# Краткий гайд по проге BH\_engine.cpp

Оглавление:

1. Общее введение и принципы
2. Mag\_and\_coils.h – магнитные поля и взаимодействия

2.1 Сетка поля и немного физики

2.2 Магниты

2.3 Соленоиды

3. Gas.h – газодинамика и химия

3.1 Физика и формулы для расчётов

3.2 Входные параметры газа

3.3 Термодинамиеские циклы и их реализация

4. Graph\_objects и пр. -- графоний и прочее

5. Текущие задачи

6. Приложения

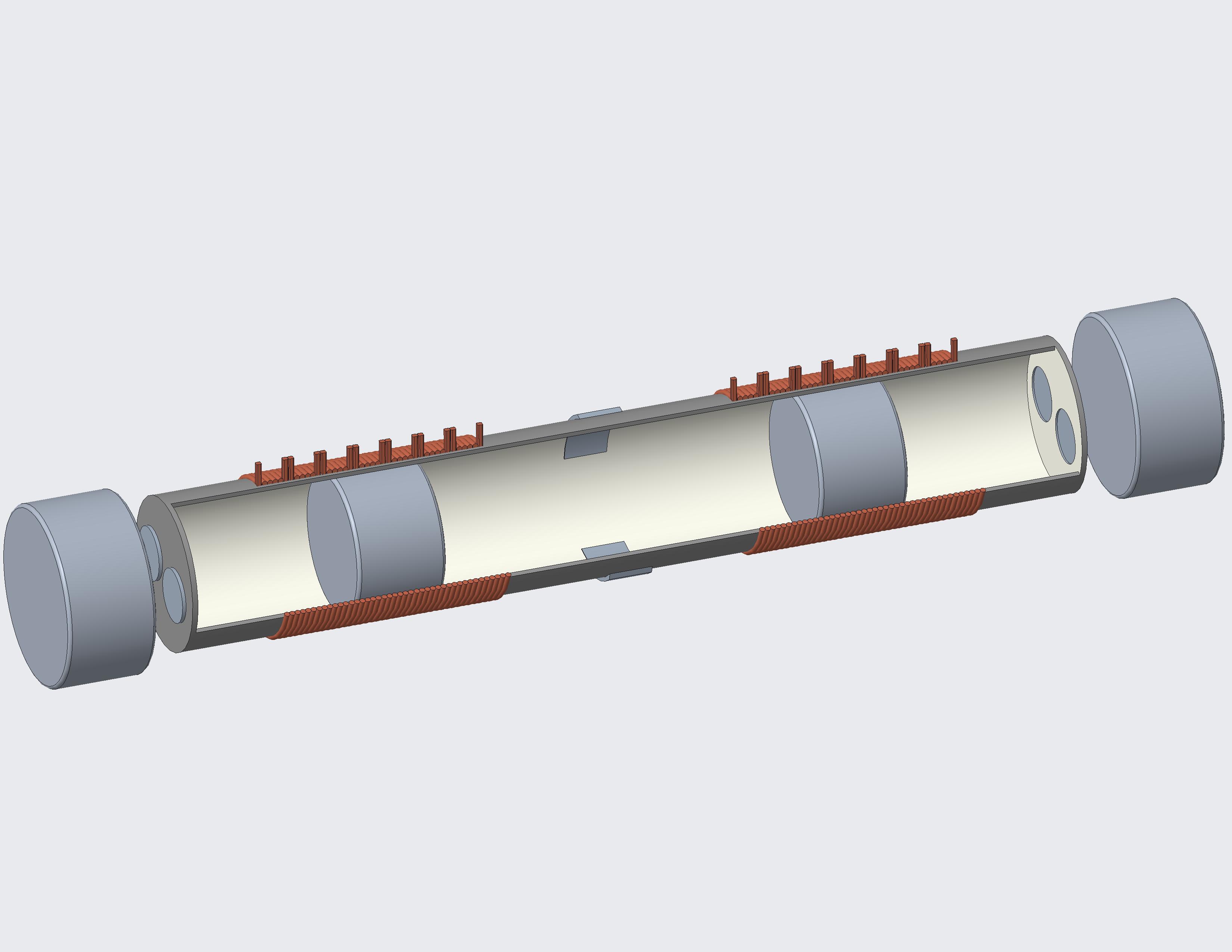
# 1.Введение

Для чего эта работа нужна вы понимаете, и автор предполагает, что вы разделяете с ними заинтересованность в данной тематике и стремление создать рабочий движок на обговоренных с вами принципах.

В данной cpp-шной программе крайне активно юзается написанная автором библиотека для С Vecters.h, созданная для максимальной компактизации кода и упрощения физических вычислений. Она включает в себя максимально полный набор операций над обыкновенными трёхмерными векторами в виде класса vecter с набором функций и переопределённых операторов. В ней присутствуют операции : +,-, скалярное умножение, векторное умножение, умножение на константу, повороты вокруг основных осей,\*=, +=, -= и тд. Просьба ознакомиться.

Так же вместо классов в коде юзается древнее struct, потому что автор недолюбливает члены private и жаждет иметь доступ ко всему, чтобы ставить костыли.

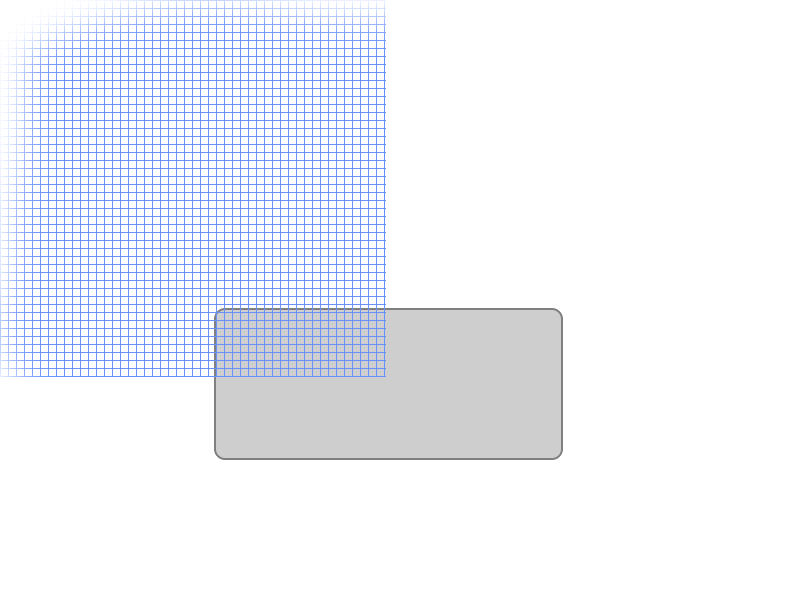
Принцип структуры кода заключается в создании готовых объектов с координатами, с функциями, которые умеют вычислять взаимодействие объектов друг с другом.



Напоминание о строении и принципе работы двигателя найдёте в приложении. Пока ток картинка. Для симуляции использовано именно это самопальное ПО, так как автор не смог найти не одного приложения, в котором без гигантских и сложных и плохоработающих костылей можно было бы одновременно симулировать магнитные поля, газодинамику и химические реакции, да ещё и достаточно оптимизированное под эти цели. Мб автор лошара, если это так, то просьба сообщить в издательство.

# 2.Магниты, катушки и поля

2.1Сетка поля

Для расчёта сил взаимодействий магнитов и прочих магнитных элементов была создана структура, которую можно назвать сеткой поля. Учитывая осевую симметрию (магниты, катушки, да и почти все остальные элементы двигателя обладают данной симметрией), расчёт можно производить только двумерный, тогда все остальные расчёты сведутся к взятию интеграла на один раз меньше за счёт умножения на константу 2π и радиус. Для начала опишем сетку с физической точки зрения:

Y

X

(x,y)

(0,0)

Сетка поля представляет собой двумерное векторное поле, соответствующее реальному магнитному полю от постоянного магнита/катушки. За координаты (0,0) взят центр магнита, ось X направлена параллельно вектору намагниченности постоянного магнита/ оси соленоида. Чтобы напомнить читателю нахрен вообще нужно магнитное поле запишем уравнения Максвелла:

;

,где F = dα— дифференциальная форма поля (кососимметрический тензор поля ранга 2 в четырёхмерном пространстве Минковского), — звёздочка Ходжа, j – четыре-вектор тока.

Хех, да я рофляю. На самом деле, в работе пригодится только сила Ампера (или же закон Лоренца, как читателю угодно), закон БСВ (Био-Савара-Лапласа, он же частный случай теоремы о циркуляции без тока смещения), магнитная индукция Фарадея, да принцип суперпозиции полей. Если последний не нуждается в пояснении, то остальное стоит напомнить (в основном для ФИВТов напрочь забывших физон со школы).

Закон ампера (в приближении что все провода/участки тока тока тонкие (простите за тавтологию)) в человеческом виде:

,где – кусок силы, – кусок длинны, j – ток в куске, – магнитное поле там же.

Именно по этому закону можно посчитать силу взаимодействия магнитов и соленоидами с током (проинтегрировав по всем их виткам dF в поле от конкретного магнита, мы считается равномерно намагниченным, и обладающим относительной магнитной проницаемостью близким к единице (что для реальных хорошо изготовленных магнитов не титанических размеров является вполне хорошим приближением), его можно представить в виде стенок цилиндра того же радиуса, что и настоящий магнит (минус защитное покрытие) с током перпендикулярно оси (вектору намагниченности) обладающим некой линейной плотностью. Переформулируя, можно сказать, что ток некой плотности течёт только вдоль боковых стенок магнита. Пояснение этого будет в пункте про магниты.

Основываясь на предыдущем факте как раз программа и рассчитывает магнитное поле от магнита. То есть в приближении, что магнит является непрерывным набором витков с током, для каждой точки пространства можно взять интеграл БСВ всем токам по боковым границам магнита (с некоторой точностью для расчётов ограничиваемся дискретным набором витков с током). Закон БСВ формулируется так:

, где – константа, – кусок проводника с током для которого считается поле, – радиус-вектор от точки пространства, в которой вычисляется поле, до куска проводника.

Таким образом расчётом магнитного поля является расчёт интеграла в каждой точке пространства, что является вычислительно сложной задачей. Благо, симметрия системы и факт того, что вычисления требуется произвести для каждого объекта всего один раз благодаря принципу суперпозиции полей и предположению, что все элементы конструкции являются магнитожёсткими или немагнитными (грубо говоря, не намагничиваются во внешнем поле, как сердечник трансформатора, усиливая и меня внешнее поле).

Далее необходимо упомянуть основной принцип выработки профита двигателя – магнитную индукцию:

,

,где – вектор нормали к площади.

Это означает, что в каждый момент времени в каждом витке каждой обмотки двигателя (соленоида/токосъёмника) возникает ЭДС равная изменению магнитного потока Ф в нём, и, соответственно, возбуждается ток. Благодаря управляющему контроллеру ток в обмотках можно будет регулировать.

Теперь, когда читатель вспомнил основы электрода применительно к сей конструкции, можно обсудить структуру кода в Mags\_and\_coils.h, отвечающую за магнитные поля. В проге используется структура

typedef struct Field\_2D

{

void \*magnit;

Field\_cell \*\*cells;

int size\_x, size\_y;

double mult;

}

Field\_2D;

Для автор уже не помнит чего, в ней присутствует указатель на магнит/другой объект, к которому она привязана, его размеры по двум измерениям (оси совпадают с определёнными в матчасти), число mult (multiplier, отвечающий за пропорциональность магнитного поля в случае изменения тока/намагниченности – в случае изменения тока в соленоидах изменяется multiplier, но не пересчитывается само поле). Основным членом структуры являет двумерный массив «клеток»:

typedef struct Field\_cell

{

vecter Mag\_vec;

} Field\_cell;

Почему не использовать Бритву Окамы и не плодить сущностей (заменить просто на массив векторов, а не клеток)? Изначально планировалось для увеличения точности вычисления впихнуть в клетку ещё производные вектора в каждой точке (производная вектора представляет собой матрицу частных производных разных компонент вектора по направлениям). Но чёт не пригодилось, хоть и эти производные считаются аналитически.

2.2 Магниты

Для начала опишем, чем является постоянный магнит с физической точки зрения (для не шарящих физон, остальные могут пропустить. Шарящим просьба не бить автора камнями)

Магнит – это кусок дерьма (в нашем случае, цилиндрической формы), обладающий магнитными свойствами засчёт вещества, из которого он состоит. Это вещество является так называемым магнитожёстким, то бишь сохраняющим свои магнитные свойства длительное время, причём даже во внешних полях до определённой велечины. Природой магнитных свойств дерьма является его внутренняя структура, как правило оно состоит из так называемых магнитных доменов. Условно можно считать, что каждый домен является некой от природы намагниченной структурой, то есть вокруг каждого домена вокруг оси (направления) течет ток – домен является минимальным магнитным диполем в материале. В размагниченном материале эти диполи повернуты хаотично. Но в сильных полях эти диполи за счёт их взаимодействия с внешним полем можно повернуть, при выключении поля все домены останутся в повёрнутом положении, так и получается постоянный магнит. Так как в намагниченном материале каждый домен можно представить, как виток тока, автору сложно сформулировать словами, смотрите крч:

Соседние токи нейтрализуют друг друга, и из влияющих на поле токов остаются только токи в боковых стенках магнита.

Важным и негативным свойством таких материалов является, что в случае нагрева дерьма выше некой критической температуры, дерьмо начинает терять свои магнитные свойства, пока не потеряет их совсем при т.н. температуре Кюри, которая зависит от сорта дерьма. (Часть для нешарящих физон кончилась) На данный момент науке известно три основных сорта дерьма, с достаточно сильными магнитными свойствами для вашей с автором совместной авантюры: NdFeB, SmCo и AlNiCo. Первый, хоть и самый распространённый, дешёвый и сильный, начинает терять свои свойства уже при 80С, SmCo держит свои свойства до 400С и лишь немного слабее первого сорта, а последний держит иногда до 500+С, но в несколько раз слабее своих собратьев. Так как в обыкновенных движках температура редко превышает 300С, автор предполагает использование дерьма сорта SmCo.

Теперь определим параметры, которые отвечают за магнит, чтобы вбить их в прогу. Вместо намагниченности материала проще использовать т.н. *коэрцитивную силу*, так как её проще найти в таблицах и использовать. Она вычисляется в [A/m] (амперах на метр), и является, фактически, линейной плотностью тока на боковых границах магнитов. Типичное значение – порядка 10^6 A/m. Остальными параметрами магнита, как объекта в пространстве, являются линейные размеры (высота h, радиус r), местоположение в пространстве, скорость движения, вес. Так же в объект магнита для упрощения работы можно впихнуть силы, действующие на него, и, как следствие, ускорение. Рассмотрим структуру

typedef struct Magnet

{

double strength, diametr, radius, weight, height;

double diametr\_SI, height\_SI, convert\_to\_SI;

vecter position, position\_SI, speed, speed\_SI, accel, force\_on;

Field\_2D Mag\_field;

vecter direction;

Gas\* left;

Gas\* right;

int sight;

void relocate(double dtime)

{

accel = force\_on \* (1 / weight);

speed.y\_proj = accel.y\_proj = speed.z\_proj = accel.z\_proj = 0;

position = position + speed\*dtime + accel\*(dtime\*dtime / 2)\*convert\_to\_SI;

speed = speed + accel\*dtime\*convert\_to\_SI;

speed\_SI = speed / convert\_to\_SI;

position\_SI = position / convert\_to\_SI;

}

Magnet()

{… }

Magnet(int destiny\_of\_mag\_pixels)

{… }

Magnet(int destiny\_of\_mag\_pixels, double \_strength, double \_diametr, double \_weight,

double \_height, vecter \_position, vecter \_speed, int \_sight)

{ }

void show\_field(int cherez\_n)

{ … }

void Set\_Field2D\_Conf()

{ … }

void Set\_Field2D\_Conf(int prec\_H, int prec\_R, int \_size\_x, int \_size\_y)

{ }

vecter Mag\_Mag\_F(Magnet mag, int H\_prec)

{ }

}

Magnet;

Автор когда кодил это говно опять бредил тотальным контролем над происходящим и впихнул кучу запасных конструкторов, которые не юзаются. Так же структура изначально написана для случайного направления магнита, а не только вдоль оси двигателя, что было необходимо автору для другого проекта. Здесь direction не используется. Названия подобраны интуитивно понятными, так что все параметры понятно что означают. Надо только понимать, что многие параметры имеют подпись \_SI, и это значит, что они написаны для координат объектов «в реальной жизни», а не в «условных единицах», как их клоны (подробнее в следующем абзаце). В структуре так же можно заметить ссылки на такие объекты, как Gas слева и справа. Эти объекты в прямом смысле задают газ, о их физике и структуре кода поговорим позже.

Теперь о методах. Во-первых, в конструктор пихаются все начальные условия, заданные в СИ, а также переменная destiny\_of\_mag\_pixels отвечающая за плотность ячеек поля на метр длины. Например, если это значение 1000, значит магнитное поля магнита будет вычисляться по клеткам 1\*1 мм. Собственно, «условные единицы» и соответствуют размеру ячеек.

Метод Set\_Field2D\_Conf, как ни странно, рассчитывает магнитное поле по БСВ размером size\_x на size\_y в условных единицах и привязывает его к магниту. Параметры int prec\_H, int prec\_R в этом методе являются точность вычисления поля. Численно – prec\_R является числом углов в правильном многоугольнике, которым мы приближаем круглый виток, а prec\_H – удвоенным количеством витков с током, на который магнит разбивается.

Метод Mag\_Mag\_F рассчитывает силу взаимодействия с другими магнитами по закону Ампера. Принимает на вход другой магнит и prec\_H (радиальной точности не присутствует благодаря радиальной симметрии задачи). Для системы без радиальной симметрии код тоже существует, более того он умеет рассчитывать ещё и моменты сил и энергии взаимодействий (один из читателей знает), но в данном случае он не пригодится.

Ну, и на последок, метод relocate осуществляет перемещение магнита за единицу времени. Очевидно, чем меньше dtime, тем точнее модель.

2.3 Катушки

С точки зрения нашей задачи о катушках необходимо знать, что они обладают ненулевым электрическим сопротивлением (которое, кстати, необходимо минимально возможное), неким количеством витков, в них может течь ток, регулируемый контроллером, в них та же осевая симметрия, витки будут идти достаточно плотно и состоять из достаточно тонкой проволоки, чтобы можно было считать катушку набором витков с током. Так же, опять присутствуют такие параметры как радиус (всегда больший радиуса магнита), длинна, расположение. В отличии от магнитов, объект статичен, но, тем не менее, силу взаимодействия удобнее считать именно магнита на катушку, а не наоборот (с вычислительной точки зрения, чтобы не считать лишние поля). Вообще говоря, поля катушек всё равно хотя бы один раз считать придётся, чтобы было возможно определить положение поршней с помощью датчиков Холла.

Рассмотрим структуру

typedef struct Coil

{

double diametr, radius, height, flow[2], current, voltage, resist;

double diametr\_SI, height\_SI, convert\_to\_SI;

vecter position,position\_SI, force\_on;

Field\_2D Mag\_field;

int turns;

vecter direction;

Coil()

{ }

Coil(int turns\_num, int destiny\_of\_mag\_pixels)

{ }

Coil(int destiny\_of\_mag\_pixels, double \_current, double \_diametr,

double \_height, vecter \_position, int \_turns)

{ }

double flow\_X\_from\_mag(Magnet& magnit)

{ }

vecter F\_from\_coil(Magnet mag)

{ }

}

Coil;

К ней применимо почти всё, что сказано о структуре Magnet. Надо заметить, что в ней присутствует член flow[2], то есть сумма потоков вектора магнитной индукции сквозь все витки в настоящий момент времени, и та же величина dtime назад, чтобы можно было вычислять ЭДС (voltage) по магнитной индукции. Метод flow\_X\_from\_mag вычисляет поток от конкретного магнита сквозь катушку (стоит помнить принцип суперпозиции в случае большого числа магнитов и других катушек). Автор забыл сделать flow\_X\_from\_coil, но это быстро фиксится.

F\_from\_coil вычисляет силу взаимодействия катушки и магнита. На силы взаимодействия катушки с катушкой всем плевать, они статичны

# 3.Gas.h— газы

Данный раздел будет уже далеко не столь тривиальным с физической точки зрения, хотя код всё ещё достаточно тривиален.

Очевидно, основа основ любого ДВС (двигателя внутреннего сгорания), это сгорание некой топливной смеси образованием тепла, которое потом преобразуется в полезную работу. А значит, для понимания физических процессов в нашем двигателе пригодятся такие вещи, как термодинамика, статфизика и химическая кинетика. Если первый раздел всем хотя бы немного знаком ещё со школы, то химическая кинетика – раздел физхимии о протекании химических реакций – будет многим читателям незнаком (кроме людей с ФАКИ и БМ).

И так, начнём с тривиальной термодинамики: всем известное уравнение Менделеева-Клайперона (идеального газа):

, где P— давление газа, V – его объём, T – температура, – количество (моль), R = – универсальна газовая постоянная.

Как можно догадаться, реальные физические процессы вряд ли могут выражаться в столь тривиальных законах, так что существует область применения данного закона. Так как вывод данного уравнения сделан в приближении, что молекулы не взаимодействуют между собой и занимают пренебрежимо малый объём, для наших целей (молекулы значительно взаимодействуют, степень сжатия в ДВС может быть достаточно большой, чтобы их объём не был совсем пренебрежимо мал) нам следует ввести коррекции в данную формулу. Запишем уравнение Ван-Дер\_Ваальса: