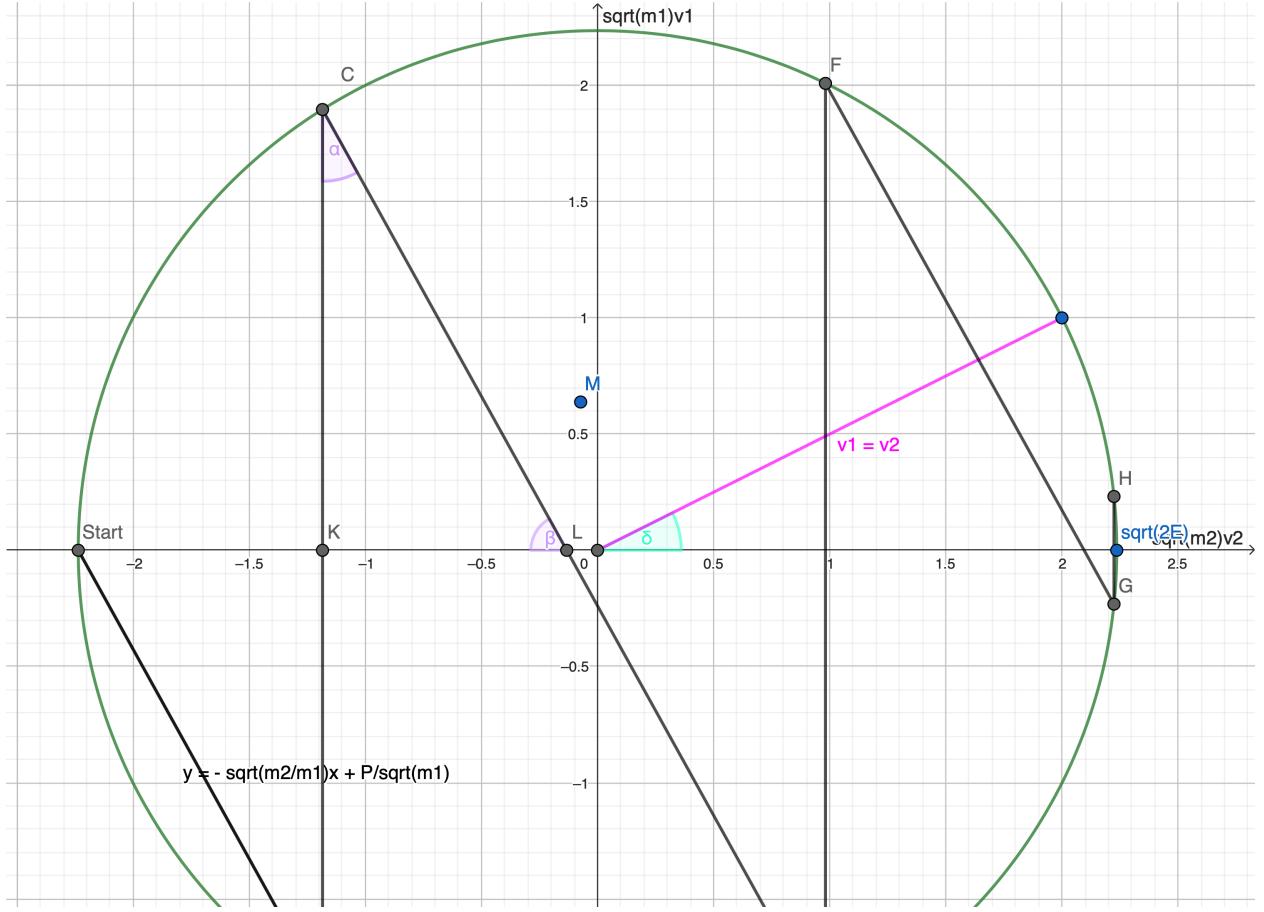


Рис. 5.6: Треугольник CKL

Найдем угол α . Рассмотрим треугольник CKL. Отсюда (6) мы знаем, что $\tan \beta = \frac{\sqrt{m_2}}{\sqrt{m_1}}$

$$\tan \beta = \tan \left(\frac{\pi}{2} - \alpha \right) = \cot \alpha = \frac{\sqrt{m_2}}{\sqrt{m_1}}$$

$$\tan \alpha = \frac{1}{\cot \alpha} = \frac{\sqrt{m_1}}{\sqrt{m_2}}$$

$$\alpha = \arctan \frac{\sqrt{m_1}}{\sqrt{m_2}} \sim \frac{\sqrt{m_1}}{\sqrt{m_2}}$$

Так можно делать, так как у нас $m_2 > m_1$

Найдем угол δ . Методом замены координат несложно понять, что уравнение прямой $v_1 = v_2$ имеет вид: $y = \frac{\sqrt{m_1}}{\sqrt{m_2}}x$,
 значит $\tan \delta = \frac{\sqrt{m_1}}{\sqrt{m_2}}$. Аналогично $\delta \sim \frac{\sqrt{m_1}}{\sqrt{m_2}}$
 Отсюда $\delta \leq \alpha$, а $|\gamma| < \delta$ по условию, так как все соударения закончатся в тот момент, когда

$v_1 < v_2$.

Итого, получили, пусть N - количество соударений. Тогда:

$$N \cdot 2\alpha < 2\pi$$

$$N \cdot \alpha < \pi$$

$$N < \frac{\pi}{\alpha}, \text{ где } N \text{ - максимальное целое}$$

$$\alpha = \arctan \frac{\sqrt{m_1}}{\sqrt{m_2}} \sim \frac{\sqrt{m_1}}{\sqrt{m_2}}$$

5.3 Проверка

5.3.1 $m_2 = m_1 \cdot 10$

$$\alpha = \arctan \frac{1}{\sqrt{10}}$$

$$N \cdot \arctan \frac{1}{\sqrt{10}} < \pi$$

Максимальное целое $N = 10$

5.3.2 $m_2 = m_1 \cdot 10^2$

$$\alpha = \arctan \frac{1}{\sqrt{100}} = \arctan \frac{1}{10}$$

$$N \cdot \arctan \frac{1}{10} < \pi$$

Максимальное целое $N = 31$ Можно заменять арктангенс на экиваленту, с учетом того что арктангенс немного меньше. Тогда лучше видна связь с числом пи:

$$N < 10\pi$$

5.3.3 $m_2 = m_1 \cdot 10^3$

$$\alpha = \arctan \frac{1}{\sqrt{1000}}$$

$$N \cdot \arctan \frac{1}{\sqrt{1000}} < \pi$$

Максимальное целое $N = 99$

5.3.4 $m_2 = m_1 \cdot 10^4$

$$\alpha = \arctan \frac{1}{\sqrt{10000}} = \arctan \frac{1}{100}$$

$$N \cdot \arctan \frac{1}{100} < \pi$$

Максимальное целое $N = 314$ Можно заменять арктангенс на экиваленту, с учетом того что арктангенс немного меньше. Тогда лучше видна связь с числом пи:

$$N < 100\pi$$

5.3.5 $m_2 = m_1 \cdot 10^5$

$$\alpha = \arctan \frac{1}{\sqrt{100000}}$$

$$N \cdot \arctan \frac{1}{\sqrt{100000}} < \pi$$

Максимальное целое $N = 993$

5.3.6 $m_2 = m_1 \cdot 10^6$

$$\alpha = \arctan \frac{1}{\sqrt{10^6}} = \arctan \frac{1}{1000}$$

$$N \cdot \arctan \frac{1}{1000} < \pi$$

Максимальное целое $N = 3141$ Можно заменять арктангенс на экиваленту, с учетом того что арктангенс немного меньше. Тогда лучше видна связь с числом пи:

$$N < 1000\pi$$

Действительно оказалось, что количество соударений не зависит от начальной скорости, а также связь с числом пи при четных степенях подтвердилаась, она вылезает из за корня.

Выводы

В ходе работы мной была исследована математическая модель абсолютно упругого удара двух тел разной массы. Были выведены формулы скоростей после взаимодействия (3) и (4). С помощью этого удалось выполнить моделирование на языке программирования c++, с результатами можно ознакомиться тут. После было проведено исследование, в ходе которого удалось вывести формулу количества соударений тела 1 до полной остановки. Результаты, полученные моделированием оказались верными. Также удалось узнать, что количество соударений не зависит от начальной скорости, а также, если $m_2 = m_1 \cdot 10^{2k}$, то число соударений равно первым $k+1$ знакам числа пи.