Ещё один ПИД-регулятор

Даниил Барков

Лето 2025

Теория

В сегодняшней статье расскажу, как на стенде измеряется угол, чем обеспечивается защита от дурака, и как мне помог чатжипити.

Итак, рабочая часть стенда представляет собой неуравновешенную балку с мотором и датчиком, называемыми вместе лучом.Исходное состояние луча - висит вертикально вниз. Тяга пропеллера создаёт момент силы, вращающий луч против часовой стрелки. Примем это направление за положительное, ограниченное углом 170°. В этом диапазоне сила тяжести всегда вращает луч в отрицательном направлении.

Измерение угла

Первостепенная задача - измерение угла отклонения луча от исходной позиции. Затем нужно проверить, что луч вращается в вертикальной плоскости для предотвращения включения стенда в неправильном положении или выключения мотора при заваливании. Первую задачу можно решить при помощи энкодера на оси, а вторую - административно. Но у квадрокоптера нет энкодеров, а аварийные ситуации случаются самопроизвольно. Поэтому буду использовать инерциальный датчик MPU6050 в модуле GY-521.

В одной из статей я описал способ определения ориентации инерциального датчика. Она представляется в виде матрицы 3×3, в которой столбцы соответствуют осям датчика, а строки - проекции на север, запад и вертикаль. Казалось бы, прикрепи датчик к балке, посмотри направление оси $\overrightarrow{z'}$, запомни направление $\overrightarrow{y'}$, после поворота посчитай угол между прежним и текущим направлением. Но такой вариант доступен только при полной уверенности в том, что ось датчика $\overrightarrow{z'}$ параллельна оси луча, а я не стану гарантировать это.

Поэтому надо обобщить обе операции для случайного закрепления датчика. Как и раньше, алгоритм прототипируется в матлабе. МК обрабатывает данные датчика, вычисляет матрицу и отправляет её компьютеру по USB-CDC в виде массива из 9 байт. При запуске стенда луч неподвижнен. Программа запоминает изначальную ориентацию датчика (M_1). Матрица вращается вокруг вектора $\overrightarrow{a}$, и получается матрица M_2. Этот вектор и является осью вращения, надо вычислить его. Что мы знаем об $\overrightarrow{a}$? Его проекции на M_1 и M_2 одинаковы, при этом матрицы связаны матрицей перехода:

$$M_{tr} = M_2 \cdot M_1^{-1}$$

То есть, при применении к $\overrightarrow{a}$ линейного оператора M_{tr} снова выходит $\overrightarrow{a}$. Получается, что $\overrightarrow{a}$ является собственным вектором для матрицы перехода. Так как матрица имеет размерность 3, то и собственных векторов тоже 3. Как выбрать нужный? Посчитать

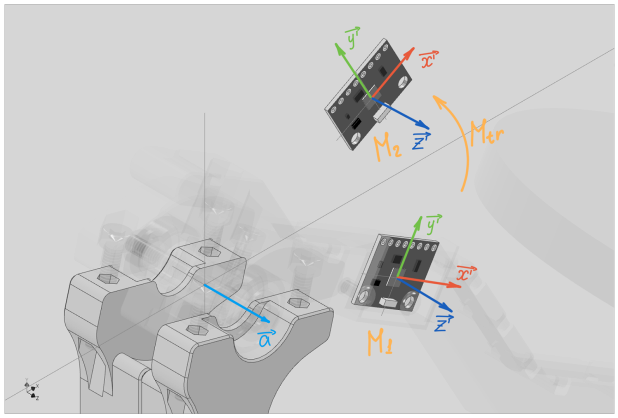

Рис. 6: Поворот датчика вокруг оси

собственные числа M_{tr} и выбрать вектор, соответсвующий $\lambda = 1$. Почему? Потому что собственное число - это скаляр, который растягивает собственный вектор. При повороте растяжения не происходит, значит $\lambda = 1$.

У этого решения есть несколько нюансов. Во-первых, погрешности измерений, расчётов и неровность механики приводят к тому, что M_{tr} не в полной мере является матрицей поворота, и $\lambda \neq 1$. Поэтому приходится выбирать число наиболее близкое к 1, то есть, равное $min(|\lambda - 1|)$. Во-вторых, вычисление собственных чисел матрицы - трудоёмкая задача как для микроконтроллера, так и для программиста. Прежде чем кинуться в написание сишных функций, я обратился за советом к чатужипити и не пожалел.

Совет 1. Транспонировать матрицу вместо вычисления обратной. Для ортогональной матрицы, к которой относится матрица ориентации, $M^{-1} = M^T$. Транспонировать матрицу проще, и это снизит нагрузку на работу МК.

Совет 2. Использовать след матрицы поворота для вычисления угла.

$$tr(M_{tr}) = M_{tr(11)} + M_{tr(22)} + M_{tr(33)} = 1 + 2cos(\psi)$$

$$\psi = acos\left(\frac{tr(M_{tr}) - 1}{2}\right)$$

ψ - угол поворота матрицы поворота.
Чтобы программа не ломалась, нужно ограничить значение $tr(M_{tr})$ отрезком [-1; 1]. Теоретически, значение следа не должно вылезать за эти границы, но увы, ошибки вычислений. Данный метод возвращает значения от 0 до π, но это не проблема для моего стенда.

Совет 3. Извлекать ось вращения из антисмимметричной части матрцы поворота.

$$\overrightarrow{a} = \frac{1}{2sin(\psi)} \cdot \begin{bmatrix} M_{tr(32)} - M_{tr(23)} \\ M_{tr(13)} - M_{tr(31)} \\ M_{tr(21)} - M_{tr(12)} \end{bmatrix}$$

Таким образом, вычисления угла и оси стали быстрыми и непринуждёнными. При запуске стенда необходимо перевести луч из нижнего положения в положение повыше. Из полученных данных будет вычислен вектор оси. Угол между осью и вертикалью должен иметь значение, близкое к $0,5\pi$. Сильное отклонение приведёт к переходу стенда в аварийное состояние и отключению мотора до перезапуска программы.

Расчёт параметров модели

Прежде чем угробить стенд жёсткими режимами работы, хотелось бы поиграть в математическое моделирование. Может быть, я получу полезный практический опыт или даже готовые коэффициенты ПИД. Для этого составлю уравнение динамики стенда, наишу ПИД-реулатор для управления тягой, абстрагируюсь от задержки измерения угла и изменения тяги, инерции пропеллера.

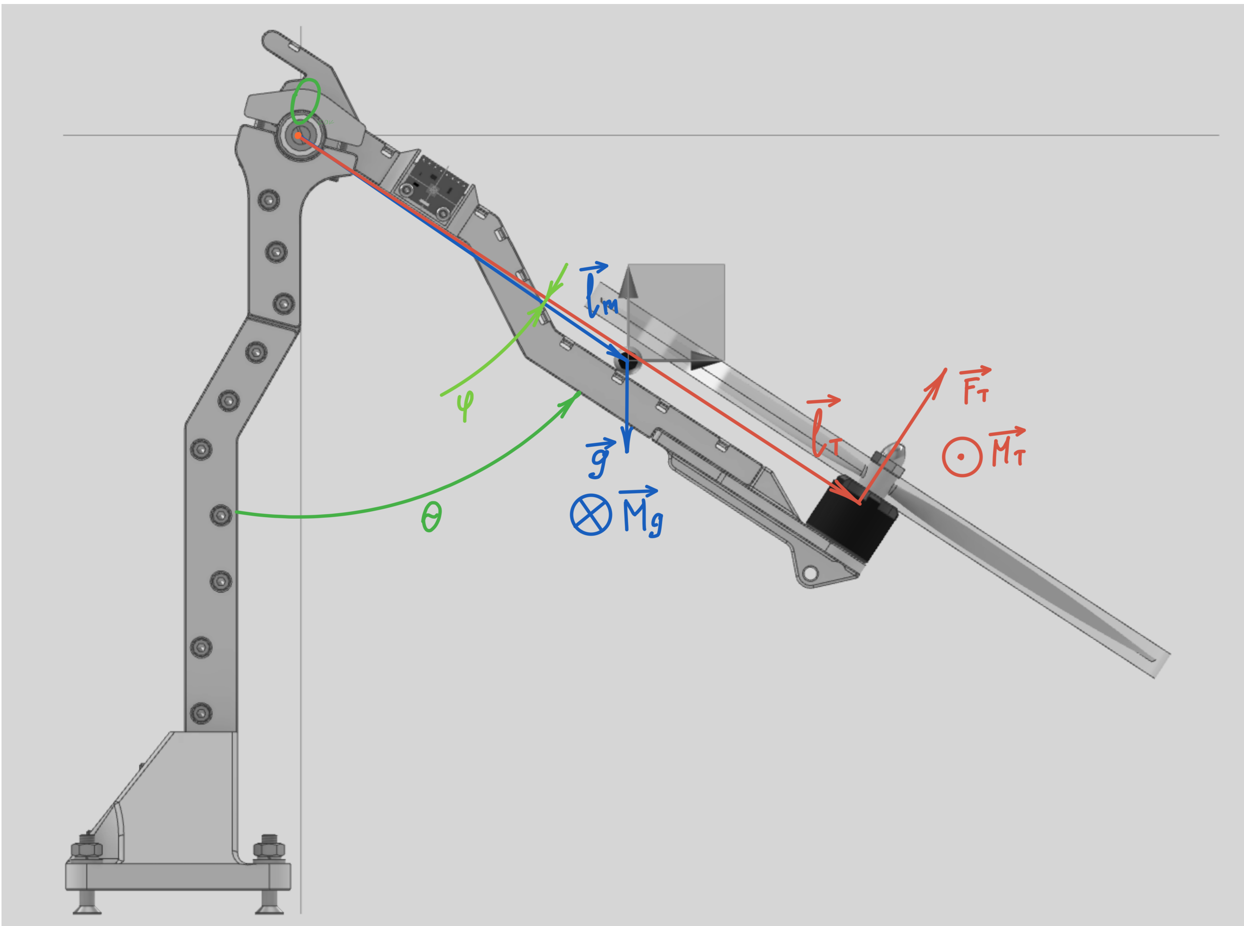

Рис. 7: Схема действия сил

Сила тяжести, действующая на луч и приложенная к центру масс луча, создаёт момент, равный

$$\overrightarrow{M_g} = \overrightarrow{l_g} \times m\overrightarrow{g}$$

или

$$M_g = m \cdot g \cdot l_g \cdot sin(\theta - \varphi)$$

m - масса луча, l_g - расстояние от оси O до центра масс луча, θ - угол отклонения луча, φ - угол между лучом и вектором до центра масс.
Так как отклонение луча ограничено 170°, направление момента не меняется.
Уравнение статики для этой системы выглядит так:

$$\overrightarrow{M_g} + \overrightarrow{M_t} = 0$$

или

$$0 = F_T \cdot l_T - m \cdot g \cdot l_m \cdot sin(\theta - \varphi)$$

l_T - расстояние между осью луча и осью пропеллера, F_T - сила тяги пропеллера.

При дисбалансе сил возникает угловое ускорение:

$$\ddot{\theta} = \frac{F_T \cdot l_T - m \cdot g \cdot l_m \cdot sin(\theta - \varphi)}{J}$$

Необходимо добавить сопротивление среды. К ним можно отнести силу трения покоя в подшипнике, силу сухого трения подшипника, силу вязкого сопротивления смазки и силу сопротивления воздуха. Силу сухого трения и трения покоя я исключу, так как хорошо смазал подшипники вэдэшкой. Включим оставшиеся силы и получим уравнение динамики системы.

$$\ddot{\theta} = \frac{F_T \cdot l_T - m \cdot g \cdot l_m \cdot sin(\theta - \varphi) - k_v\dot{\theta} - k_a\dot{\theta}^2 \cdot sign(\dot{\theta})}{J}$$

J - момент инерции, k_v - коэффициент вязкого сопротивления смазки, k_a - коэффициент сопротивления воздуха.

Момент инерции и расстояние до центра масс вычислены в САПРе. Для этого я взвесил каждую деталь и прописал массу в физические свойства модели. Равномерность распределения массы по объёму является большим допущением для деталей, напечатанных на FDM-принтере, но это лучшее, что я смог реализовать.

k_v и k_a - эмпирические коэффициенты. Я получил их значения экспериментально. Для этого закрепил мачту горизонтально, отвёл луч, позволил ему свободно колебаться под действием силы тяжести и получил функцию затухающих колебаний в табличном виде $\theta_{exp}(t)$. При этом диффур обретает вид

$$\begin{cases} \ddot{\theta} = (-m \cdot g \cdot l_m \cdot sin(\theta - \varphi) - k_v\dot{\theta} - k_a\dot{\theta}^2 \cdot sign(\dot{\theta})) \cdot J^{-1} \\ \theta(0) = -39^\circ \\ \dot{\theta} = 0 \end{cases}$$

Затем написал матлаб-скрипт, численно решающий приведённый диффур с такими же начальными условиями, как у $\theta_{exp}(t)$. Обозначу вычисленные данные как $\theta_{sim}(t)$. Оба графика представлены на рисунке. Как можно увидеть, частоты значительно различаются, примерно в два раза. Она зависит от параметров l_m и J. Подобрав ручками J, я понял, что увеличение его вдвое исправляет проблему. После проверки масс всех деталей, я вспомнил, что мой любимый САПР указывает момент инерции относительно центра масс. Чтобы пересчитать значения J для вращения вокруг O, нужно применить теорему Гюйгенса-Штейнера:

$$J = J_C + m \cdot l_m^2$$

где J_C - момент инерции, рассчитанный САПРом.

Отклонение частоты уменьшилось до 3%. Теперь можно подобрать k_v и k_a. Сделать это можно следующим способом. Так как диффур исчерпывающий, можно надеяться, что

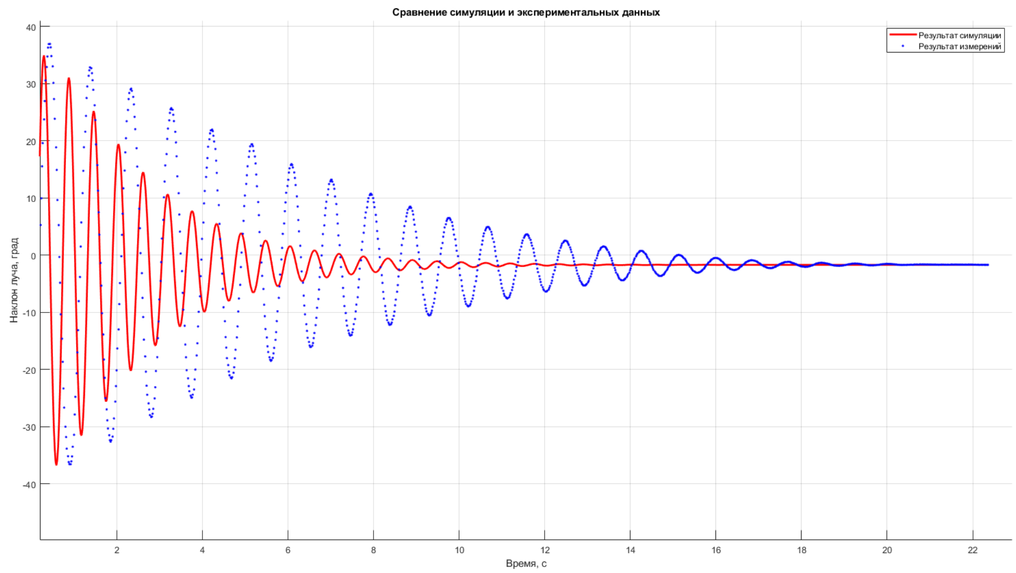

Рис. 8: Это фиаско

при правильном выборе k_v и k_a графики совпадут. Хорошо бы сделать подбор автоматическим и оптимальным по времени. Сперва нужно выбрать критерий совпадения графиков. Сумма квадратов невязок отлично подходит на роль меры.

$$RSS(k_v, k_a) = \sum_{i=0}^{N} (\theta_{sim}(t_i, k_v, k_a) - \theta_{exp}(t_i))^2$$

Но вот незадача. Временные метки у $\theta_{exp}(t)$ и $\theta_{sim}(t)$ различаются, выбирать пары значений θ с одинаковым аргументом не получится. К тому же, частоты незначительно различаются, на шестнадцатом периоде колебания сдвинуты на $0,5\pi$, поэтому RSS никогда не станет равным нулю (И что? Обоснуй. Мб будет плохо сходиться. Почему?). Однако известно, что k_v и k_a характеризуют форму огибающей и практически не влияют на период колебаний. Значит, достаточно сравнивать огибающие этих графиков. Для этого из $\theta_{exp}(t)$ и $\theta_{sim}(t)$ выделяются пики с помощью функции findpeaks, обозначенные $\gamma_{exp}(t)$ и $\gamma_{sim}(t)$. Затем надо интерполировать $\gamma_{sim}(t)$ функцией interp1 методом сплайн, подставить временные метки из $\theta_{exp}(t)$ и получить соответствующие значения $\gamma_{simInt}(t)$. Теперь можно сравнить огибающие. Таким образом вычисление k_v и k_a сводится к минимизации функции RSS:

$$RSS(k_v, k_a) = \sum_{i=0}^{N} (\gamma_{simInt}(t_i, k_v, k_a) - \gamma_{exp}(t_i))^2 \to \min_{k_a, k_v \in R^2}$$

Для этого используется функция fmincon из пакета Optimization Toolboxt. С начальным приближением $k_{v0} = 0, k_{a0} = 0$ решение заняло 16 итераций. Результат можно видеть на графике выше, а полученные значения в таблице.

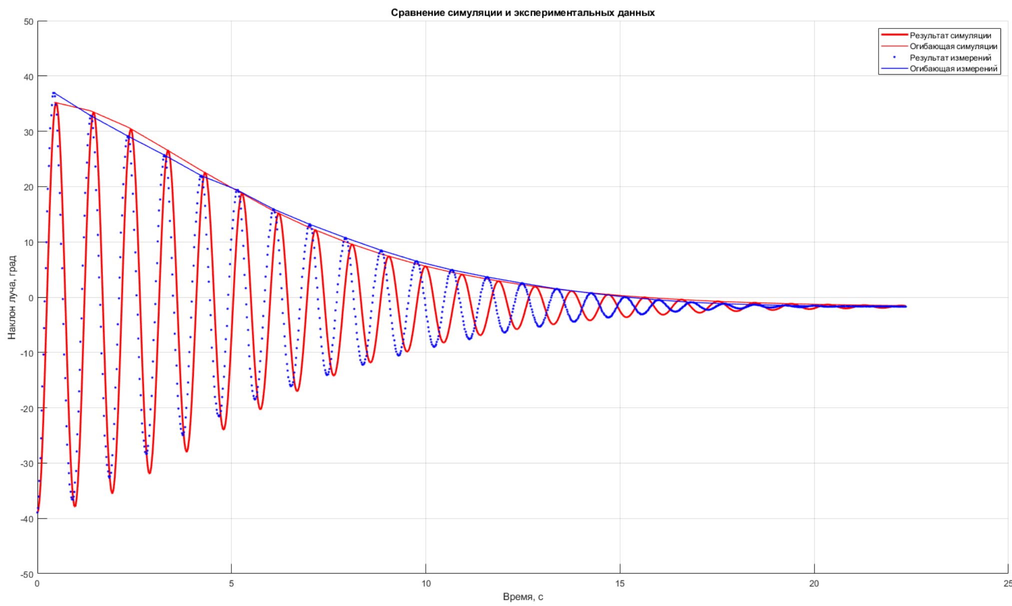

Рис. 9: Графики отлично накладываются

J, кг·м^2	5,52e-3
m, кг	0,181
l_m, м	0,139
ψ, град	1,71
k_v, попугаи	1,85e-3
k_a, попугаи	1,06e-3

Симуляция PID-регулятора

Наконец, можно приступить к моделированию PID-регулятора. Ограничу задачу достижением и поддержанием угла луча, но в будущем попробую также регулировать скорость вращения луча.

Итак, формула PID-регулятора всем известна.

$$\begin{cases} u(t) = K_P \cdot e_{rr}(t) + K_I \cdot \int_0^t e_{rr}(\tau)\, d\tau - K_D \cdot \frac{de_{rr}(t)}{dt} \\ e_{rr}(t) = \theta_{set} - \theta(t) \end{cases}$$

θ_{set} - установочное значение, к которому стремится $\theta(t)$. $e_{rr}(t)$ - ошибка регулирования. $u(t)$ - управляющее воздействие, стремящееся уменьшить ошибку. K_P, K_I, K_D - коэффициенты при пропорциональной, интегральной и дифференциальной составляющей регулятора соответственно. Их значения и определяют работу регулятора, их и предстоит подобрать.

Сперва необходимо врезать в диффур PID-регулятор.

$$\begin{cases} \ddot{\theta}(t) = (F_{max} \cdot u(t) \cdot l_T - m \cdot g \cdot l_m \cdot sin(\theta(t) - \varphi) - k_v\dot{\theta}(t) - k_a\dot{\theta}^2(t) \cdot sign(\dot{\theta}(t))) \cdot J^{-1} \\ u(t) \in [0,\ 1] \\ \dot{\theta}(0) = 0 \\ \theta(0) = \theta_0 \end{cases}$$

F_{max} - максимальная тяга, которую может выдать мотор. θ_0 - Начальный угол Регулятор управляет тягой, которая может быть только положительной и имеет ограничение сверху. Поэтому удобно ограничить также управляющий параметр $u(t)$. Для решения диффура необходимо принять начальные условия $\dot{\theta}(0)$ и $\theta(0)$. Регулятор начинает работу при неподвижном опущенном луче, поэтому они равны нулю.

Это дифференциальное уравнение не решается аналитически. Хорошо, что Матлаб может решить его численно. Скрипт выглядит так:

```
% Parameters
l_m = norm([138.760 -4.145 ])/1000*1;    % Distance to the center of masses (m)
m = 0.181;    % Mass (kg)
psi = atan2(-4.145, 138.760); % Rad
J = 2036.662 /10^6+m*l_m^2; % kg*m^2
g = 9.81; % m/s^2
kv = 0.0001850;
ka = 0.0001058;
l_T = 0.254;
F_T=10;
alpha = deg2rad(90);

% Initial conditions
O0 = deg2rad(0);    % Initial angle (30 degrees)
omega0 = 0;   % initial angular velocity
integral0 = 0;
Y0 = [O0; omega0; integral0];

% time span
tspan = [0 10];

% Solution
[t, Y] = ode45(@(t, Y) pendulumODE(t, Y, l_m, m, g, J, psi, kv, ka, F_T, l_T, alpha, 0.3, 0.5, 0.05, 0.09), tspan, Y0);

% Graphics
figure(1);
clf(figure(1))

plot(t, rad2deg(Y(:,1)), 'r', 'LineWidth', 1.5);
xlabel('Time (c)');
ylabel('Angle theta (degree)');
title('Anle dynamisc');
ylim([0 120])
grid on;

yline(rad2deg(alpha))
```

```
37
38 function dYdt = pendulumODE(t, Y, l, m, g, J, psi, kv, ka, F_T, l_T, alpha, Kp, Ki, Kd, Ks)
39     if Y(1) >= 0.99*pi
40         Y(2) = 0;
41         Y(1) = 0.99*pi;
42     end
43     O = Y(1);            % Angle theta
44     omega = Y(2);        % Angular velocity dO/dt
45     e = (alpha-O);
46
47     d_integral=e;
48
49     if Y(3)>10
50         Y(3) = 10;
51     end
52     if Y(3)<-10
53         Y(3) = -10;
54     end
55     integral = Y(3);
56
57     U = Kp*e+Ki*integral-Kd*omega+Ks*sin(alpha);
58
59     if U>1
60         U = 1;
61     end
62     if U < 0.001
63         U = 0.001;
64     end
65     F = F_T*U;
66
67     domegadt = (F*l_T-m*g*l*sin(O-psi)-kv*omega*t-ka*omega^2*sign(omega)*t)/J;
68
69     dYdt = [omega; domegadt; d_integral];  % Returns vector of derivatives
70 end
```

Функция ode45 решает диффур итерациями, поэтому на каждом шаге можно выполнять математические операции над переменными, сравнивать и менять их значения. Это позволяет встроить в диффур различные проверки. Например, на строках 59-64 реализовано ограничение управляющего параметра.

Пора перейти к подбору коэффициентов. Для этого надо