

СЕМИНАР 6

Модель НПА 6DOF

Система управления курсом

Блок формирования сигналов на движительно-рулевой комплекс

Настройка системы управления

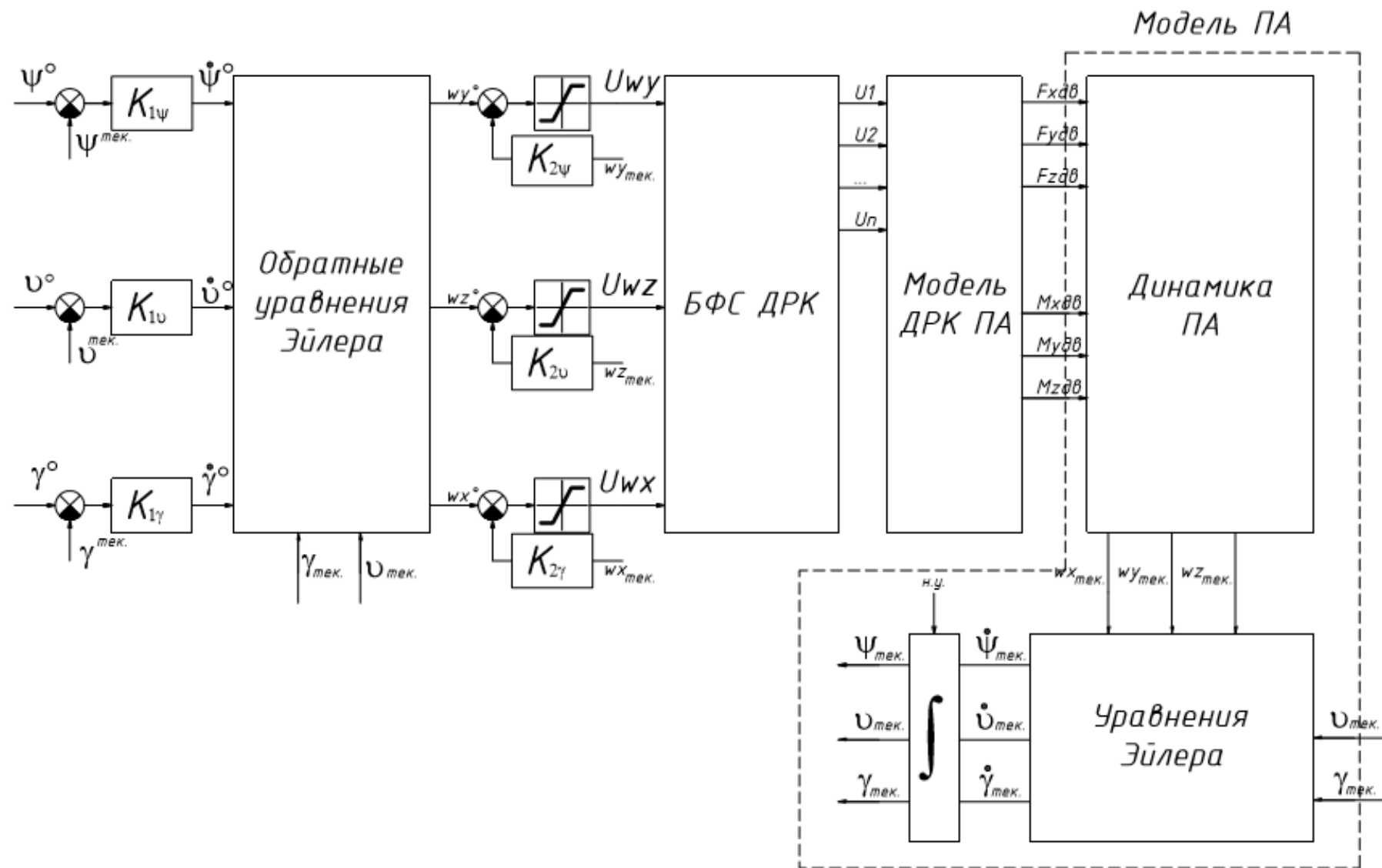
Зачем программировать модель? Зачем использовать kx-pult?

1. Возможность отладить алгоритмы заранее.
2. Лучшее понимание объекта управления
3. Возможность вывести и записать любые переменные программы
4. Возможность в ходе выполнения программы менять значения коэффициентов/переменных программы

Что будем делать?

- Вспоминать как выглядит математическая модель подводного аппарата:
 - Кинематика
 - Динамика
 - Модель винтомоторного агрегата
- Вспоминать как выглядит регулятор сепаратного канала управления (для примера канал курса)
 - Рисовать структурную схему
 - Программировать
- Разбираться с тем что такое блок формирования сигналов на движительно-рулевой комплекс (БФС ДРК)
 - Думать как получить уравнения БФС ДРК
 - Рисовать понятную схему
 - Программировать
- Проверять на практике (используя kx-pult 😊)

Общая схема системы управления ориентацией



Математическая модель НПА. Кинематика

$OX_gY_gZ_g$ - полусвязанная с НПА

$OXYZ$ — связанная с НПА

$OX^0Y^0Z^0$ — заданное положение НПА

$\psi \vartheta \gamma$ — текущие углы курса, дифферента и крена

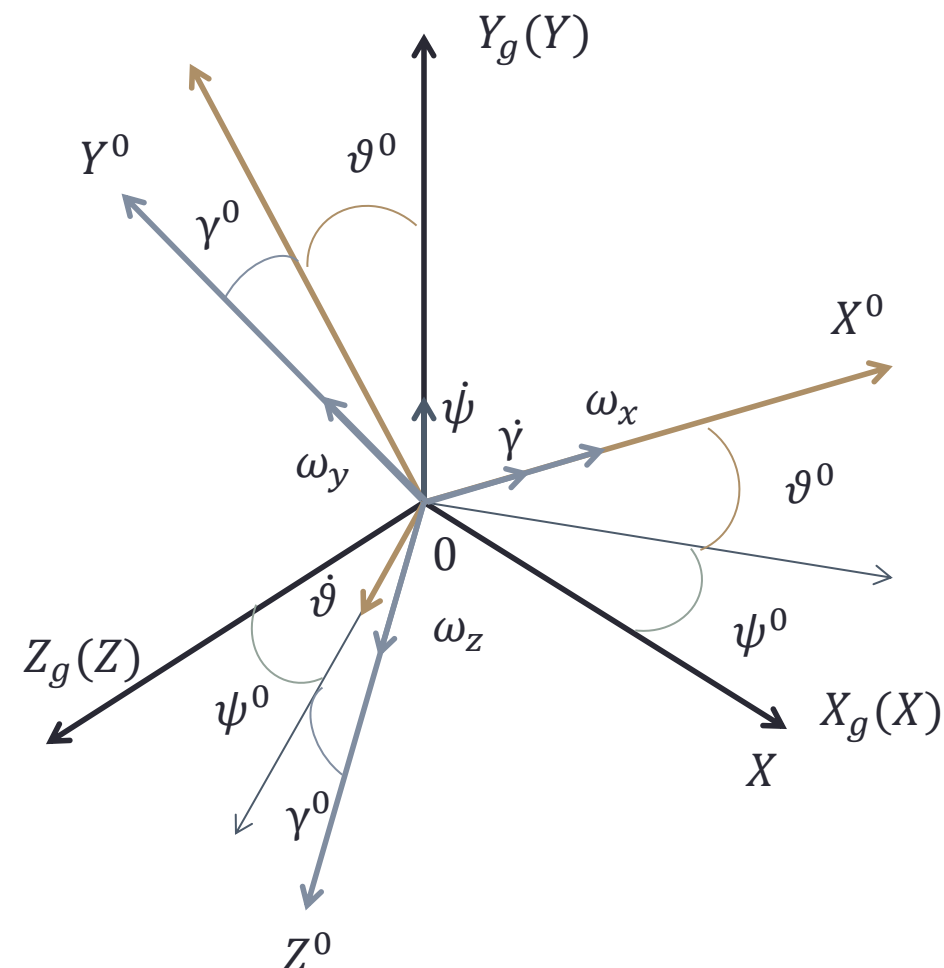
$\psi^0 \vartheta^0 \gamma^0$ - заданные углы курса, дифферента и крена

Кинематические уравнения для углов Эйлера:

$$\dot{\psi} = \frac{1}{\cos(\vartheta)} [\omega_y \cos(\gamma) - \omega_z \sin(\gamma)],$$

$$\dot{\vartheta} = \omega_y \sin(\gamma) + \omega_z \cos(\gamma),$$

$$\dot{\gamma} = \omega_x - \operatorname{tg}(\vartheta) [\omega_y \cos(\gamma) - \omega_z \sin(\gamma)],$$



Математическая модель НПА. Динамика

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = M_{дв}$$

M — матрица массо-инерционных характеристик

НПА:

$$M = \begin{bmatrix} I_x + \lambda_{44} & 0 & 0 \\ 0 & I_y + \lambda_{55} & 0 \\ 0 & 0 & I_z + \lambda_{66} \end{bmatrix}$$

Вектор моментов ДРК:

$$\tau = \begin{bmatrix} M_{двx} \\ M_{дву} \\ M_{двz} \end{bmatrix}$$

Вектор угловых скоростей НПА в СК $Oxyz$:

$$v = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

$C(v)$ — матрица сил и моментов инерции НПА:

$$C(v) = \begin{bmatrix} 0 & -(I_z + \lambda_{66})\omega_z & (\lambda_{55} + I_y)\omega_y \\ (\lambda_{66} + I_z)\omega_z & 0 & -(I_x + \lambda_{44})\omega_x \\ -(I_y + \lambda_{55})\omega_y & (\lambda_{44} + I_x)\omega_x & 0 \end{bmatrix}$$

$M_{дв}$ — вектор управляющих моментов ДРК

Матрица демпфирующих гидродинамических сил:

$$D(v) = \begin{bmatrix} -(C_{\omega_{x1}} + C_{\omega_{x2}}|\omega_x|) & 0 & 0 \\ 0 & -(C_{\omega_{y1}} + C_{\omega_{y2}}|\omega_y|) & 0 \\ 0 & 0 & -(C_{\omega_{z1}} + C_{\omega_{z2}}|\omega_z|) \end{bmatrix}$$

Математическая модель ВМА

Линеаризованная модель ВМА:

$$W_{\text{дв}j}(p) = \frac{F_j(p)}{U_j(p)} = \frac{K_{\text{дв}j}}{(T_1 p + 1)}$$

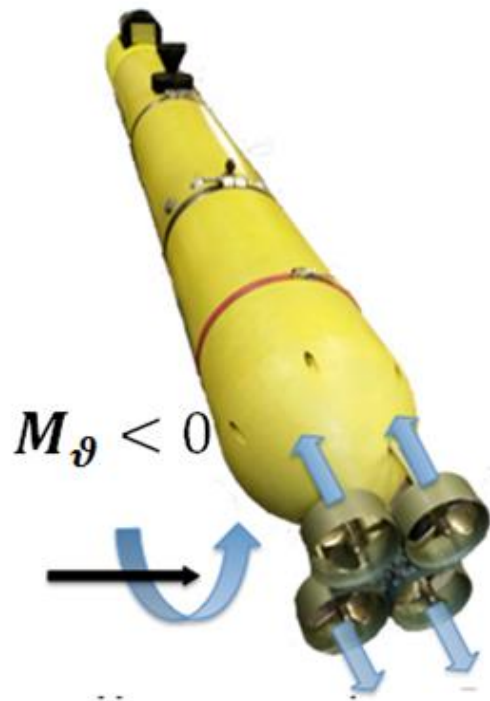
Модель в пространстве состояний:

$$\frac{dF_j}{dt} = \frac{1}{T_1} (U_j K_{\text{дв}j} - F_j)$$

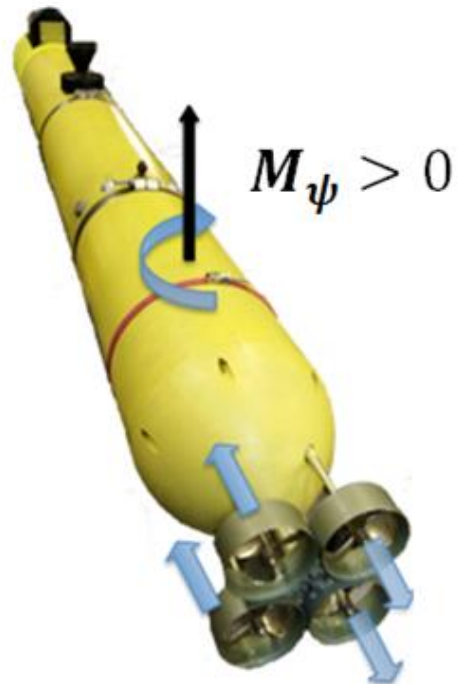
Хм, отлично..это модель двигателя, но в уравнениях динамики был вектор $M_{\text{дв}x,y,z}$ Как получить его?

Формирование сил и моментов ДРК

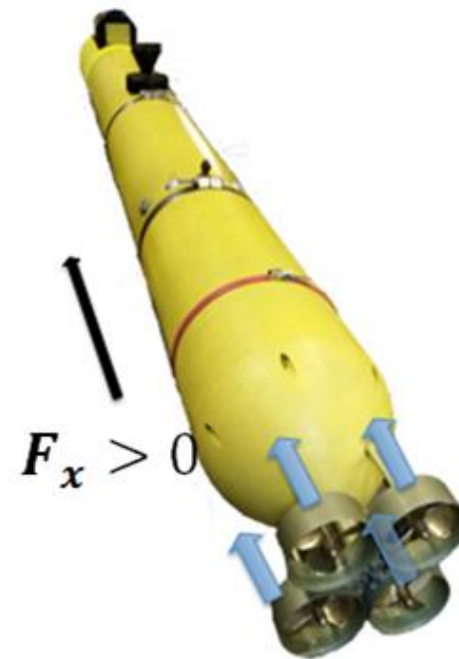
Дифферент



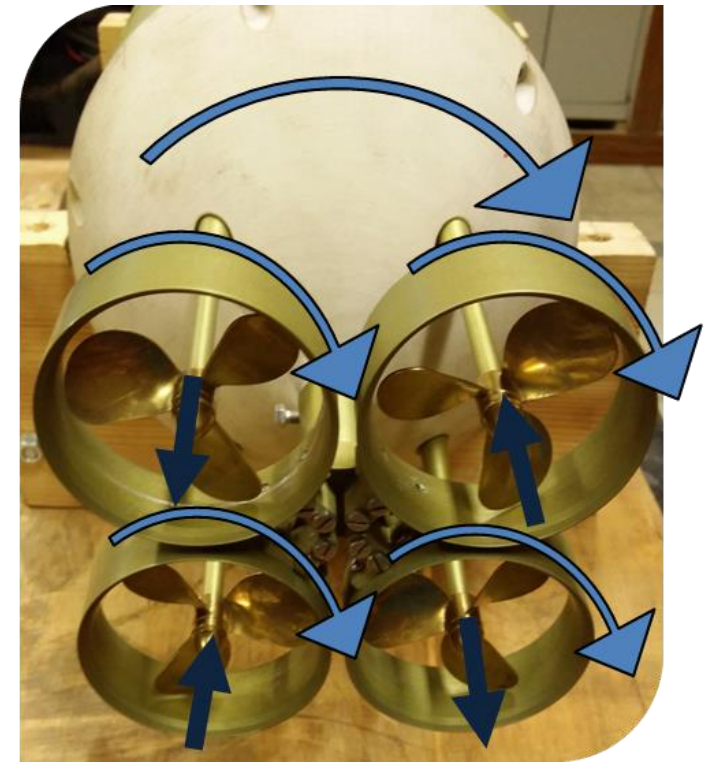
Курс



Марш



Крен

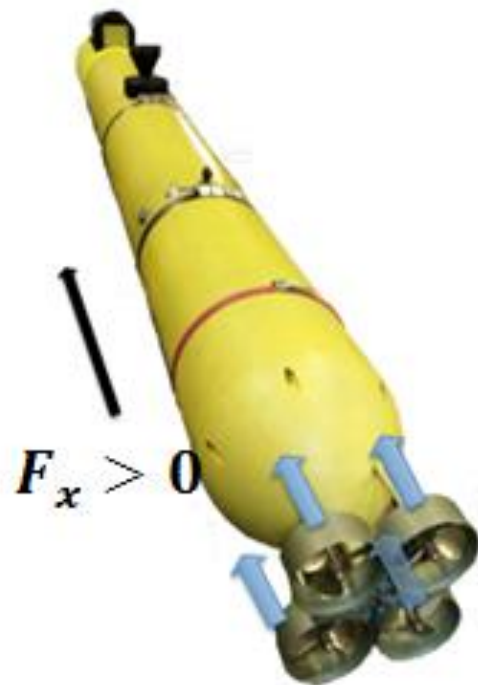


Поступательное движение.

$$F_{dx} = P_{mvp_x} + P_{mvl_x} + P_{mnp_x} + P_{mnl_x};$$

$$F_{dy} = 0;$$

$$F_{dz} = 0;$$



P_{mvp_x} – тяга маршевого вертикального правого ВМА

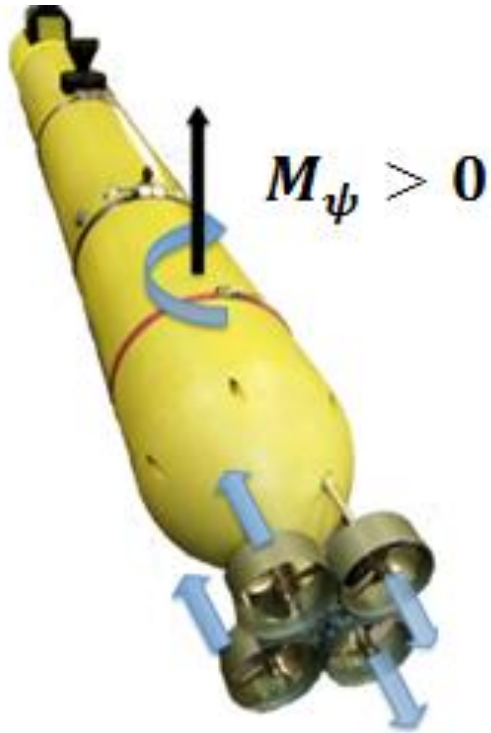
P_{mvl_x} – проекция тяги маршевого вертикального левого ВМА на продольную ось АНПА

P_{mnp_x} – проекция тяги маршевого нижнего правого ВМА

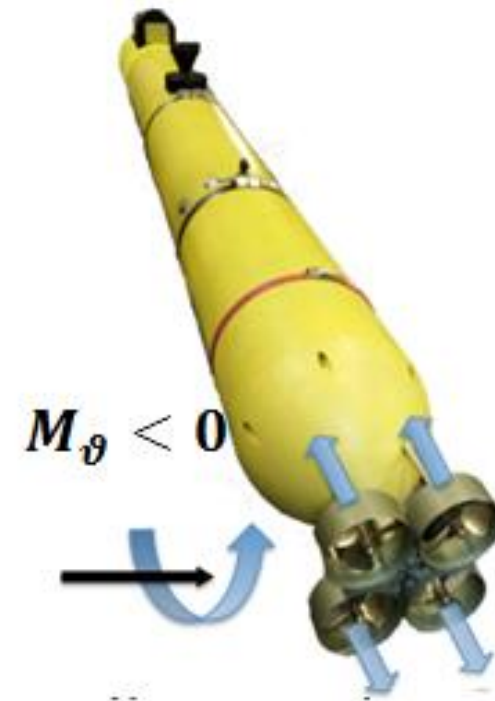
P_{mnl_x} – проекция маршевого нижнего левого ВМА

Вращательное движение

$$M_{dy} = l_2 * (-P_{mvp_x} + P_{mvl_x} + P_{mnl_x} - P_{mnp_x}) ;$$

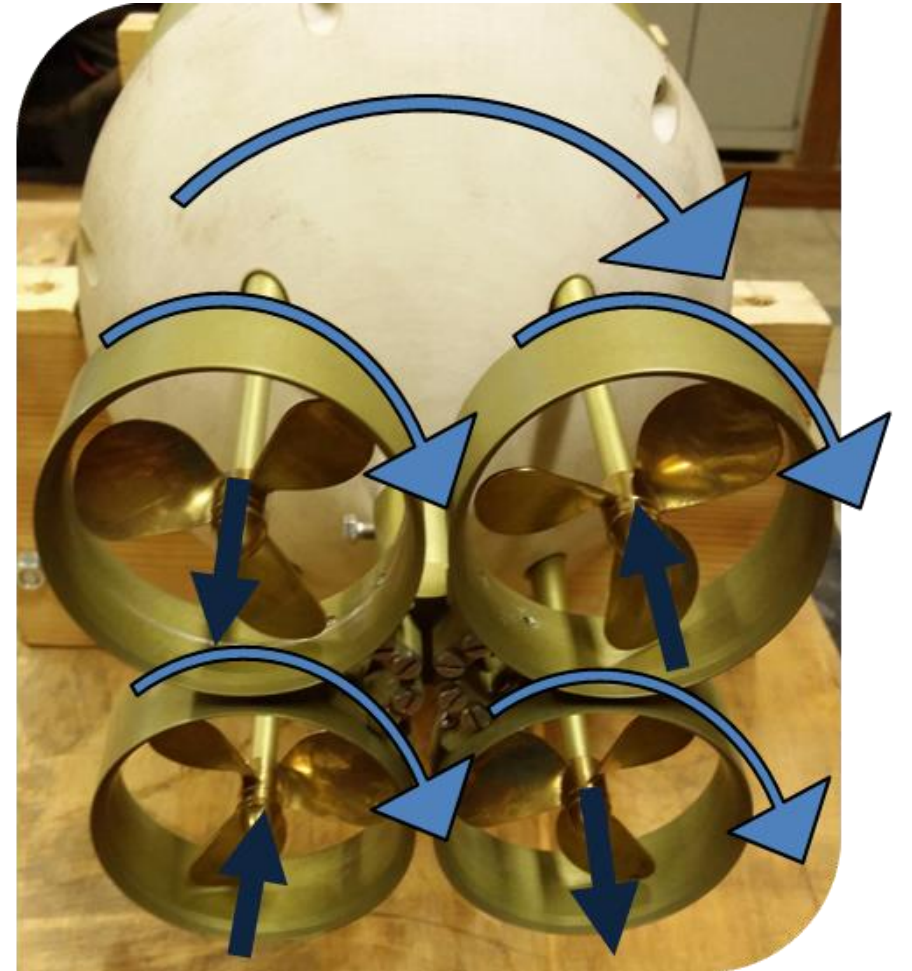


$$M_{dz} = l_1 * (-P_{mvp_x} - P_{mvl_x} + P_{mnl_x} + P_{mnp_x}) ;$$



Модель ТНПА. ROV_Model. Вращательное движение.

$$M_{dx} = k_{\gamma} * (P_{mvp_x} + P_{mnl_x} - P_{mnp_x} - P_{mvl_x}) ;$$



Класс модели ROV_Model

Методы:

`model`(управляющие сигналы на движители) – метод, который рассчитывает производные для всех параметров состояния модели

`runge`(управляющие сигналы на движители, шаг) – метод, который выполняет интегрирование параметров состояния математической модели

Класс модели ROV_Model

```
runge (U1, U2,...,h) {  
    a1 = xn; - инициализация  
    model (U1, U2, ...); (k1 = da)  
    a = a1 +  $\frac{h}{2}k_1$   
    model (U1,U2,...); (k2=da)  
    a = a1 +  $\frac{h}{2}k_2$   
    model (U1,U2,...); (k3=da)  
    a = a1 +  $hk_3$   
    model (U1,U2,...); (k4=da)  
     $x_{n+1} = a1 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$   
}
```

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$k_1 = f(x_n, u),$$
$$k_2 = f\left(x_n + \frac{h}{2}k_1, u\right),$$
$$k_3 = f\left(x_n + \frac{h}{2}k_2, u\right),$$
$$k_4 = f(x_n + hk_3, u),$$

где x_{n+1} - значение переменной состояния на следующем шаге

x_n - текущее значение переменной состояния

h - шаг интегрирования (dt)

u – управляющий сигнал

Работа с классом ROV_Model

Создание объекта
класса ROV_Model;

Запуск таймера

Вызов метода
ROV_Model::runge()
по тикку таймера

Входными параметрами для функции runge() являются заданные напряжения на каждый ВМА вашего аппарата и шаг интегрирования.

ROV_Model. Runge()

`void ROV_Model::runge(const float Umvl, const float Umnl, const float Umvp, const float Umnp, const float Ttimer)`

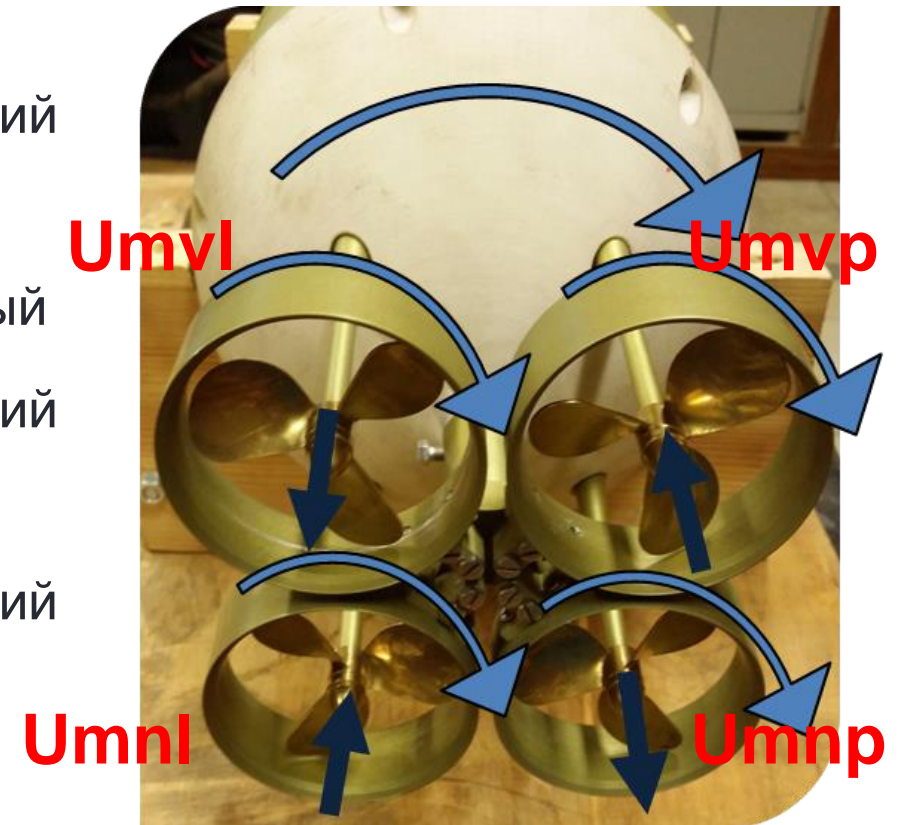
Umvl – заданное напряжение на ВМА маршевый верхний левый

Umnl – заданное напряжение на ВМА маршевый нижний левый

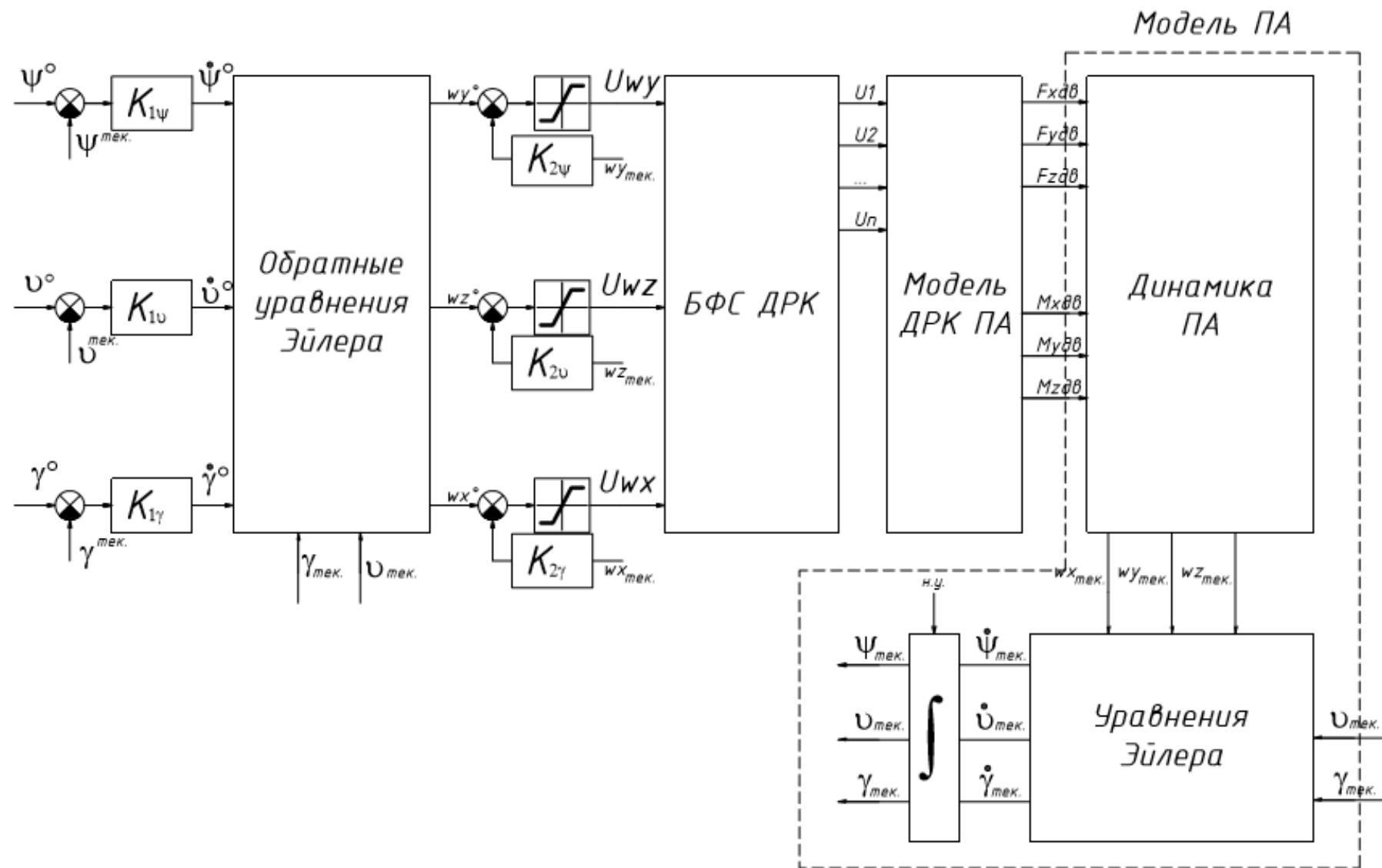
Umvp - заданное напряжение на ВМА маршевый верхний правый

Umnp - заданное напряжение на ВМА маршевый нижний правый

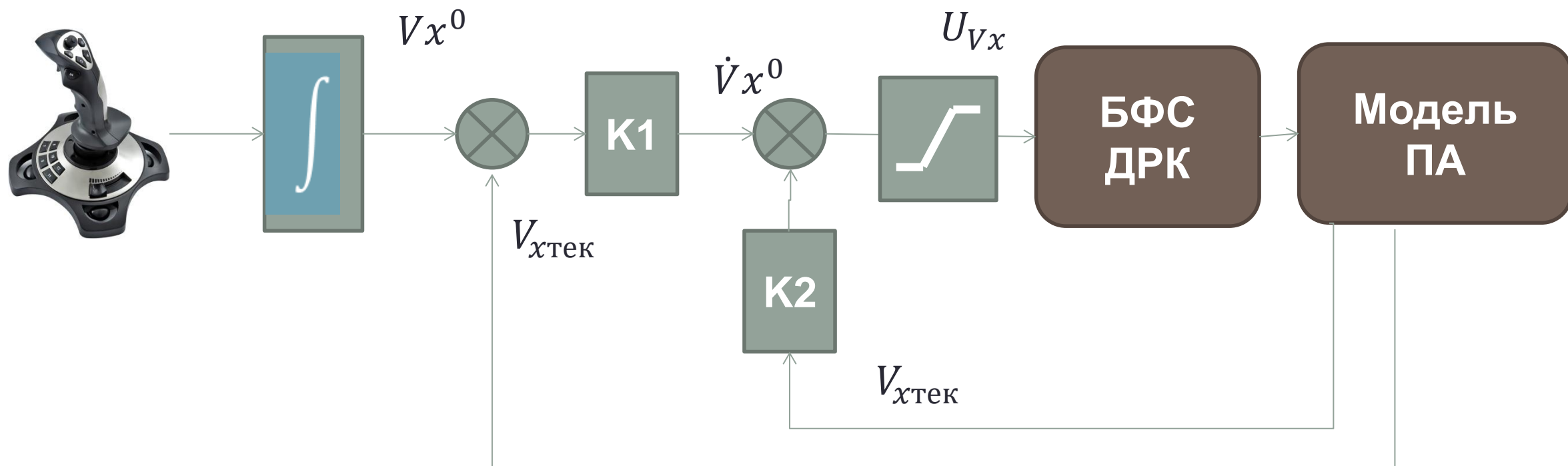
Ttimer – период запуска таймера в с



Общая схема системы управления ориентацией



Общая структура контура системы управления



БФС ДРК – блок формирования сигналов на движительно-рулевой комплекс. БФС ДРК распределяет сигналы с контуров управления на отдельные ВМА НПА.

Этапы работы над системой управления

- Составление математической модели подводного аппарата
- Программирование математической модели
- Выбор закона управления (включая аналитическую проверку)
-
- Получение уравнений блока формирования сигналов на движительно-рулевой комплекс
- Составление схемы контура управления
- Программирование и проверка БВС ДРК
- Программирование и проверка системы управления
- Настройка системы управления

Уравнения БФС ДРК

Можно получить из уравнений модели:

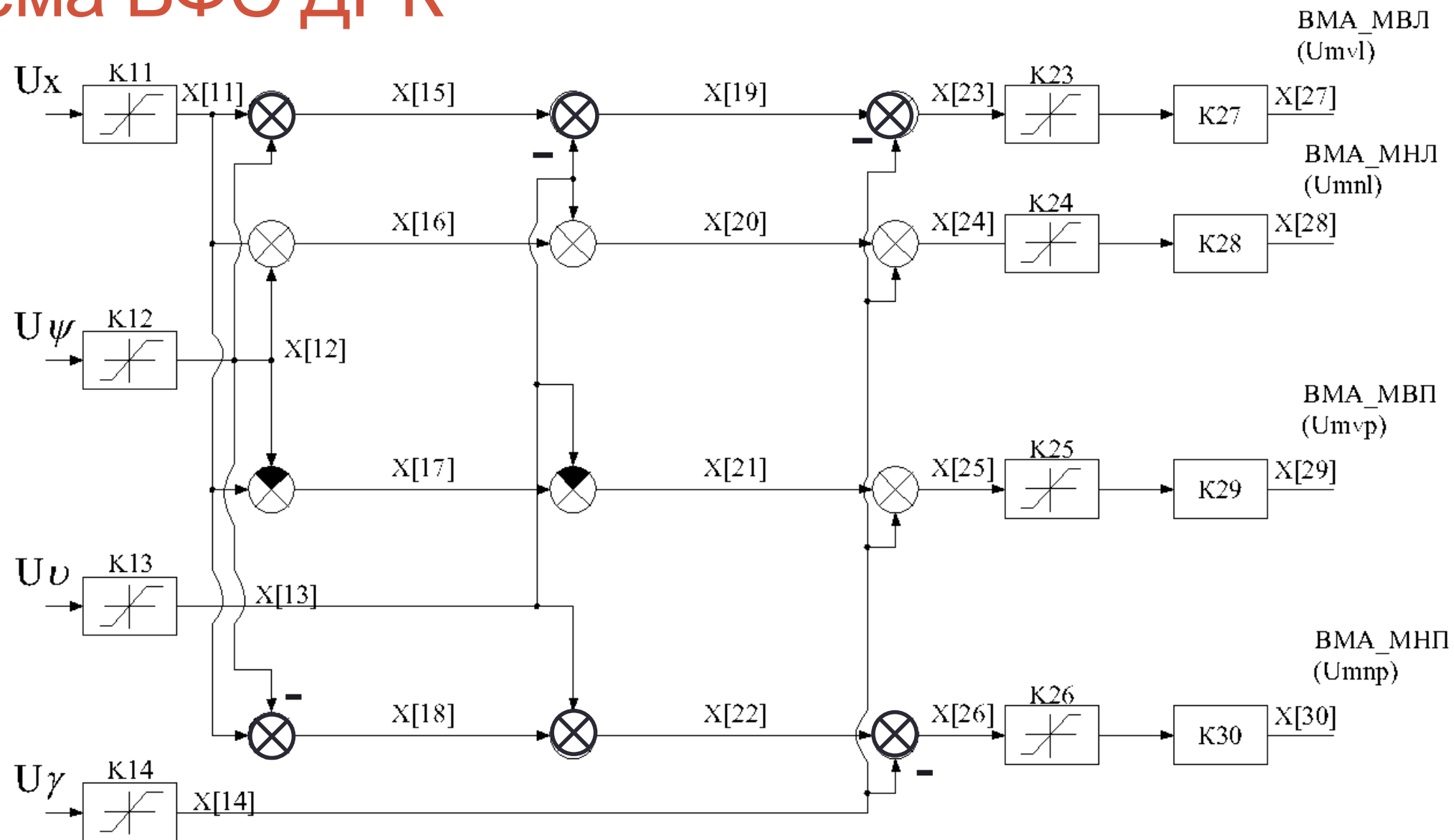
$$U_{mvl} = -U_{\vartheta} + U_{\psi} + U_x - U_{\gamma}$$

$$U_{mnl} = U_{\vartheta} + U_{\psi} + U_x + U_{\gamma}$$

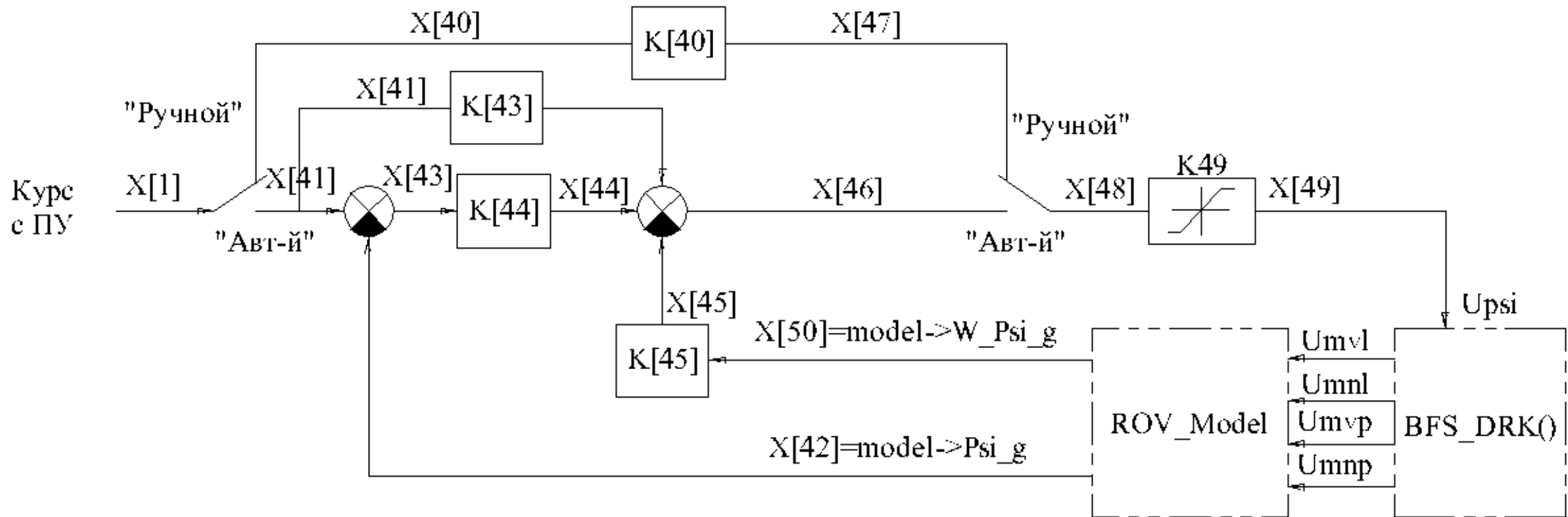
$$U_{mvp} = -U_{\vartheta} - U_{\psi} + U_x + U_{\gamma}$$

$$U_{mnp} = U_{\vartheta} - U_{\psi} + U_x - U_{\gamma}$$

Схема БФС ДРК



Контуры СУ. Курс. SU_ROV::Control_Kurs()



Практическая часть

1. Настроить контур курса
2. Самостоятельно написать и настроить контур управления дифференциалом

Практическая часть

Структура проекта:

main.cpp	
configdata.cpp, configdata.h	
kx_protocol.cpp, kx_protocol.h	
qkx_coeffs.cpp, qkx_coeffs.h	
qpiconfig.cpp, qpiconfig.cpp	
su_rov.cpp, su_rov.h	Класс, в котором реализована СУ, БФС ДРК, происходит запуск таймера и математической модели НПА
rov_model.cpp, rov_model.h	Математическая модель НПА

Этапы выполнения практической части

- Настройка и подключение kx-pult
- Программирование БФС ДРК
- Программирование контура курса
- Настройка контура курса

Подключение kx-pult.

```
1  #ifndef SU_ROV_H
2  #define SU_ROV_H
3
4  #include <QObject>
5  #include "kx_protocol.h"
6  #include "rov_model.h"
7
8  extern double X[2000][2];
9  extern QVector<double> K;
10
11 class SU_ROV : public QObject
12 {
13     Q_OBJECT
14 public:
15     explicit SU_ROV(QObject *parent = nullptr);
16
17 public slots:
18     void tick();
19 private:
20     Qkx_coeffs *K_Protocol;
21     x_protocol *X_Protocol;
22     ROV_Model *model;
23     QTimer time;
24     float T; // период таймера
25 };
26
27 #endif // SU_ROV_H
28
```

```
1  #include "su_rov.h"
2
3  SU_ROV::SU_ROV(QObject *parent) : QObject(parent)
4  {
5      K_Protocol = new Qkx_coeffs("protocols.conf", "ki");
6      X_Protocol = new x_protocol("protocols.conf", "xi", X);
7      model = new ROV_Model();
8      T = 0.01;
9      time.start(T * 1000); // запуск проводим в мсек
10     connect(&time, SIGNAL(timeout()), SLOT(tick()));
11
12 }
13
14
15 void SU_ROV::tick()
16 {
17     X[4][0] = K[1]; // проверка работы kx-pult
18
19
20 }
```

Проверка наличия связи с kx-pult

Noname - KX Pult

Coefficients (K)	Commands (C)	Graphics (X)	Configuration
<p>K</p> <p>receiver: 127.0.0.1:13043 - Opened</p> <p>sender: 127.0.0.1:13042</p> <p>type: 0xBB</p> <p>address K: 0x1B</p> <p>address pult: 0x1A</p> <p>sended count: <input type="text" value="0"/></p> <p>received count: <input type="text" value="0"/></p> <p>wrong received count: <input type="text" value="0"/></p> <p>missed received count: <input type="text" value="0"/></p>		<p>X</p> <p>receiver: 127.0.0.1:13040 - Opened</p> <p>sender: 127.0.0.1:13041</p> <p>type: 0xAA</p> <p>address X: 0x0A</p> <p>address pult: 0x0B</p> <p>sended count: <input type="text" value="1109"/></p> <p>received count: <input type="text" value="178"/></p> <p>wrong received count: <input type="text" value="0"/></p> <p>missed received count: <input type="text" value="0"/></p>	

Тестируете передачу коэффициентов

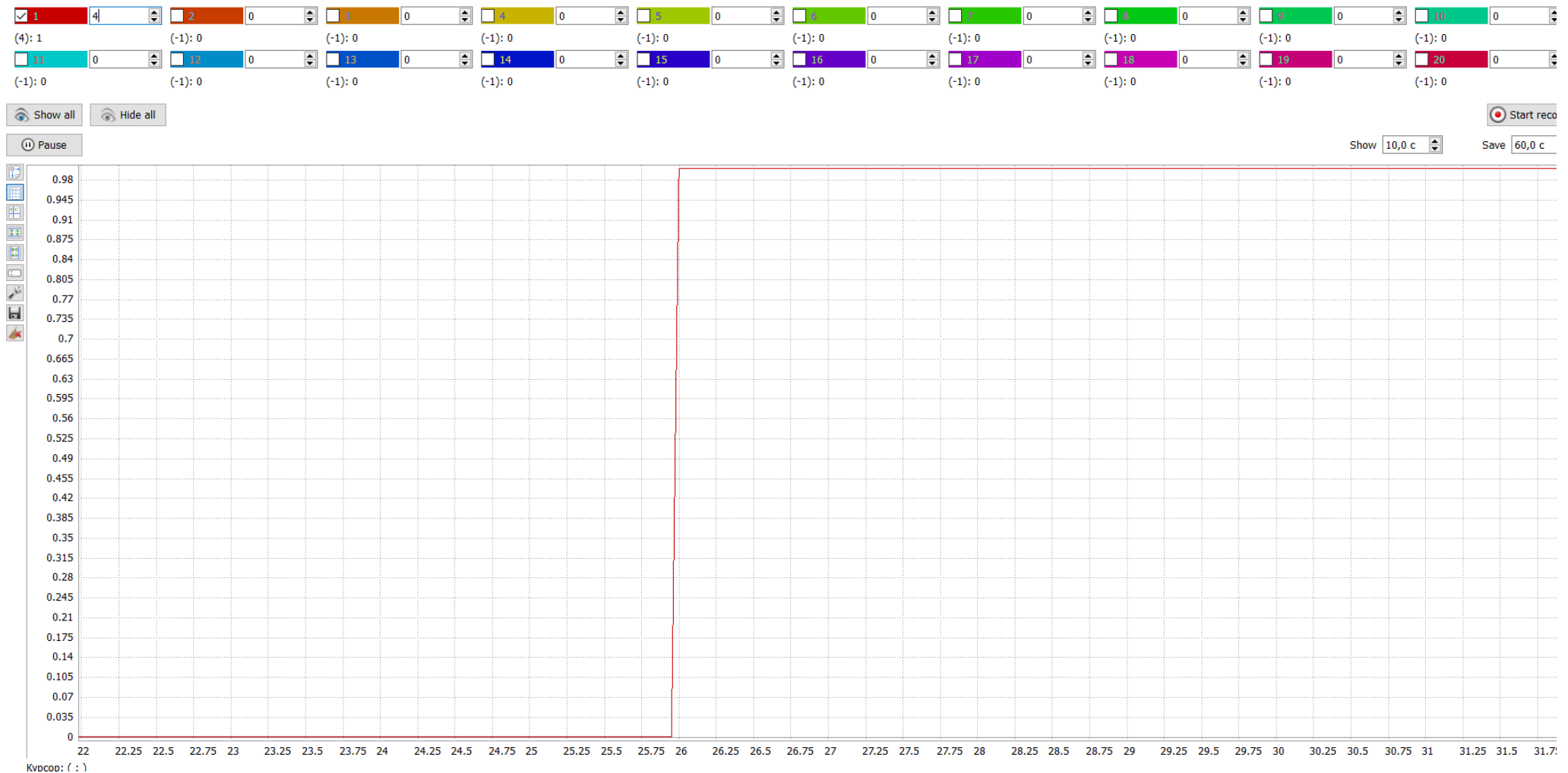
15/04/2020 19:27 - Update settings from "kx_pult.conf"
15/04/2020 19:27 - Read K file "k.dat": 1000 coeffs, 43428 bytes
15/04/2020 19:27 - Update descriptions from "k_description.h"
15/04/2020 19:29 - Update descriptions from "k_description.h"
15/04/2020 19:51 - Write K file "k.dat": 1000 coeffs, 43428 bytes
15/04/2020 19:52 - K send

Index	Name	Expression	Calculated	Type	Comment
0		1	1		
1		1	1		
2		0	0		
3		0	0		
4		0.00000000	0		
5		0.00000000	0		
6		0	0		
7		0.00000000	0		
8		0.00000000	0		

В нашем проекте при проверке $X[4][0] = K[1]$. Чтобы проверить работоспособность механизма в нашем проекте зададим ненулевое значение K_1

Для передачи коэффициентов нажимаем Send и смотрим, что появилось сообщение о том, что коэффициенты отправлены.

Смотрим изменение значения X[4][0]



Общая идея работы с классом SU_ROV

- К классу подключен механизм коэффициентов и х-ов. Программирование системы управления происходит в использовании этих переменных.
- В классе есть таймер. По тику таймера вызывается слот, в котором происходит основная работа:
 - На основе текущего состояния модели (углов ориентации, скоростей, координат и т.п.) регуляторы контуров рассчитывают управляющие сигналы.
 - Управляющие сигналы из контуров передаются в БФС ДРК и формируют управляющие сигналы на отдельные движители НПА
 - Управляющие сигналы движителей подаются на вход метода интегрирования уравнений математической модели.



Общая идея работы с классом SU_ROV

Su_rov.cpp

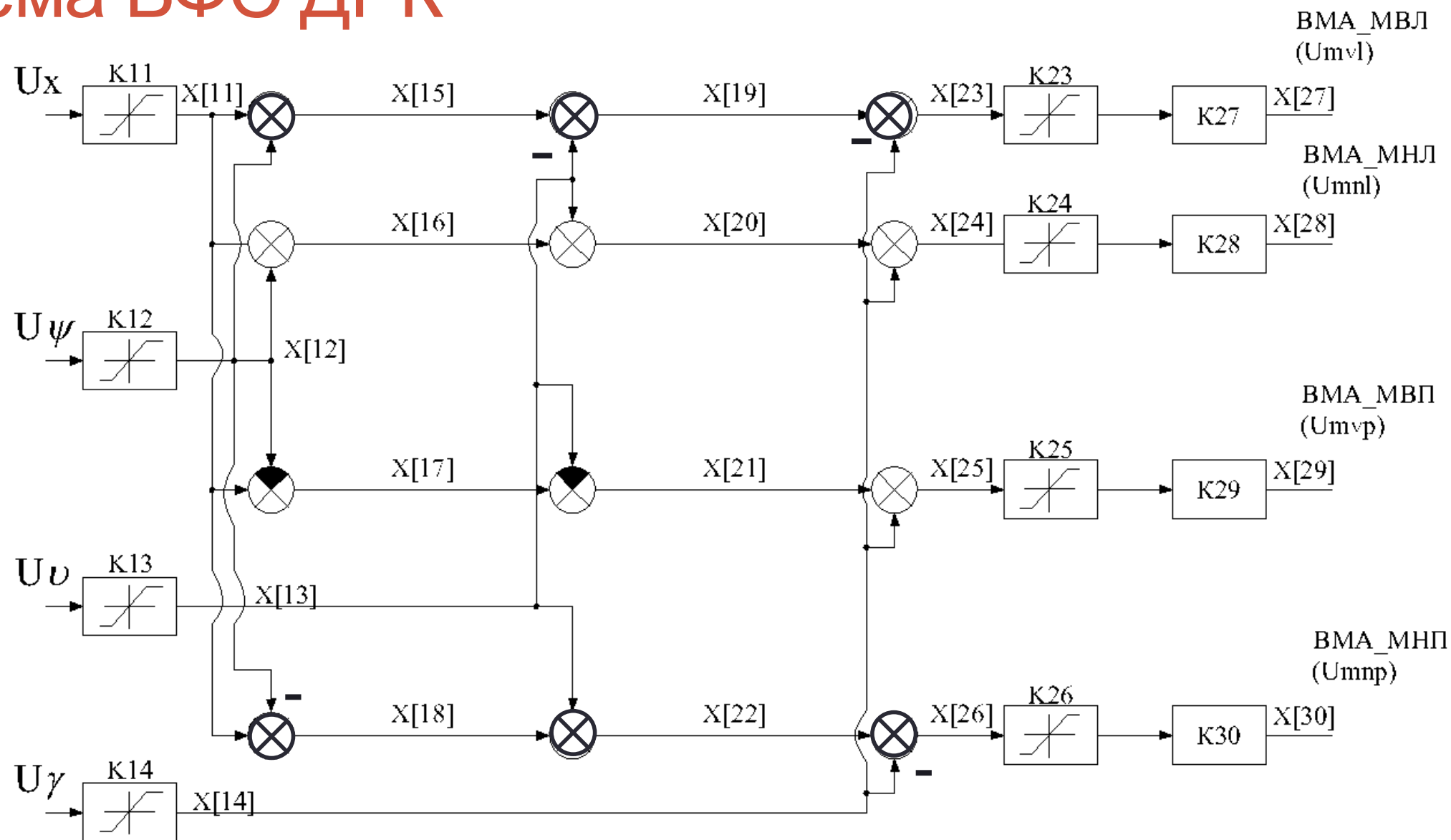
```
void SU_ROV::tick()
{
    ... X[1][0]=K[1]; //проверка работы kx-pult
    ... //метод, который записывает параметры модели в X-ы
    ... getDataFromModel();
    ... //метод, в котором реализован контур курса
    ... yawControlChannel();
    ... //блок формирования сигналов на движительно-рулевой комплекс
    ... // BFS_DRK(Upsi, Uteta, Ugamma, Ux);
    ... BFS_DRK(X[49][0], 0,0,0);
    ... //математическая модель АНПА
    ... //runge( Umv1, Umn1, Umvp, Umnp, dt)
    ... model->runge(X[27][0],X[28][0],X[29][0],X[30][0],0.01);
}
```

tick() – метод, который вызывается по таймеру

Запись данных с модели в X-ы

```
14
15 ▼ void SU_ROV::getDataFromModel() {
16     ... X[32][0] = model->Fx;
17     ... X[33][0] = model->vx_global;
18     ... X[34][0] = model->vx_local;
19     ... X[35][0] = model->x_global;
20     ... X[36][0] = model->y_global;
21     ... X[37][0] = model->z_global;
22     ... X[39][0] = model->Tetta_g;
23     ... X[40][0] = model->Gamma_g;
24     ... X[42][0] = model->Psi_g; //курс
25     ... X[50][0] = model->W_Psi_g; //угловая скорость по курсу
26 }
27
```

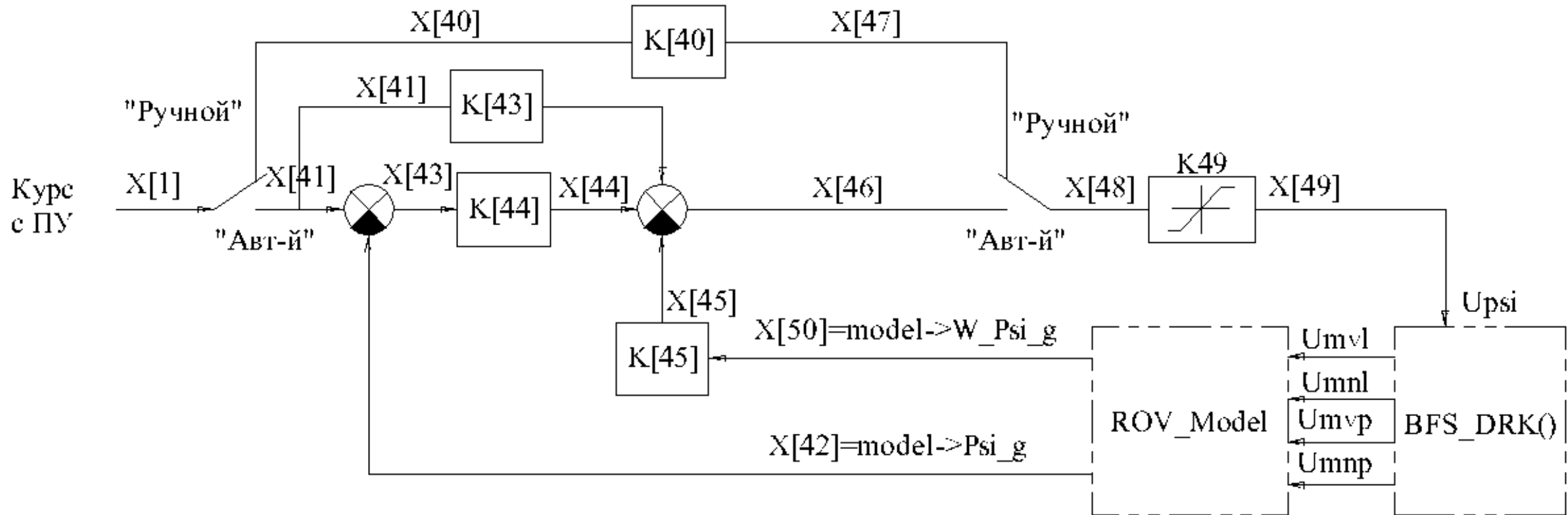
Схема БФС ДРК



БФС ДРК. SU_ROV::BFS_DRK()

```
57 void SU_ROV::BFS_DRK(double Upsi, double Uteta, double Ugamma, double Ux){
58     //ограничим входные задающие сигналы в бфс ДРК
59     X[11][0] = saturation(Ux, K[11]);
60     X[12][0] = saturation(Upsi, K[12]);
61     X[13][0] = saturation(Uteta, K[13]);
62     X[14][0] = saturation(Ugamma, K[14]);
63
64     //далее по структурной схеме БФС, вычисляем значения после первого сумматора
65     X[15][0] = X[11][0] + X[12][0]; //промежуточное значение для ВМА МВЛ (управление курсом и маршем)
66     X[16][0] = X[11][0] + X[12][0]; //промежуточное значение для ВМА МНЛ (управление курсом и маршем)
67     X[17][0] = X[11][0] - X[12][0]; //промежуточное значение для ВМА МВП (управление курсом и маршем)
68     X[18][0] = X[11][0] - X[12][0]; //промежуточное значение для ВМА МНП (управление курсом и маршем)
69
70     //далее по структурной схеме БФС, вычисляем значения после второго сумматора
71     X[19][0] = X[15][0] - X[13][0];
72     X[20][0] = X[16][0] + X[13][0];
73     X[21][0] = X[17][0] - X[13][0];
74     X[22][0] = X[18][0] + X[13][0];
75
76     //далее по структурной схеме БФС, вычисляем значения после третьего сумматора
77     X[23][0] = X[19][0] - X[14][0];
78     X[24][0] = X[20][0] + X[14][0];
79     X[25][0] = X[21][0] + X[14][0];
80     X[26][0] = X[22][0] - X[14][0];
81
82     //ограничим и промасштабируем управляющие значения напряжений для ВМА
83     X[27][0] = saturation(X[23][0], K[23]) * K[27]; //управляющее напряжение на ВМА МВЛ
84     X[28][0] = saturation(X[24][0], K[24]) * K[28]; //управляющее напряжение на ВМА МНЛ
85     X[29][0] = saturation(X[25][0], K[25]) * K[29]; //управляющее напряжение на ВМА МВП
86     X[30][0] = saturation(X[26][0], K[26]) * K[30]; //управляющее напряжение на ВМА МНП
87 }
```

Контуры СУ. Курс. SU_ROV::Control_Kurs()



Регулятор канала управления курсом

```
27  
28 ▼ void SU_ROV::yawControlChannel()  
29 {  
30     ... X[41][0]=X[1][0];  
31     ... X[43][0]=X[41][0]-X[42][0];  
32     ... X[44][0]=X[43][0]*K[44];  
33  
34     ... X[45][0]=X[50][0]*K[45];  
35     ... X[46][0]=X[44][0]-X[45][0]+X[41][0]*K[43];  
36     ... X[48][0]=X[46][0];  
37     ... X[49][0]=saturation(X[48][0],K[49]);  
38 }  
39
```

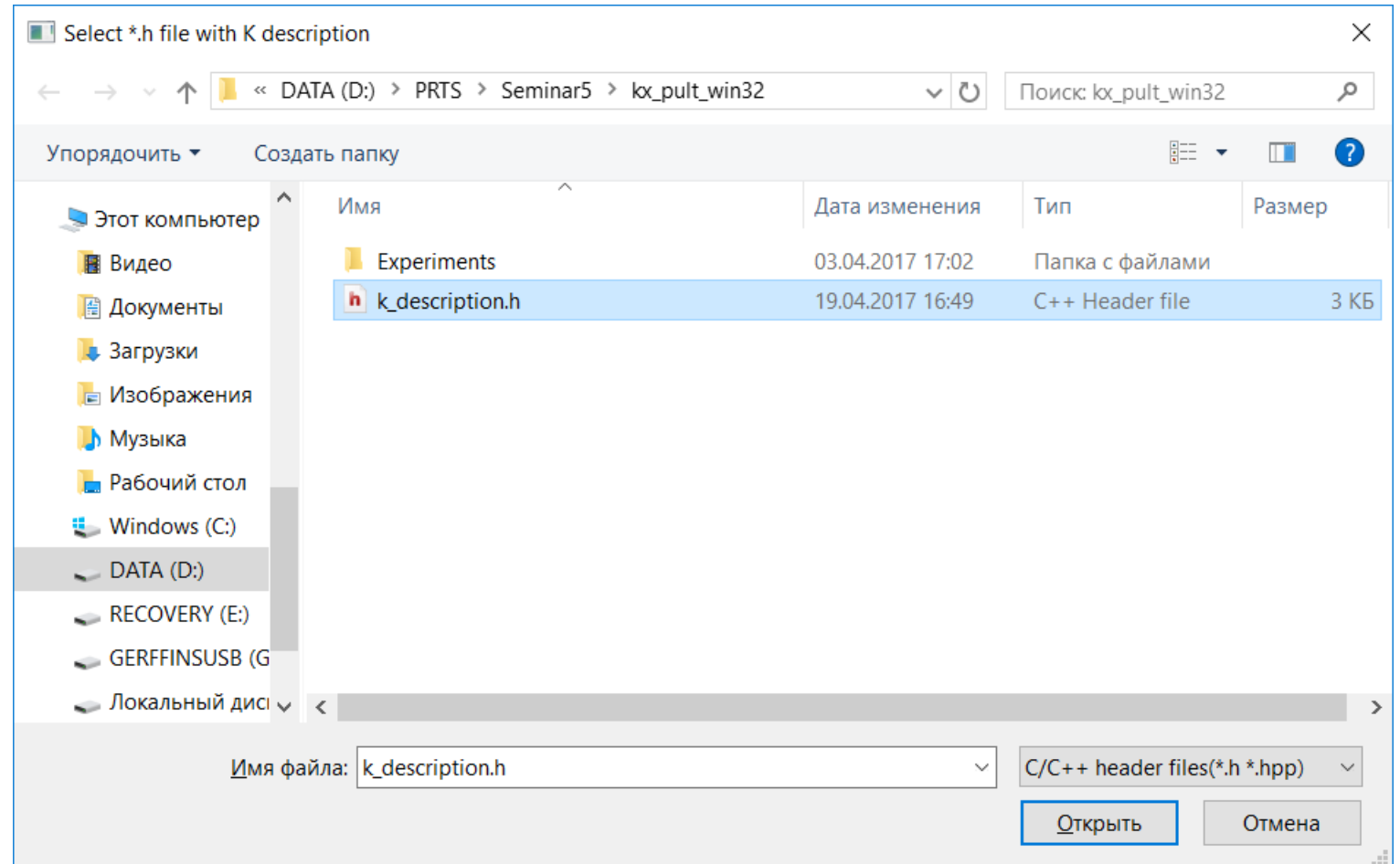
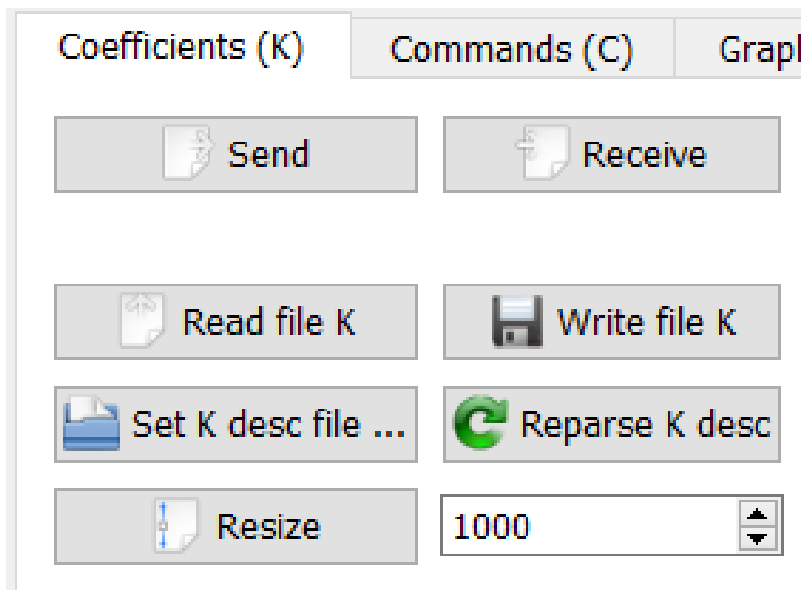
БФС ДРК. SU_ROV::BFS_DRK(). Описание коэффициентов в kx_pult.

Файл описания переменных:

```
Инструменты  Окно  Справка
k_description.h*  Umnp_limit: int
1  #ifndef K_DESCRIPTION_H
2  #define K_DESCRIPTION_H
3
4
5
6  enum KDescription {
7      ... Umars_h_limit = 11, //f Ограничение максимального сигнала СУ по маршруту
8      ... Upsi_limit, //f Ограничение максимального сигнала СУ по курсу
9      ... Uteta_limit, //f Ограничение максимального сигнала СУ по дифференту
10     ... Ugamma_limit, //f Ограничение максимального сигнала СУ по крену
11
12     ... Umvl_limit = 23, //f Ограничение максимального напряжения на ВМА МВЛ
13     ... Umnl_limit, //f Ограничение максимального напряжения на ВМА МНЛ
14     ... Umvp_limit, //f Ограничение максимального напряжения на ВМА МВП
15     ... Umnp_limit, //f Ограничение максимального напряжения на ВМА МНП
16
17 };
18
19 #endif // K_DESCRIPTION_H
20
```

БФС ДРК. Подключение описания коэффициентов в kx_pult.

В меню kx_pult'a выберите Set K desc file.. и подключите ваш файл описания переменных



БФС ДРК. Результат!)

8		0.00000000	0		
9		0.00000000	0		
10		0	0		
11	Umarsh_limit	10	10	double	Ограничение максимального сигнала СУ по маршу
12	Upsi_limit	10	10	double	Ограничение максимального сигнала СУ по курсу
13	Uteta_limit	10	10	double	Ограничение максимального сигнала СУ по дифференту
14	Ugamma_limit	10	10	double	Ограничение максимального сигнала СУ по крену
15		0.00000000	0		
16		0.00000000	0		
17		0.00000000	0		
18		0.00000000	0		
19		0.00000000	0		
20		0	0		
21		0	0		
22		0	0		
23	UmvI_limit	10	10	double	Ограничение максимального напряжения на ВМА МВЛ
24	UmnI_limit	UmvI_limit	10	double	Ограничение максимального напряжения на ВМА МНЛ
25	UmvP_limit	UmvI_limit	10	double	Ограничение максимального напряжения на ВМА МВП
26	UmnP_limit	UmvI_limit	10	double	Ограничение максимального напряжения на ВМА МНП
27		0.00000000	0		
28		0.00000000	0		

Контурь СУ. Курс. Описание коэффициентов kx_pult

K_description.h

```
....Kurs_ruchnoi_scale=40, //f Коэффициент усиления по курсу в ручном режиме
....Kurs_otladka=43, //n Коэффициент для настройки контура скорости
....Kurs_K1, //f Коэффициент K1 контура курса
....Kurs_K2, //f Коэффициент K2 контура курса
....Limit_Upsi=49, //f Ограничение максимального управляющего сигнала по курсу
```

Kx_pult

38		0.00000000	0		
39		0.00000000	0		
40	Kurs_ruchnoi_scale	1	1	double	Коэффициент усиления по курсу в ручном режиме
41		0.00000000	0		
42		0.00000000	0		
43	Kurs_otladka	0	0	int	Коэффициент для настройки контура скорости
44	Kurs_K1	3	3	double	Коэффициент K1 контура курса
45	Kurs_K2	2	2	double	Коэффициент K2 контура курса
46		0.00000000	0		
47		0.00000000	0		
48		0	0		
49	Limit_Upsi	10	10	double	Ограничение максимального управляющего сигнала по курсу
50		0.00000000	0		
51		0.00000000	0		
--		-----	-		

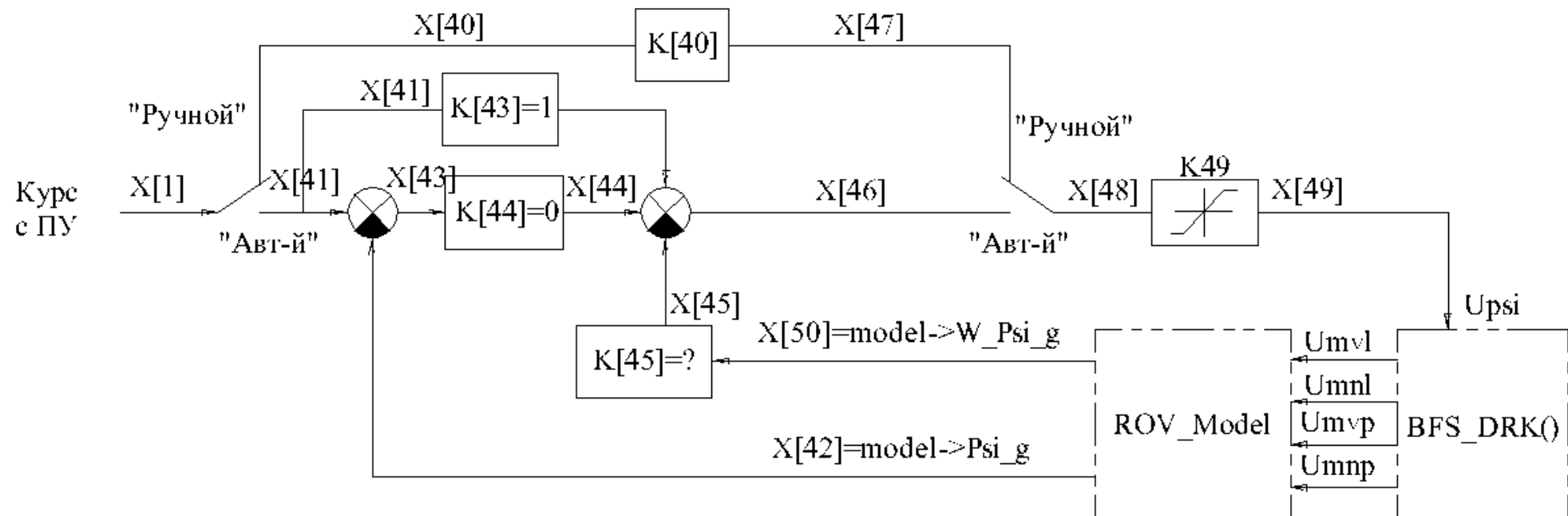
Контуры СУ. Курс. Настройка!

Этап 1. Контур скорости.

Для настройки контура скорости необходимо выставить следующие значения для коэффициентов:

$K[43]=1$; $K[44]=0$;

И подобрать коэффициент $K[45]$ таким образом, чтобы переходный процесс на выходе имел 5%-перерегулирование (или иные параметры качества, заданные по вашему ТЗ)



Контуры СУ. Курс. Настройка!

Этап 1. Контур скорости.

Noname - KX Pult

Coefficients (K) Commands (C) Graphics (X) Configuration


Send Receive

Read file K Write file K

Set K desc file ... Reparse K desc

Resize 1000

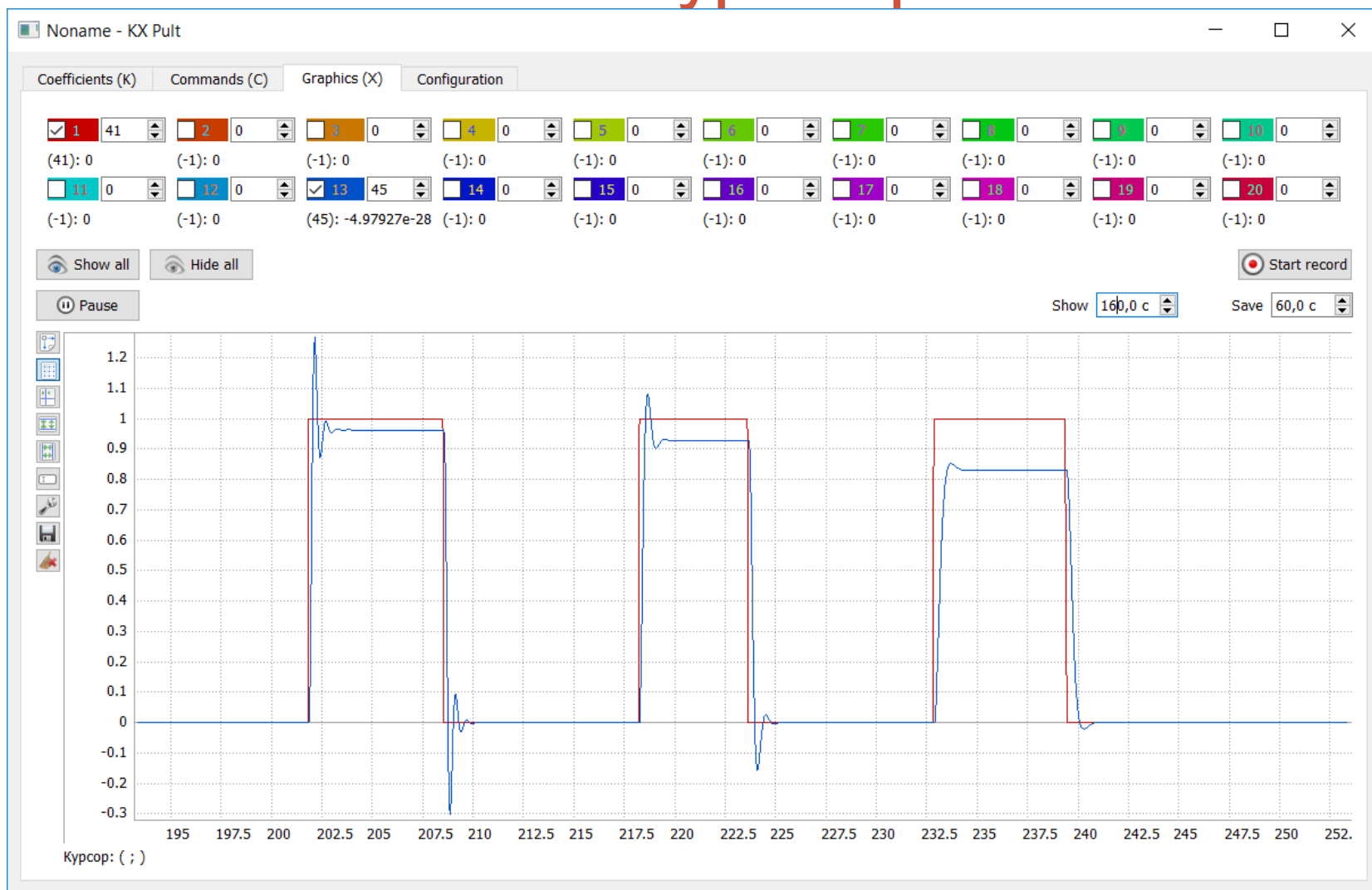
20/04/2017 14:34 - K sended
20/04/2017 14:41 - Write K file "k.dat": 1000 coeffs, 43581 bytes
20/04/2017 14:41 - K sended
20/04/2017 14:53 - Write K file "k.dat": 1000 coeffs, 43581 bytes
20/04/2017 14:53 - K sended
20/04/2017 14:01 - Write K file "k.dat": 1000 coeffs, 43581 bytes
20/04/2017 14:01 - K sended

☐ Hide empty ☐ Hide expressions ☐ Hide without errors Search: ☒ Auto calculate  Calculate 100%

Index	Name	Expression	Calculated	Type	Comment
33		0.00000000	0		
34		0.00000000	0		
35		0.00000000	0		
36		0.00000000	0		
37		0.00000000	0		
38		0.00000000	0		
39		0.00000000	0		
40	Kurs_ruchnoi_scale	1	1	double	Коэффициент усиления по курсу в ручном режиме
41		0.00000000	0		
42		0.00000000	0		
43	Kurs_otladka	1	1	int	Коэффициент для настройки контура скорости
44	Kurs_K1	0	0	double	Коэффициент K1 контура курса
45	Kurs_K2	2	2	double	Коэффициент K2 контура курса
46		0.00000000	0		
47		0.00000000	0		
48		0	0		
49	Limit_Upsi	10	10	double	Ограничение максимального управляющего сигнала по к...
50		0.00000000	0		
51		0.00000000	0		
52		0.00000000	0		

Контуры СУ. Курс. Настройка!

Этап 1. Контур скорости.



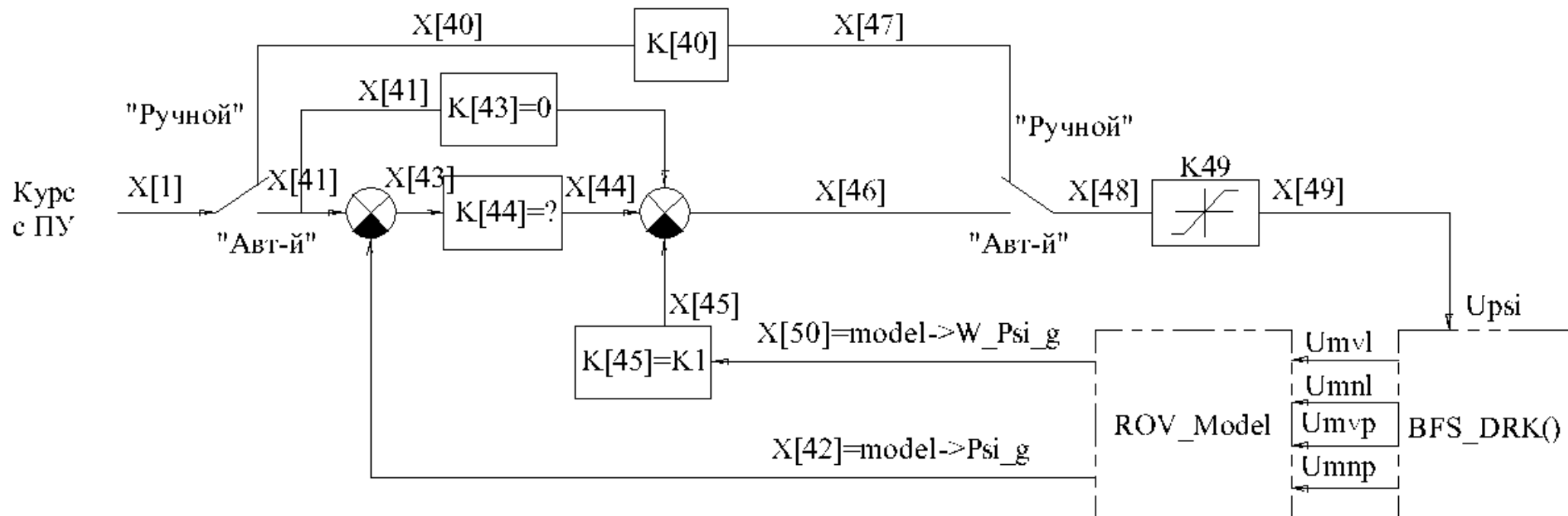
Контуры СУ. Курс. Настройка!

Этап 2. Контур положения.

Для настройки контура скорости необходимо выставить следующие значения для коэффициентов:

K[43]=0; K[45]=2;

И подобрать коэффициент $K[44]$ таким образом, чтобы переходный процесс на выходе имел 5%-перерегулирование (или иные параметры качества, заданные по вашему ТЗ)



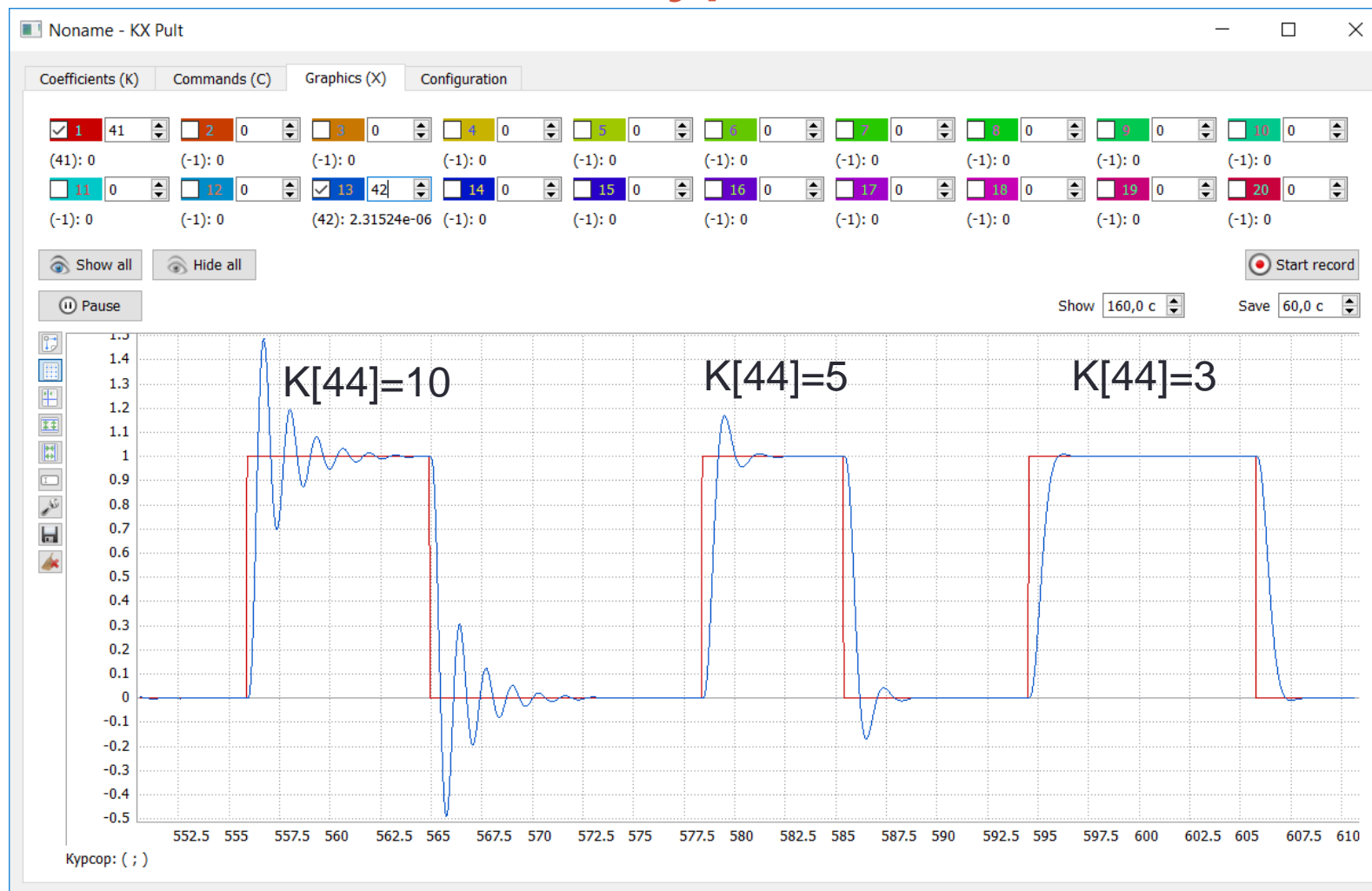
Контуры СУ. Курс. Настройка!

Этап 2. Контур положения.

39		0.00000000	0		
40	Kurs_ruchnoi_scale	1	1	double	Коэффициент усиления по курсу в ручном режиме
41		0.00000000	0		
42		0.00000000	0		
43	Kurs_otladka	0	0	int	Коэффициент для настройки контура скорости
44	Kurs_K1	3	3	double	Коэффициент K1 контура курса
45	Kurs_K2	2	2	double	Коэффициент K2 контура курса
46		0.00000000	0		
47		0.00000000	0		
48		0	0		
49	Limit_Upsi	10	10	double	Ограничение максимального управляющего сигнала по к
50		0.00000000	0		
51		0.00000000	0		

Контуры СУ. Курс. Настройка!

Этап 2. Контур положения.



Практическая часть 2

- Самостоятельная проработка и настройка контура дифферента.

НЛО № 2 про enum

Перечисление (enumeration) представляет собой набор целочисленных констант, задающих все допустимые значения переменной данного типа.

- enum тип_перечисления {список констант} список_переменных;
тип_перечисления и список_переменных указывать не обязательно.

Основные моменты:

- Каждая константа обозначает целое число (следовательно можно использовать вместо целочисленных переменных);
- Значение константы на 1 превышает значение предыдущей;
- Первая константа равна 0.
- В C++ enum задает полный тип;

НЛО № 2 про enum

Примеры:

```
enum SU_MODE {Ruchnoi, Automatiz};
```

Если хотим объявить переменную типа SU_MODE:

```
SU_MODE mode;
```

Или хотим использовать только объявленные константы, не создавая переменную типа SU_MODE:

```
int mode;
```

```
...
```

```
mode=Ruchnoi;
```