# Описание работы программы

Если тело бросить под углом к горизонту, то в полете на него действуют сила тяжести и сила сопротивления воздуха. Если силой сопротивления пренебречь, то остается единственная сила – сила тяжести. Поэтому вследствие 2-го закона Ньютона тело движется с ускорением, равным ускорению свободного падения .

Введем в программный код константу, представляющую ускорение свободного падения.

var g = 9.80665; // ускорение свободного падения, м / с^2

Проекции ускорения на координатные оси равны:



Проекции скорости тела, следовательно, изменяются со временем следующим образом:



где  - начальная скорость,  - угол бросания.

По закону сохранения импульса, до взаимодействия:



где  - импульсы клюшки и шарика до взаимодействия соответственно,  - массы клюшки и шарика,  - скорости клюшки и шарика до взаимодействия.

В программном коде зададим константы для значений масс:

var m1 = 1.5; // масса клюшки, кг

var m2 = 0.1; // масса шарика, кг

После взаимодействия характеристики объектов изменятся следующим образом:



где  - импульсы клюшки и шарика после взаимодействия соответственно,  - массы клюшки и шарика,  - скорости клюшки и шарика после взаимодействия.

Согласно 3-му закону Ньютона, , следовательно:





Поскольку начальная скорость шарика  (шарик покоится), то



Скорость шарика после взаимодействия (удара):



Скорость клюшки до и после удара можно определить, если отнести ширину  датчика ко времени прохода клюшки над датчиком.

Зададим в программном коде константу, определяющую ширину датчика.

var S = 0.01; // ширина датчика, м

Время пролета клюшки над первым и вторым датчиками можно получить, зная разницу времени их срабатывания. Время срабатывания хранится в property вещи.

var t0 = me.h1 - me.l1; // время пролета клюшки над первым датчиком, с

var t1 = me.h2 - me.l2; // время пролета клюшки над вторым датчиком, с

Рассчитаем скорость клюшки до и после удара.

var v01 = S / t0; // скорость клюшки до удара, м / с

var v1 = S / t1; // скорость клюшки после удара, м / с

Тогда начальная скорость мяча будет определяться следующим соотношением.

var v0 = m1 \* (v01 - v1) / m2; // м / с

Положим угол бросания (константа как характеристика определенной клюшки).

var alpha = Math.PI / 4; // радиан

Координаты тела, следовательно, изменяются так:



Если принять за начало координат вторую линию положение второго датчика, то , . Тогда:



В момент приземления координата . Отсюда получаем для времени полета:



Второе значение времени, при котором высота равна нулю, равно нулю, что соответствует моменту бросания.

Дальность полета - это значение координаты  в конце полета, т.е. в момент времени, равный :



Дальность полета мяча в метрах в программном коде присвоим выходной переменной.

result = Math.pow(v0, 2) \* Math.sin(2 \* alpha) / g;

# Комментарии по модели

Разумеется, в рассматриваемой модели механика реального удара при игре в гольф сильно упрощена. В модели принимаются следующие допущения.

* Существует только по одному датчику в первой и второй линиях, т.е. удар может производиться только по прямой.
* После приземления мяч не отскакивает (абсолютно неупругий удар).
* Датчики посылают серверу время начала и окончания пролета клюшки, данные сохраняются в property объекта thing.
* Все пересылаемые данные исключительно корректные (например, время начала пролета меньше времени окончания и т.п.), не существует потерь при передачи по каналу связи.

Такая модель имеет низкую конкурентоспособность на рынке производителей устройств для эмуляции удара при игре в гольф.

Модель рекомендуется использовать в учебных целях для освоения и закрепления знаний о физических законах взаимодействия двух тел.