Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

ОТЧЁТ

По дисциплине «Программирование систем управления»

Выполнили: Шамраев Алексей

Боев Глеб

Преподаватель: Томашевич С.И.

Санкт-Петербург

2021 г.

ЗАДАНИЕ

Необходимо построить ПИД-регулятор к объекту управления, который описывается передаточной функцией:

$$W(s) = \frac{b_0}{s^2 + sa_1 + a_0} \quad (1)$$

В задании необходимо выполнить ряд ключевых действий, присущих процессу разработки программы для регуляторов. Реальный объект управления, как правило, представляет собой физическое устройство, оснащенное датчиками. Взаимодействие с датчиками преимущественно осуществляется через цифровой протокол обмена данными. Таким образом, чаще всего в реальной жизни от объекта управления принимается массив байт, которые необходимо интерпретировать в соответствии с форматом протокола.

1. Получить данные с объекта управления (в программе – объект типа ControllerBackend). Для этого необходимо подключаться к соответствующему сигналу). Синтаксис подключения следующий:

```
QObject::connect(emitter_address, signal_address, catcher_address, slot_address);
```

Для того чтобы концепция сигнал-слот работала, необходимо, чтобы оба объекта были унаследованы от *QObject* (как в предыдущем семестре было сделано в задании с фильтрами). После подключения сигнала к слоту каждый раз, когда сигнал издается, запускается функция в слоте. Таким образом, необходимо создать свой класс, в котором будет производиться вся дальнейшая обработка, унаследоваться от *QObject*, создать слот и подключиться к сигналу. Каждый раз, когда объект управления испускает сигнал об изменении состояния, подключенный слот будет запускаться. В этом слоте необходимо рассмотреть данные, которые приходят от объекта управления. Для этого можно использовать синтаксис вывода в консоль, предварительно подключив соответствующую библиотеку (*# include*):

```
qDebug() << someVariable;
```

2. Разобрать данные, которые приходят с объекта управления в виде массива байтов *QByteArray*. Каждое сообщение содержит стартовый байт, тип команды, непосредственно сами данные и проверочную сумму. Требуется определить, каким образом рассчитывается проверочная сумма,

и реализовать проверку корректности сообщений. Некорректные сообщения необходимо отбрасывать.

3. Для последующей обработки данных в программной среде Matlab необходимо настроить сохранение логов. Для этого используется *QFile* и *QTextStream*, позволяющие осуществлять запись данных в текстовом виде в файл.

4. Написать скрипт в среде Matlab для чтения сохраненного лога, например, с использованием функций *fileread*, *strsplit*, *str2double*. Для проверки правильности отобразить данные и сравнить с тем, что было отображено в программе при записи лога.

5. Идентифицировать параметры объекта управления (1) с использованием метода наименьших квадратов. Для этого провести численное дифференцирование сигналов для получения значений скорости и ускорений. Предварительно данные следует отфильтровать с использованием функции *lowpass* и рассчитать значение частоты среза для фильтра низких частот. Этот фильтр впоследствии необходимо реализовать в коде для фильтрации сигнала выхода с объекта управления перед подачей на регулятор. Для получения корректных параметров необходимо задать вход объекта управления вручную (с использованием курсора) или с использованием подготовленной функции (желательно содержащей как можно больше гармоник). Перед реализацией регулятора необходимо согласовать с преподавателем полученные параметры во избежание неправильной настройки регулятора.

6. Настроить в среде Simulink ПИД-регулятор, например, с использованием специальных инструментов (блок PID Controller). Перевести полученную модель регулятора в вид Вход-Состояние-Выход (непрерывный или дискретный) для последующей реализации в коде. Основное требование: время переходного процесса должно составлять не более трех секунд.

7. Реализовать рассчитанный на предыдущем пункте регулятор в коде. На вход регулятора подавать ошибку между задающим сигналом и

отфильтрованным выходом. Проверить качество регулятора на основе следующего задающего сигнала (на интервале до 60 секунд):

$$70 - 10e^{-0.1t}\left(\frac{-(t-2)^2 + 60t}{100}\right)\sin\left(\frac{t}{2}\right) \tag{2}$$

Для реализации экспоненты, синуса и степени в C++ использовать функции *std::exp(), std::sin(), std::pow()* соответственно. Результирующий лог необходимо загрузить в Matlab и рассчитать качество управления:

$$E = \sum_{i=0}^{n}(r[i] - y[i])^2, \tag{3}$$

где r – задающее воздействие, y – выход объекта управления. Для получения усредненного значения применить формулу

$$\overline{E} = \sqrt{\frac{E}{n}}. \tag{4}$$

ХОД РАБОТЫ

0. Описание программы

Имеется программа в которой эмулируется объект. Для этого объекта нужно синтезировать регулятор. Объект посылает сообщения, регулятор их обрабатывает, посылает управление. Интерфейс программы показан на рисунке 1.

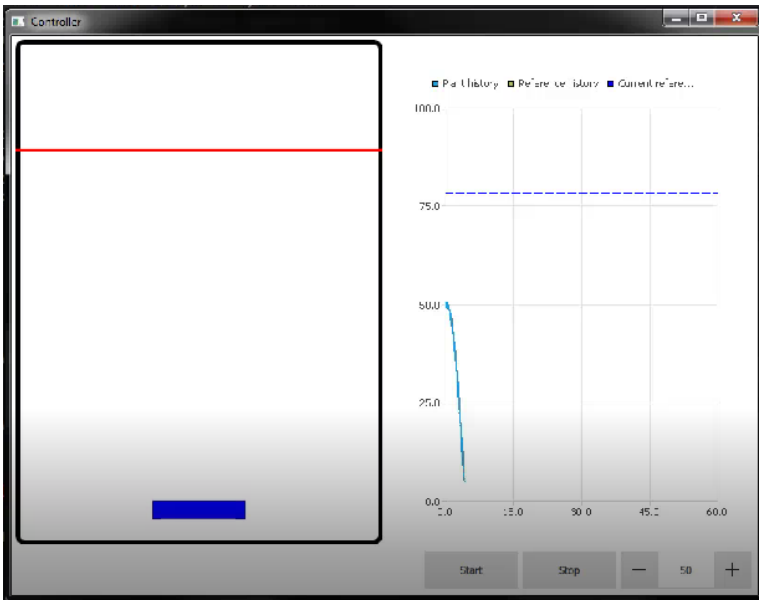

Рисунок 1 - Интерфейс программы, эмулирующей объект управления

1. Получение данных с объекта управления

Контроллер получает такие данные с объекта при нулевых начальных условиях без задавания референсного уровня:

controller received message (1): "aa 0a c7 00 a4 3e a0 3f d8 02 e9" t = 0 s
controller received message (2): "aa 0a 1b 95 6e 3e a0 3f d8 02 36" t = 0.019 s
controller received message (3): "aa 0a e7 ab f2 be a0 3f d8 02 50" t = 0.04 s
controller received message (4): "aa 0a b1 04 90 3d a0 3f d8 02 10" t = 0.059 s
controller received message (5): "aa 0a 61 49 0c bd a0 3f d8 02 1f" t = 0.079 s

Видно, что объект посылает данные с частотой 0.02 с, сам пакет данных имеет 11 байт, причем меняются только 4 байта в середине и 1 в конце.

Контроллер получает такие данные с объекта при начальных условиях 50 (ввод в программу):

controller received message (1): "aa 0a c5 3d 49 42 a0 3f e1 02 fc" t = 0 s
controller received message (2): "aa 0a 6d c5 48 42 a0 3f e1 02 cd" t = 0.02 s
controller received message (3): "aa 0a cb bf 45 42 a0 3f e1 02 78" t = 0.04 s
controller received message (4): "aa 0a 61 aa 47 42 a0 3f e1 02 f5" t = 0.06 s
controller received message (5): "aa 0a 0c ed 46 42 a0 3f e1 02 08" t = 0.08 s

Коммит:
https://github.com/Shamraev/CSP_2021_project_sem2/commit/d883fcc4db5d82891019dd6ea9a49903e672a039

2. Алгоритм проверки корректности сообщений. Разбор сообщения

Как уже было указано, последний байт меняется и в задании сказано что это контрольная сумма.

Итак, **пакет данных состоит из байт**:

1 - стартовый байт

2 - байт команды

3-10 - байты сообщения

11 - байт контрольной суммы

Разберем сообщение:

"aa 0a c7 00 a4 3e a0 3f 55 01 6d"

Последний байт 0x6d или в unicode 109, это и есть контрольная сумма:

$$s = 109$$

Предположим, что контрольная сумма - это просто сумма первых 10 байтов, тогда получим (в unicode):

$$s_{pred} = \sum_{i=0}^{n-1} message(i) = 146,$$

где message - сообщение, n - длина в байтах сообщения.

Не сходится, тогда попробуем:

$$s_{pred} = 255 - \sum_{i=0}^{n-1} message(i) \quad (2)$$

$$s_{pred} = 255 - 146 = 109 = s$$

Следовательно, алгоритм подсчета контрольной суммы производится по формуле (2). В коде это прописали, вывод сообщений:

controller received message (1): "aa 0a c7 00 a4 3e a0 3f 55 01 6d" t = 0 s
predicted checksum if: first 10 sum: 146 , real checksum: 109
predicted checksum if: ff - (first 10 sum): 109 , real checksum: 109

Также, в коде пропишем отбрасывание некорректных сообщений. Вывод консоли:

controller received message (760): "aa 0a 72 13 ea be a0 3f be 02 7f" t = 15.1811 s
predicted checksum if: ff - (first 10 sum): 127 , real checksum: 127
corrupted messages: 16

Как видно, на 15 секундах выявлено 16 некорректных сообщений.

Если менять референсный уровень, то и последние 4 байта полезного сообщения тоже меняются (7-10 байты).

Начальное условие поставим в 50.

Преобразуем байты (3-6) и (7-10) в два числа типа float, data1 и data2:

controller received message (1): "aa 0a c5 3d 49 42 a0 3f 76 00 69" t = 0 s
predicted checksum if: ff - (first 10 sum): 105 , real checksum: 105
corrupted messages: 0
data1 as float: 50.3103 data2 as float: 1.08594e-38

Как видно, в сообщении находятся 2 числа типа float, одно указывает текущее положение объекта по оси ординат (q), а второе число указывает референсный уровень.

Коммит:
https://github.com/Shamraev/CSP_2021_project_sem2/commit/51bb093511181cf4e7f28c3cb6727d76c6b8c500

3. Сохранение логов

Есть два способа создания логов в файл:

1. Бинарный файл
2. Текстовый файл

Бинарный файл занимает меньше места, его быстрее парсить, но сам парсинг будет слежнее парсинга текстового файла.

Будем использовать второй подход. Для этого используем QFile и QTextStream. В файл записываем 2 числа через пробел. Первое - положение q, второе - референсный уровень.

Пример:

20.3163 50
20.2169 50
19.4905 50
20.0087 50

Коммит:
https://github.com/Shamraev/CSP_2021_project_sem2/commit/ca433918da464479d7b06ad7af0d376ea63e44e4

4. Чтение сохраненного лога в Matlab

Используем функции *fileread*, *strsplit*, *str2double*.

На рисунке 2 изображен график изменения координаты объекта управления в Qt.

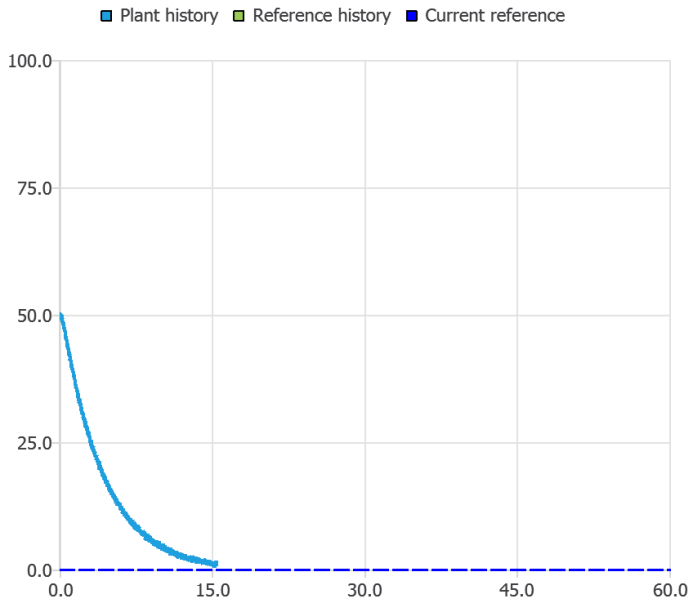

Рисунок 2 - График изменения координаты объекта в Qt

На рисунке 3 изображен график изменения координаты объекта управления, распарсенный в Matlab.

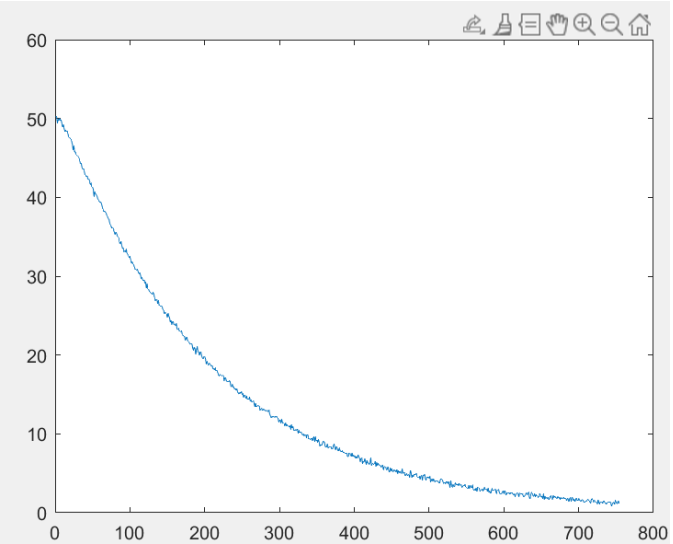

Рисунок 3 - График изменения координаты объекта в Matlab

Коммит:
https://github.com/Shamraev/CSP_2021_project_sem2/commit/d88a355ca2d6671e5673ccc64fc30a1645aa4939

5. Идентификация параметров объекта управления

5.1 Фильтрация

Используем фильтр низких частот. Результаты показаны на рисунках 5, 6 и 7, координаты q, скорости от q, ускорения от q соответственно.

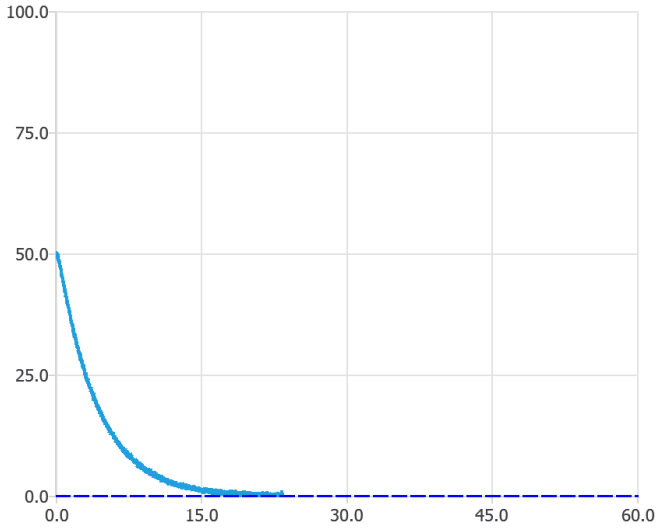

Рисунок 4 - График исходного сигнала q в Qt

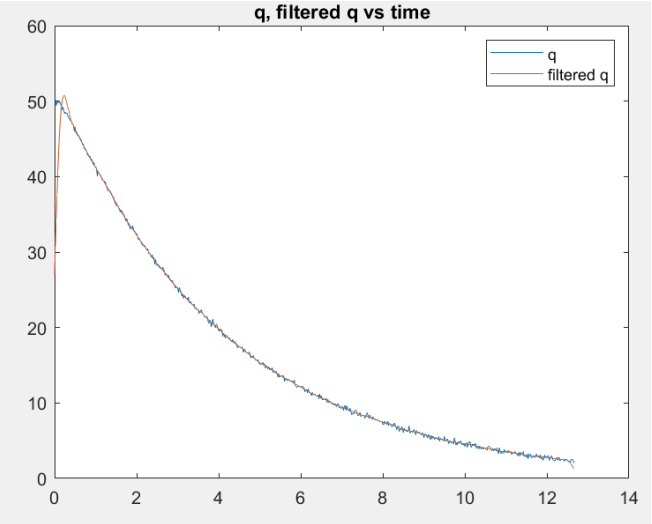

Рисунок 5 - График исходного и отфильтрованного сигналов q в Matlab с отн. частотой среза 5e-6

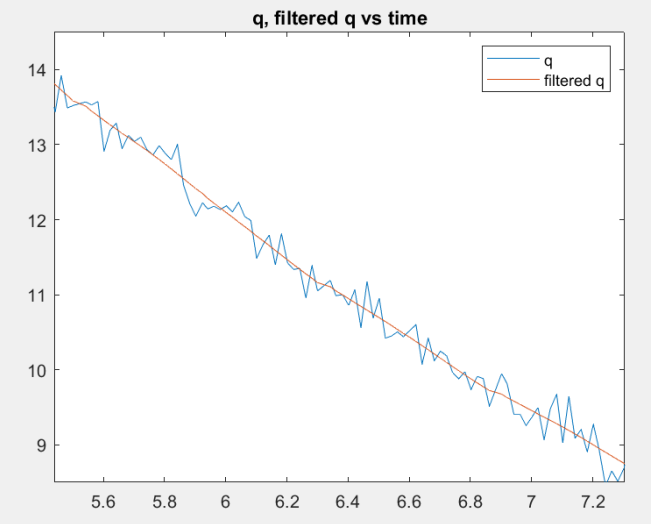

Рисунок 6 - График исходного и отфильтрованного сигналов q в Matlab с отн. частотой среза 5e-6 в увеличенном масштабе

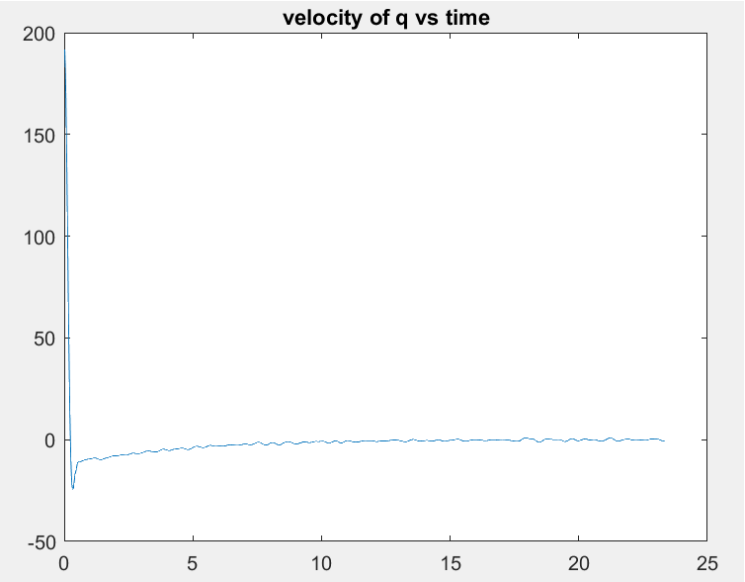

Рисунок 6 - График скорости отфильтрованной q в Matlab

Коммит:
https://github.com/Shamraev/CSP_2021_project_sem2/commit/48fe74dbb372bc17ee7a906d19c1a71cfa3636ce

5.2 Параметрическая идентификация

Наша система имеет вид:

$$W(s) = \frac{b_0}{s^2 + sa_1 + a_0};$$

Необходимо найти параметры a0, a1, b0.

Найдем эту систему в управляемой фробениусовой канонической форме пространства состояний:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases}$$

$$A = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} b_0 & 0 \end{bmatrix};$$

Тогда имеем уравнения:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -a_0 x_1 - a_1 x_2 + u; \\ y = b_0 x_1 \end{cases}$$

Известно : $y, \dot{y}, \ddot{y}, u$

$$\dot{y} = b_0 \dot{x}_1$$

$$x_1 = \frac{y}{b_0}$$

$$\dot{x}_1 = \frac{\dot{y}}{b_0} = x_2$$

подставив в $\dot{x}_2$ получим:

$$\dot{x}_2 = -\frac{a_0}{b_0} y - \frac{a_1}{b_0} \dot{y} + u;$$

$$\ddot{y} = -a_0 y - a_1 \dot{y} + b_0 u.$$

В матричном виде:

$$\ddot{y} = \begin{bmatrix} y & \dot{y} & u \end{bmatrix} \cdot \begin{bmatrix} -a_0 \\ -a_1 \\ b_0 \end{bmatrix}$$

где a_0, a_1 и b_0 - параметры системы, u - сигнал управления, y - переменная состояния (положение объекта по оси ординат).

$$\ddot{y} = [y\ \dot{y}\ u] \cdot \begin{bmatrix} -a_0 & -a_1 & b \end{bmatrix}^T = [y\ \dot{y}\ u] \cdot p,$$

где p=$\begin{bmatrix} -a_0 & -a_1 & b \end{bmatrix}^T$ - вектор неизвестных параметров, которые нужно найти.

Параметрическую идентификацию будем производить методом наименьших квадратов.

То есть задача такова:

$$p = [y\ \dot{y}\ u]^{\dagger} \cdot \ddot{y}$$

Чтобы **найти параметры p** проделаем следующие шаги:

1. Подадим управление на объект
2. В файл лога запишем y, u, t

3. Произведем фильтрацию y
4. Численно продифференцируем y фильтрованное - получим ẏ
5. Отфильтруем ẏ и численное продифференцируем - получим ÿ
6. Найдем $[y\ \dot{y}\ u]^{\dagger}$- псевдообратную матрицу к $[y\ \dot{y}\ u]$
7. Произведем операцию $p = [y\ \dot{y}\ u]^{\dagger} \cdot \ddot{y}$

В результате получили **p = [-0.4753 -0.7146 0.6664]=[-a0 -a1 b0].**
График показан на рисунке 7.

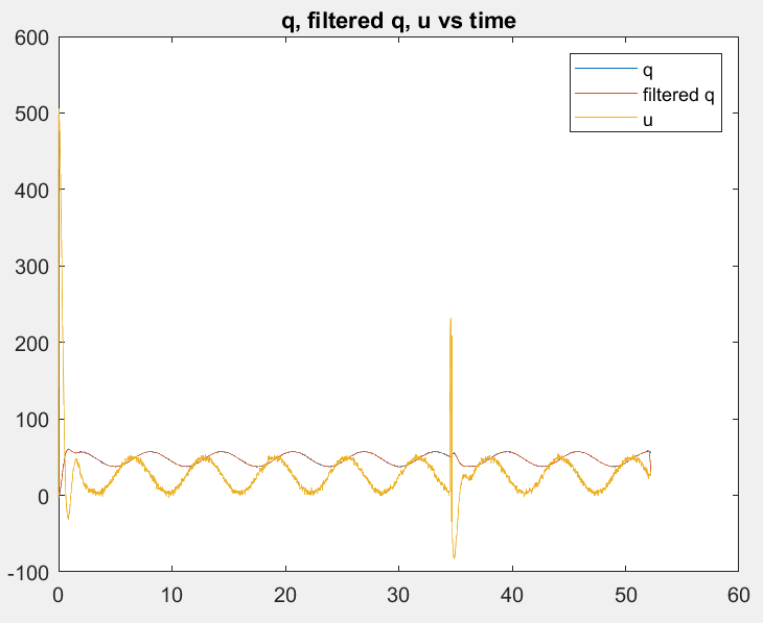

Рисунок 7 - График q, filtered q, u

Коммит:
https://github.com/Shamraev/CSP_2021_project_sem2/commit/77076a43d7ab1f52f84a0b2f2fdff475a5592e02

5.3 Индентификация с помощью матлаба

В видео https://youtu.be/qhIjIu-Zk10?t=428 показан процесс параметрической идентификации с помощью интрумента матлаба - System identification. На вход подаем управление u и изменяющуюся координату q. Желаемый сигнал был взят из пункта 7, который имеет гармоники, и в целом динамичен.

Найденная система:

$$W = \frac{2.099}{s^2+4.893s+1.161} \quad (3)$$

6. Настройка ПИД регулятора

Произведем настройку ПИД-регулятора для системы (3). Схема, синтезированная в Simulink, представлена на рисунке 8.

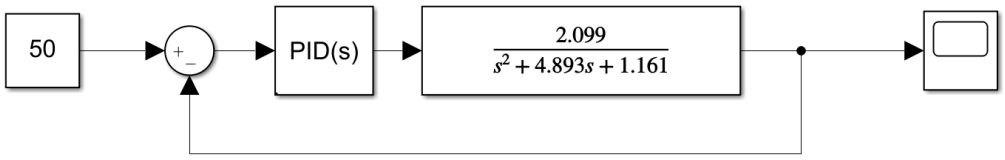

Рисунок 8 - Схема ПИД-регулятора

На схеме представлены единичный сигнал, блок PID Controller, передаточная функция с параметрами b0, a1, a0, полученными ранее.

Коэффициенты для ПИД-регулятора получены путем автонастройки в блоке PID Controller, коэффициенты P, I, и D соответственно: **[22.07 24.3 4.49].**

Результат представлен на рисунке 9.

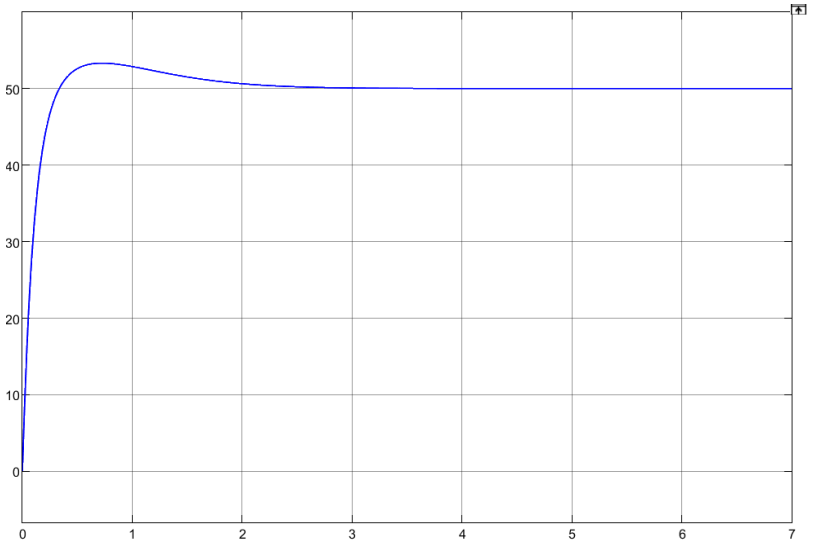

Рисунок 9 - Переходный процесс с регулятором

7. Реализация управления

7.1 Реализация фильтра низких частот на C++

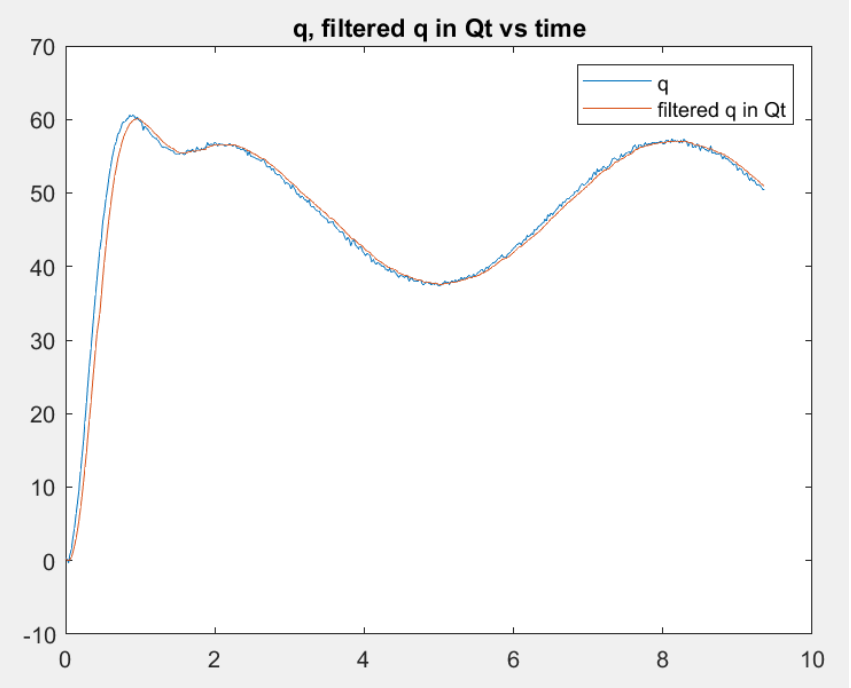

Рисунок 10 - Результат фильтрации в Qt

7.2 Реализация ПИД регулятора на C++

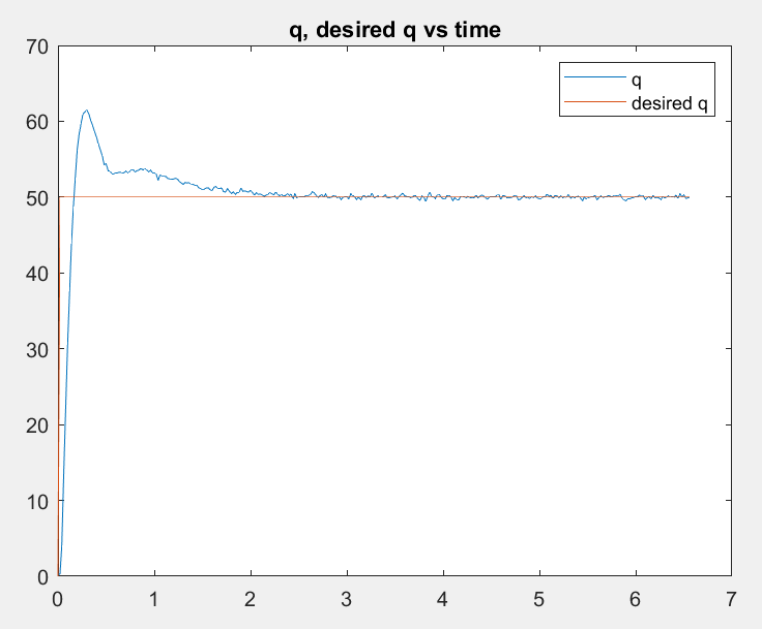

Рисунок 11 - Переходной процесс реальной системы

7.3 Слежение за задающим сигналом

Задающий сигнал:

$$70 - 10e^{-0.1t}\left(\frac{-(t-2)^2 + 60t}{100}\right)\sin\left(\frac{t}{2}\right)$$

Результат слежения показан на рисунке 12.

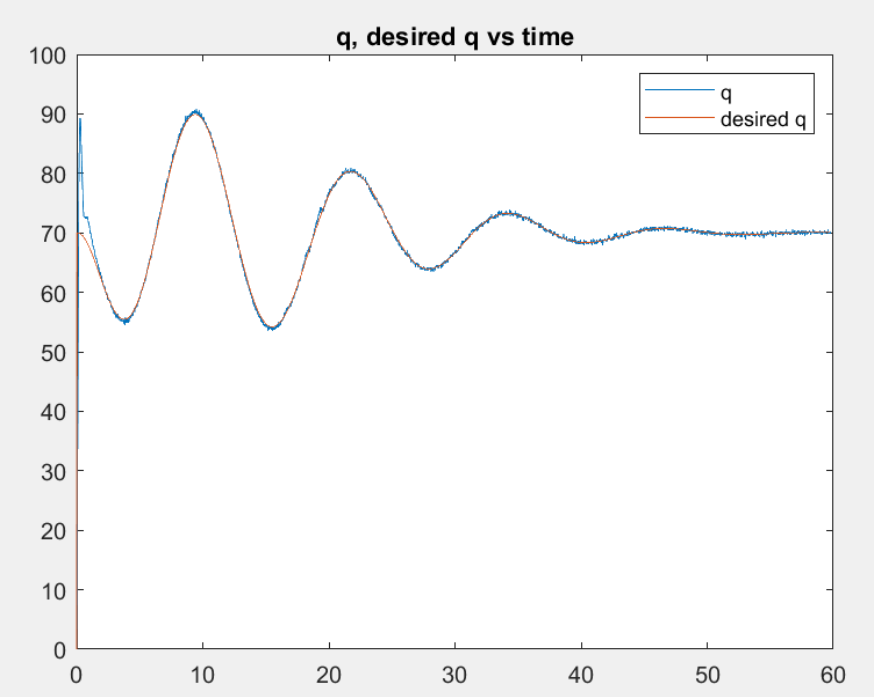

Рисунок 12 - Результат слежения за координатой q

Качество управления:

$$E = \sum_{i=0}^{n}(r[i] - y[i])^2,$$

где r – задающее воздействие, y – выход объекта управления.

Для получения усредненного значения применим формулу:

$$\overline{E} = \sqrt{\frac{E}{n}}.$$

Результат на 60 секундах:

$$\bar{\mathrm{E}} = 2.476$$

Листинг программы в Qt creator (а также все материалы по проекту) можно найти по ссылке: https://github.com/Shamraev/CSP_2021_project_sem2

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был разобран протокол передачи данных, реализованы классы контроллера, фильтра, ПИД контроллера.

Также довольно интересно было идентифицировать объект. Не совсем получилось с помощью псевдообращения, но очень просто с помощью матлабовской утилитой System Identification. Данные лучше всего использовать динамичного переходного процесса, с множеством гармоник.

Настроен ПИД регулятор по идентифицированной модели. Вывод: использовать и модель и регулятор в непрерывном режиме.

Качество слежения за координатой $\bar{E} = 2.476$на 60 секундах.

Сделаем вывод: огромную роль играет идентификация объекта для точной настройки регулятора на “компьютере”. Разносторонняя работа с задачей управления, приближенная к реальному миру (о системе известно мало, надо найти регулятор, протоколы и пр.).