Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет: «Специальное машиностроение»

Кафедра: «Робототехнические системы и мехатроника»

Лабораторная работа № 7

по курсу «Теория автоматического управления»

Вариант 13

Выполнил: Садовец Роман
Группа: СМ7-62Б

Проверил(а):

Москва, 2024 г.

1. Контроллер на нечёткой логике

Имеется линейная система автоматического управления (далее САУ), имеющая вид, представленный на рисунке 1, в которой линейная часть задаётся передаточной функцией (1). Основная задача пункта – разработка контроллера на нечеткой логике. С принципом работы регулятора на нечеткой логике можно ознакомиться по ссылке из приложения 2.

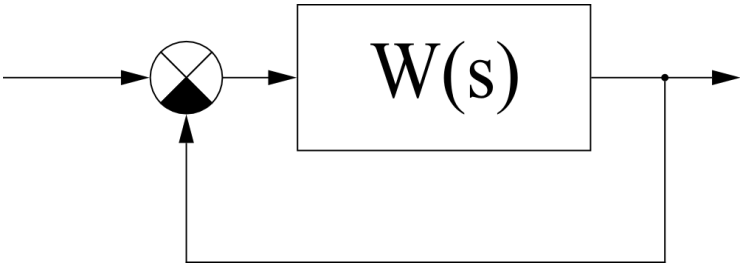

Рис. 1. Структурная схема линейной САУ

$$W_{Л}(p) = \frac{30}{p*(0{,}1p+1)^2} \quad (1)$$

Реализуем структурную схему средствами Simulink (рис. 2). Расчётные параметры в MatLab:

```
%% Linear part
Linear_part.k_1 = 30;
Linear_part.T_1 = 0.1;
Linear_part.T_2 = 0.1;
```

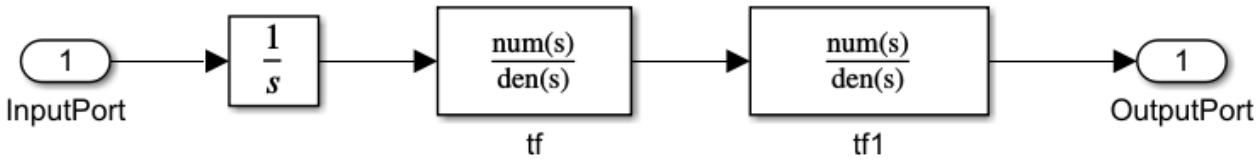

Рис. 2. Структурная схема линейной части САУ в Simulink

Перед разработкой регулятора на нечёткой логике, рассмотрим структурную схему с возбуждающим воздействием (рис. 3) и выходом системы (рис. 4).

Следующим пунктом выдвинем *требования* к системе после ввода регулятора:

$$\sigma_{max} \leq 20\% \quad (2)$$

$$t_{\text{пп}} \leq 1 \text{ сек} \quad (3)$$

После введения регулятора на нечеткой логике, структурная схема примет вид, представленный на рисунке 5. Блок контроллера на нечеткой логике, Fuzzy Logic Controller with Ruleviewer, настраивается с помощью специальной команды в Workspace Matlab - fuzzy(). После вызова функции появится окно с настройкой:

1. Входных/выходных параметров
2. Функций принадлежности
3. Баз правил
4. Инференции
5. Дефаззификации

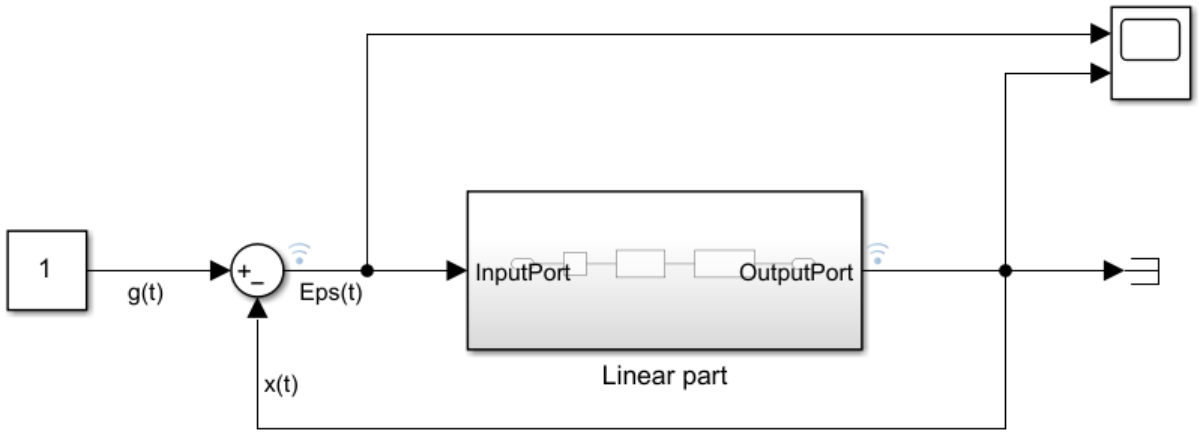

Рис. 3. Структурная схема линейной САУ в Simulink без контроллера

```
%% Setup Fuzzy Logic controller

Fuzzy_Logic_parameters.Ts = 0.005;

% Start setup menu of Fuzzy Logic controller
fuzzy();
```

Назначим входы и выход (п. 1) и инференцию (п. 4) как показано на рисунке 6. В нашем случае система обладает одним входом и одним выходом (SISO). В качестве входа системы назначим *величину ошибки системы Eps*, а выхода – *показания на выходе контроллера y*. В качестве реализации метода нечеткой вывода воспользуемся *методом Мамдани*.

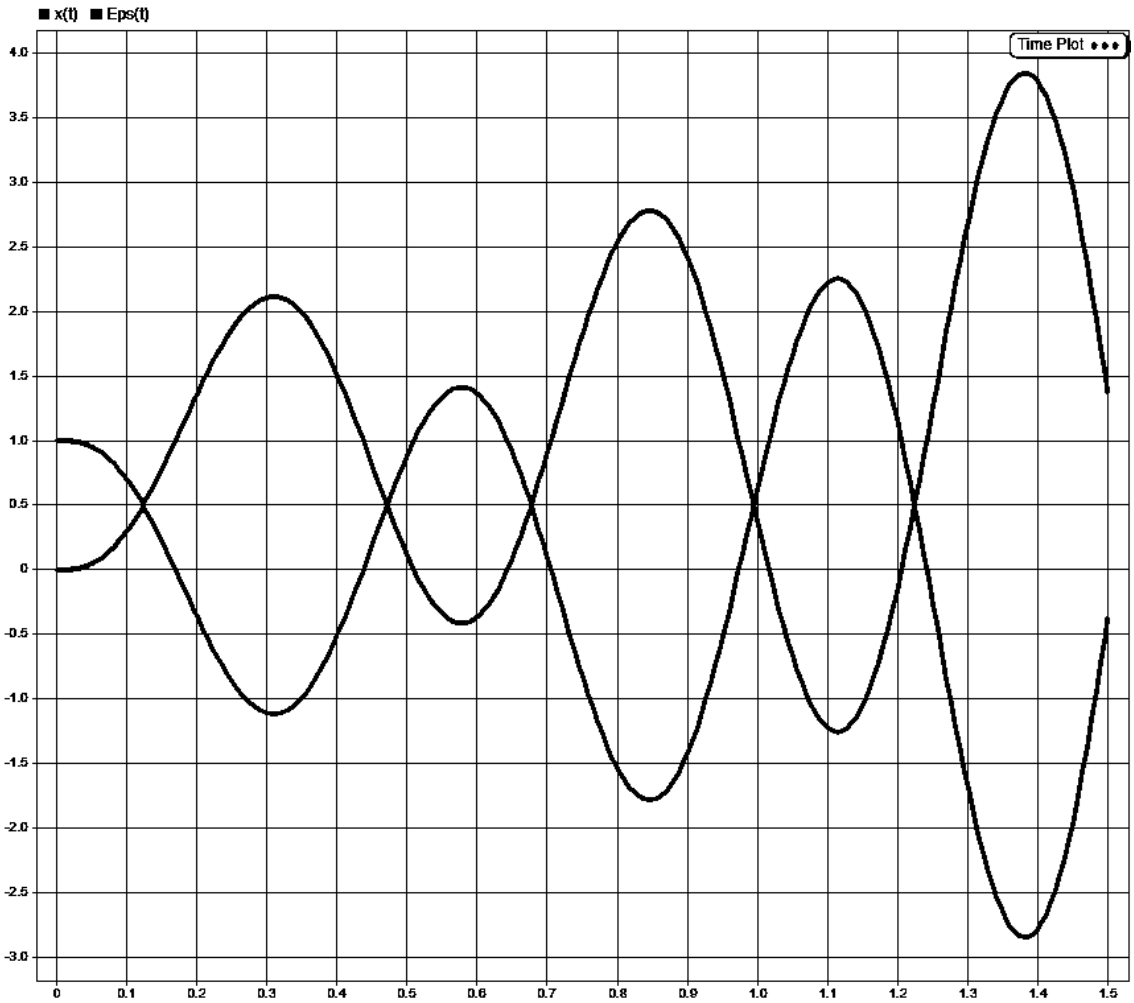

Рис. 4. Выход системы x(t) и ошибки Eps(t) без контроллера

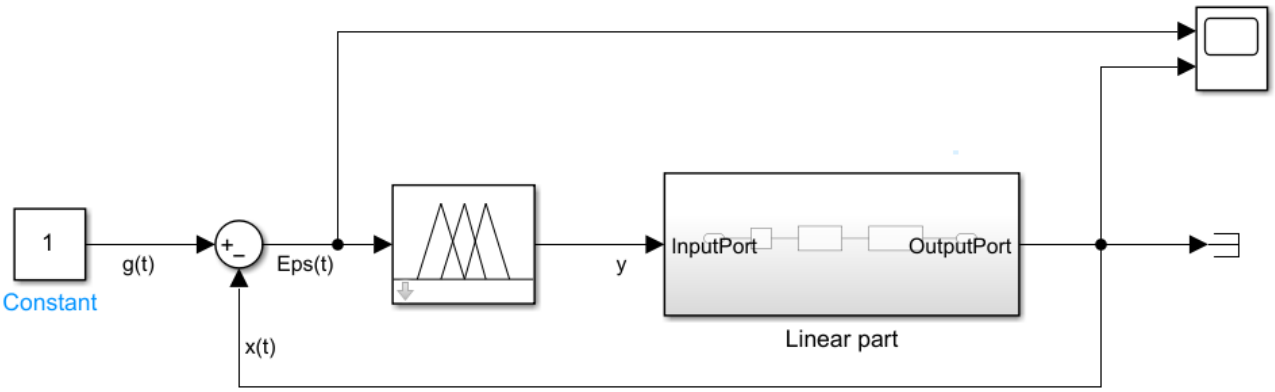

Рис. 5. Структурная схема линейной САУ в Simulink

Назначим лингвистические переменные (термы), описывающие систему:

1. Значение меньше возбуждающего сигнала – neg
2. Идеальная величина – perfect
3. Значение больше возбуждающего сигнала - pos

В качестве функций принадлежности выберем трапеции (рис. 7, 8). Важно, чтобы крайняя сторона трапеции, выходящая за рамки допустимого сигнала,

было дальше этой самой границы. Это необходимо для предотвращения случая подачи на вход сигнала по значению больше рассчитываемого.

Базу правил вывода показана на рисунке 9. Поскольку модель SISO, то вес каждого правила равен единице.

Полученную модель можно вывести либо напрямую в Workspace, либо сохранить в файле в формате “.fis”.

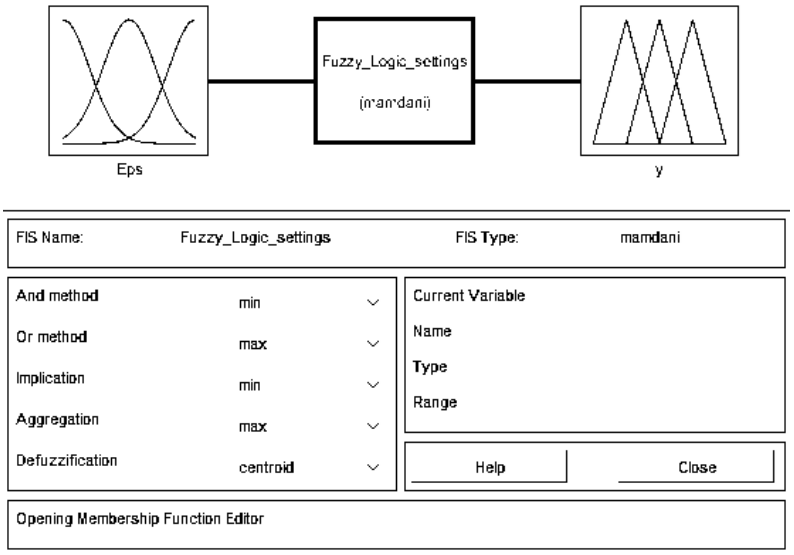

Рис. 6. Назначение входов и выходов контроллера, а также метода нечеткого вывода

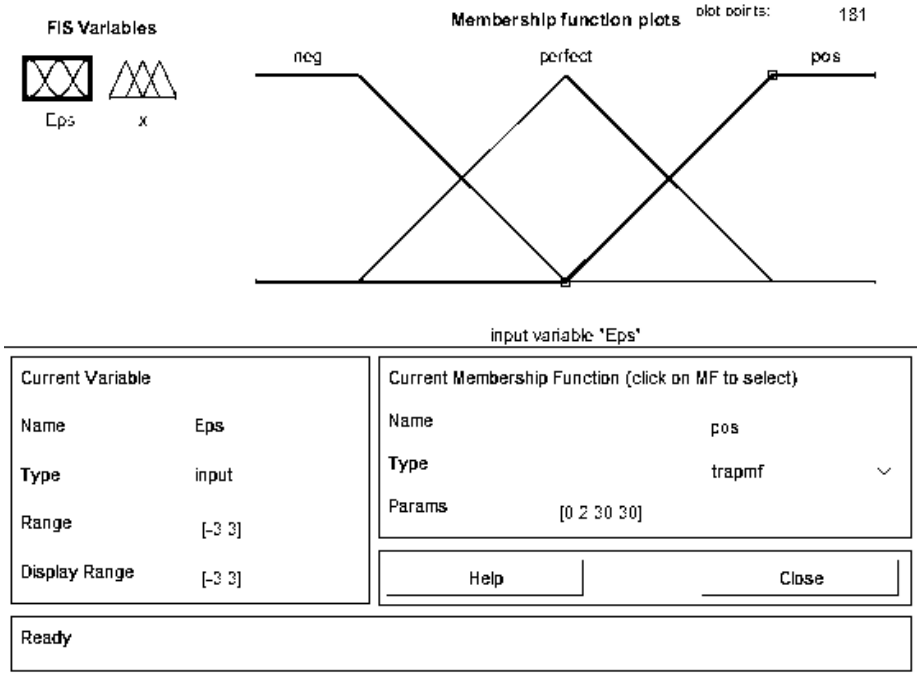

Рис. 7. Функции принадлежности для входного сигнала Eps

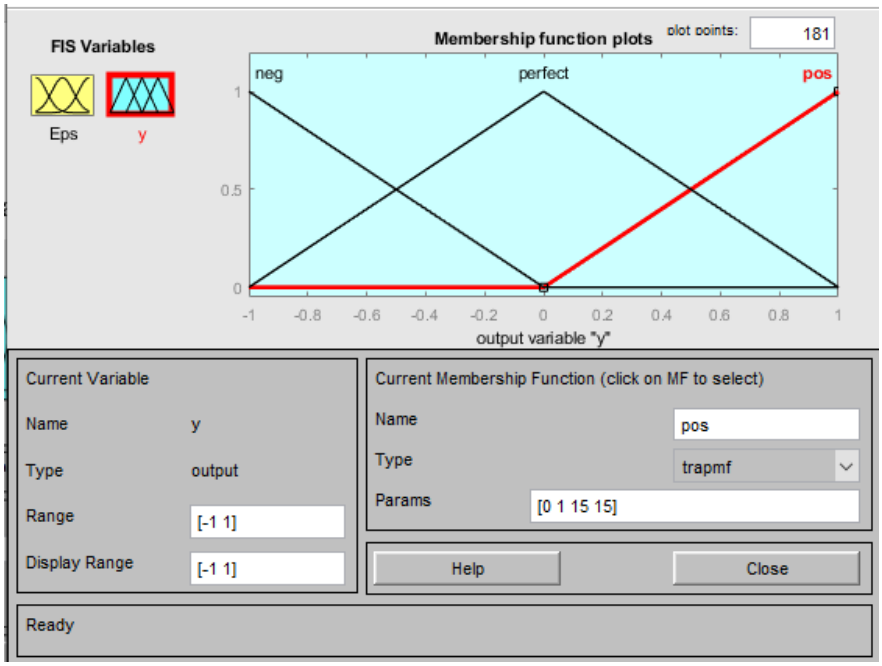

Рис. 8. Функции принадлежности для выходного сигнала y

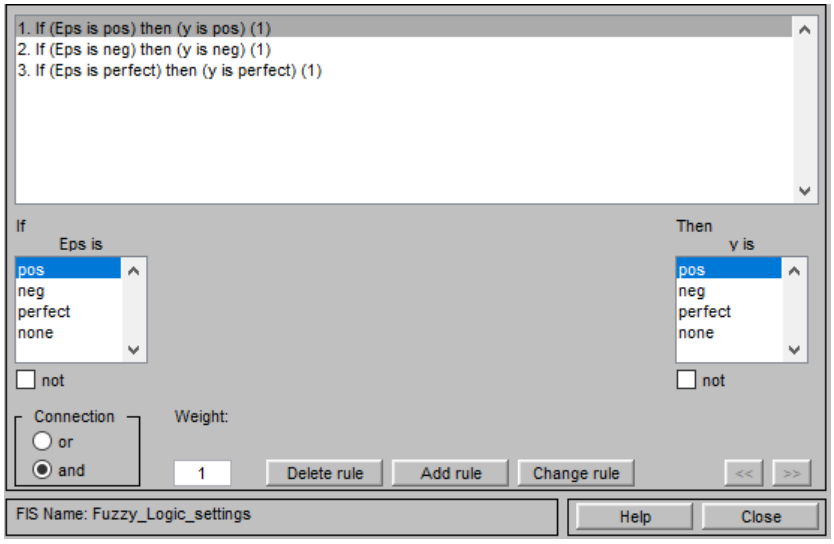

Рис. 9. База правил вывода регулятора на нечеткой логике

В качестве программного кода данный регулятор примет следующий вид:

```
[System]
Name='Fuzzy_Logic_settings'
Type='mamdani'
Version=2.0
NumInputs=1
NumOutputs=1
NumRules=3
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'
```

```
[Input1]
Name='Eps'
Range=[-3 3]
NumMFs=3
MF1='pos':'trapmf',[0 2 30 30]
MF2='neg':'trapmf',[-30 -30 -2 0]
MF3='perfect':'trimf',[-2 0 2]

[Output1]
Name='y'
Range=[-1 1]
NumMFs=3
MF1='pos':'trapmf',[0 1 15 15]
MF2='neg':'trapmf',[-15 -15 -1 0]
MF3='perfect':'trimf',[-1 0 1]

[Rules]
1, 1 (1) : 1
2, 2 (1) : 1
3, 3 (1) : 1
```

После экспорта настроек из настройки модели, получим значения, представленные на рисунке 10. Графики соответствую заданным требованиям к системе.

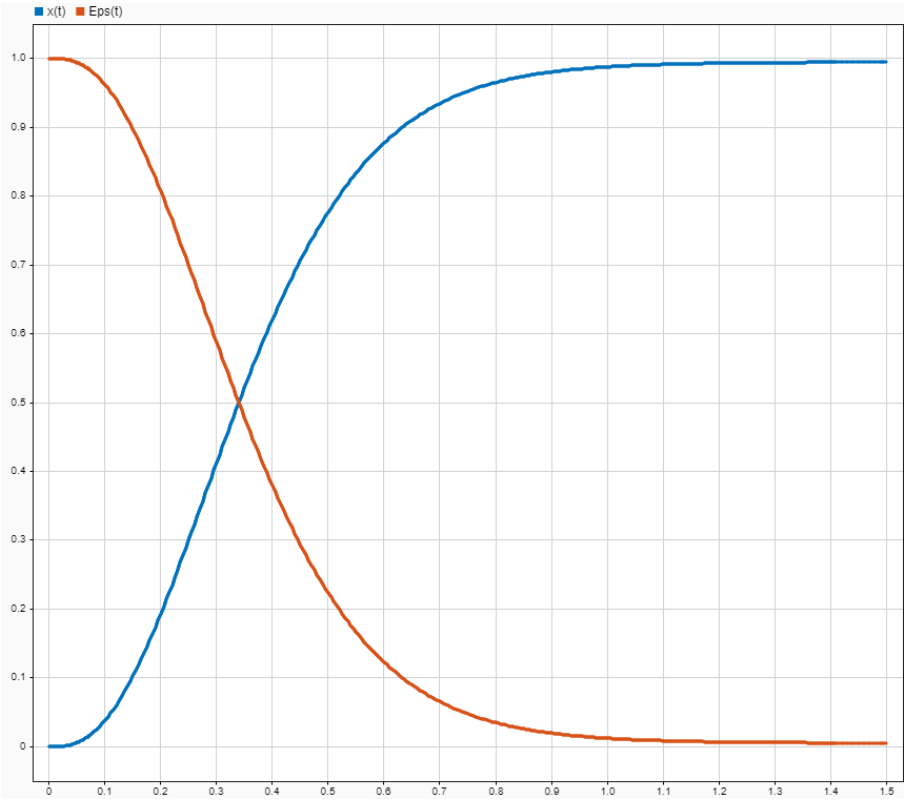

Рис. 10. Графики функций ошибки Eps(t) и сигнала на выходе линейной части САУ x(t)

2. LQR-контроллер

Для дальнейшего удобства работы переведем передаточную функцию линейной части САУ (1) в переменные состояния:

$$W_Л(p) = \frac{30}{0{,}01p^3+0{,}2p^2+p} = \frac{X(p)}{G(p)} \rightarrow$$

$$\rightarrow 0{,}01 * \dddot{x} + 0{,}2 * \ddot{x} + \dot{x} = 30 * g$$

$$y = x_1; \qquad x_1 = x, \qquad x_2 = \dot{x} = \dot{x_1}, \qquad x_3 = \ddot{x} = \dot{x_2},$$

$$\dot{x_3} = -20 * x_3 - 100 * x_3 + 30 * g$$

Тогда система примет вид (2):

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}, \; x(0) = x_0 \tag{4}$$

где $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -100 & -20 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 0 \\ 3000 \end{bmatrix}; C = [1 \quad 0 \quad 0]; D = 0$

Для понимания работы LQR-контроллера рекомендуется обратиться к приложению 3.

```
%% State space
State_space.A = [0 1 0
    0 0 1
    0 -100 -20];
State_space.B = [0
    0
    3000];
State_space.C = [1 0 0];
State_space.D = 0;
```

Основный замысел LQR-контроллера – это подача на вход линейных обратных связей по состоянию в формате:

$$u = -Kx, \qquad K \in \mathbb{R}^{mxn} \tag{5}$$

с минимизацией следующего квадратичного критерия качества:

$$J_{LQR} = \int_0^\infty [x(t)^T * Q * x(t) + u(t)^T * R * u(t) + 2 * x(t)^T * N * u(t)]dt, \tag{6}$$

где m и n – размерности матрицы управления u и состояния x соответственно.

Примем равными (в общем случае необходимы дополнительный расчёты):

$$Q = 5 * C^T * C, \qquad R = E, \qquad N = 0 \tag{7}$$

Чтобы функционал (6) был конечен, необходимо и достаточно, чтобы системы (4), замкнутая обратной связью (5) (далее ОС), была устойчива. Управляемость пары (A, B) гарантирует существование стабилизирующих обратных связей. В программе дополнительно введена проверка на управляемость системы (4).

В самом общем случае матрицу K находятся через решение уравнения Риккати:

$$A^T * P + P * A + Q - (P * B + N) * R^{-1} * (B^T * P + N^T) = 0 \quad (8)$$

Для решения подобной задачи в Matlab уже реализована команда по расчёту параметра *K* для контроллера, а именно *[K, S, P] = lqr(A, B, Q, R, N).*

```
%% LQR parameters

LQR_parameters.Q = 5 * State_space.C' * State_space.C;
LQR_parameters.R = 1;

%% Check conrollabillity of system

LQR_parameters.Co = rank( ctrb(State_space.A, State_space.B) );

if LQR_parameters.Co < rank(State_space.A)
   error('ERROR! System dont have controllability')
end

%% Calculate K parameter for LQR controller

[LQR_parameters.K, ~, ~] = lqr(State_space.A, State_space.B,
LQR_parameters.Q, LQR_parameters.R);

%% State space of new close-loop model with LQR controller
% new_sys = ss(A-B*K, B, C, D);
% step(new_sys)
```

Реализуем структурную схему с LQR-контроллером (рис. 11). Управление будет реализовано по положению, поэтому возбуждающее воздействие находиться в цепи $y = x_1$. Стоит понимать, что при отсутствии LQR-контроллера на выходе будут графики, аналогичные рисунку 4. После моделирования системы получим графики, представленные на рисунке 12 (для значения ошибки системы *Error(t)*) и рисунке 13 (для значения положения $y = x_1$)

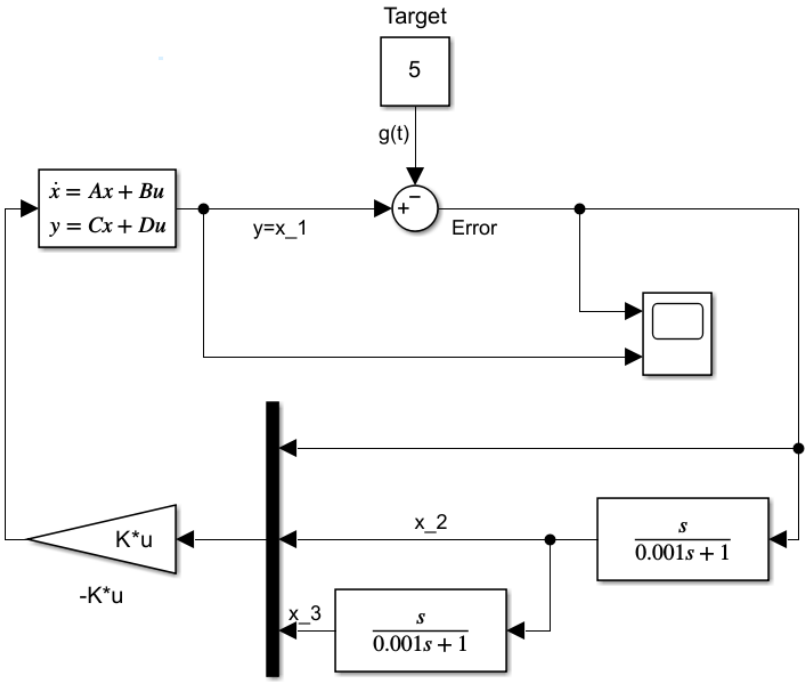

Рис. 11. База правил вывода регулятора на нечеткой логике

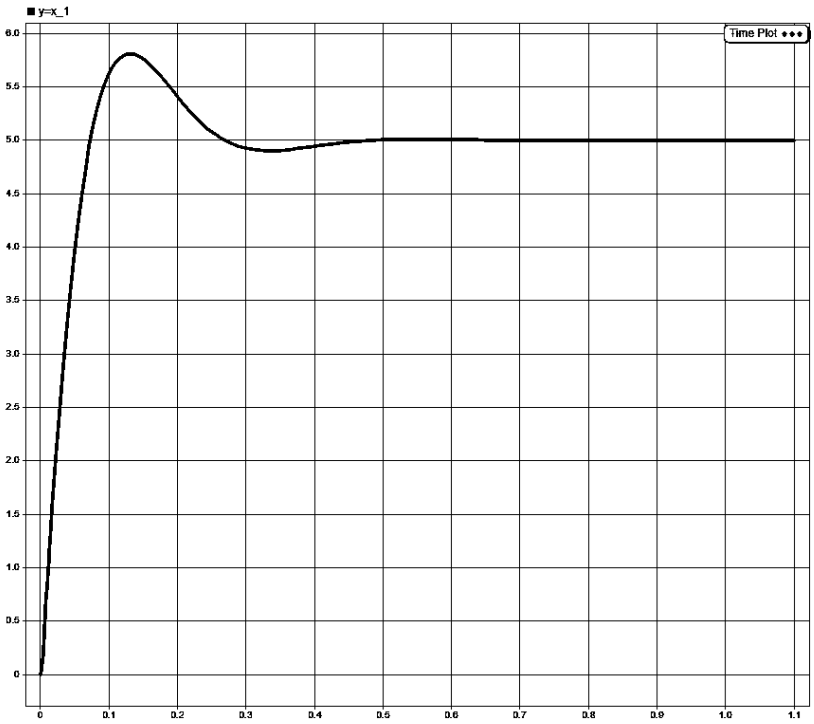

Рис. 12. Значение сигнала $y=x_1$ САУ с LQR-контроллером

Как можно заметить по выходным графикам, система с LQR-контроллером также удовлетворяем требованиям к системе (2) и (3).

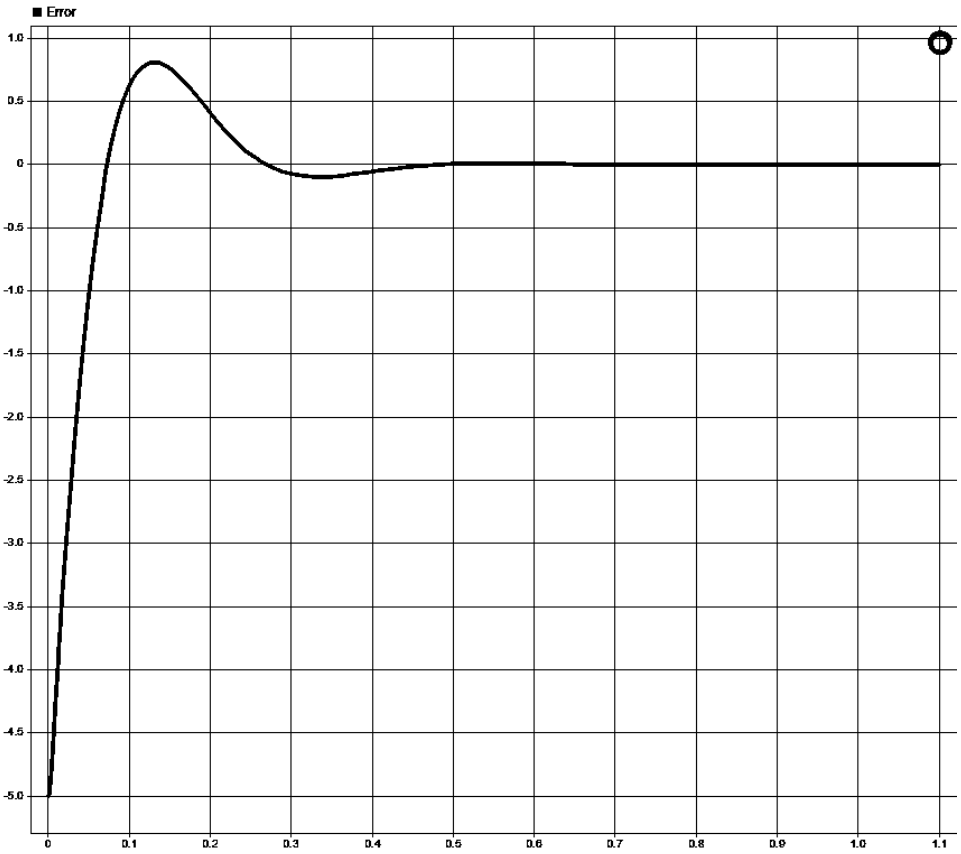

Рис. 13. Значение сигнала Error(t) САУ с LQR-контроллером

3. Наблюдатель Люенбергера

Задача наблюдения: необходимо найти такое преобразование, принимающее на вход управление *u(t)* и выход системы *y(t)*, а также выдающее оценку состояния системы $\hat{x}(t)$, что в установившемся режиме оценка состояния $\hat{x}(t)$ будет сколь угодно мало отличаться от реального состояния

$$\hat{x}(t) = \zeta(u, y) - ? : \lim_{t \to \infty} |x(t) - \hat{x}(t)| = 0 \quad (9)$$

Один из вариантов решение задачи наблюдения – *Наблюдатель Люенбергера*. Он не учитывает помехи в системе, опирается исключительно на модель системы, и описывается следующим дифференциальным уравнением:

$$\frac{d}{dt}\left(\hat{x}(t)\right) = A * \hat{x}(t) + B * u(t) + L * [\, y(t) - \hat{y}(t)], \quad (10)$$

где L – матрица констант; $\hat{y}(t) = C * \hat{x}(t)$. Структурная схема системы с наблюдателем представлена на рисунке 14 (рисунок взят из приложения 4).

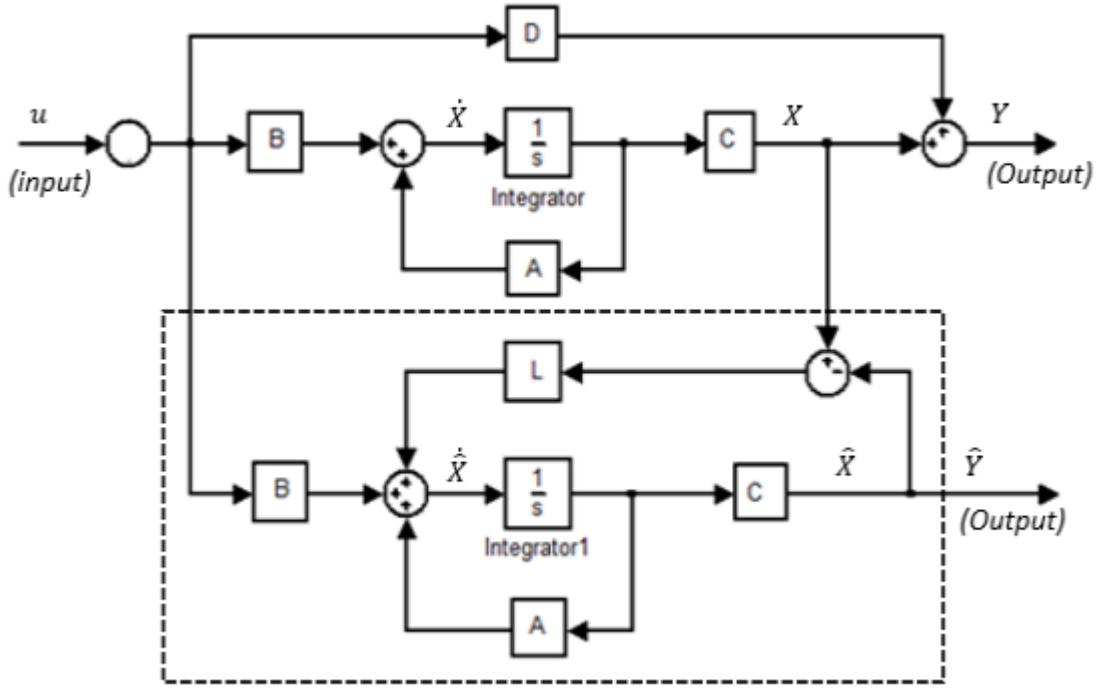

Рис. 14. Структурная схема САУ с наблюдателем Люенбергера

Доказательство его работоспособности и правильности решения можно прочитать в приложении 5.

Основная сложность в наблюдателе – это расчёт матрицы L. Её выбор отталкивается от полюсов Наблюдателя, что является нетривиальной задачей и

обычно опирается либо на опыт эксплуатации и тесты, либо на линеаризацию в конкретной точке. В качестве проверки работоспособности воспользуемся первым методом, поскольку он значительно проще в реализации.

В Matlab существуют команды, помогающие в расчете подобных матрица, в частности *acker()* (для SISO систем) и *place()* (для Multiple Input Multiple Output (MIMO) систем.

Таким образом, общая методология расчёта:

1. Переводим систему в переменные состояния (уже реализовано в системе (4))
2. Рассчитываем нули наблюдаемого объекта
3. Амперическим путем подбираем нули для Наблюдателя
4. Рассчитываем матрицу L для полученных нулей системы

```
%% Observer parameters
% First of all, define Observer gain L
% Will be use a pole placement method
% 1. Setup state space model
Observer_parameters.setup.sys = ss(State_space.A, State_space.B, ...
    State_space.C, State_space.D);

% 2. Search poles of the Observed system (define eigenvalues
% of the system).
Observer_parameters.setup.observed =
pole(Observer_parameters.setup.sys);

% 3. Define Observers poles [A-L*C]
% For this we can use two approaches:
% A - Linearizetion in point to the n derivative
% B - Random (expirence and black magic)
% We choose second one :)
Observer_parameters.setup.a = 5;
Observer_parameters.setup.b = 5;
Observer_parameters.setup.polo = Observer_parameters.setup.a * ...
    real(Observer_parameters.setup.observed) + ...
    imag(Observer_parameters.setup.observed) /
Observer_parameters.setup.b * 1i;

% 4. Calculate feedback vector (Observer gain) L
Observer_parameters.L = acker(State_space.A', State_space.C', ...
    Observer_parameters.setup.polo).';
```

Реализуем САУ с Наблюдателем Люенбергера в Simulink (рис. 15). Сигнал Shift добавлен для проверки работоспособности наблюдателя, что он действительно подстраивается под сигнал модели. Модель самой системы и Наблюдателя представлены на рисунках 16 и 17 соответственно. Будем детектировать три сигнала для переменных состояний x_1, x_2, x_3, а именно сигнал с реальной модели x_i, сигнал со сдвигом $x_i + S$, и сигнал, получаемый наблюдателем $\hat{x}_i, i = \overline{1,3}$. *Штрихпунктирная* – сигнал на выходе из модели; *штриховая* – сигнал на выходе со сдвигом; *сплошная* – выхода с наблюдателя Люенбергера. Все эти графики можно увидеть на рисунках 18, 19 и 20.

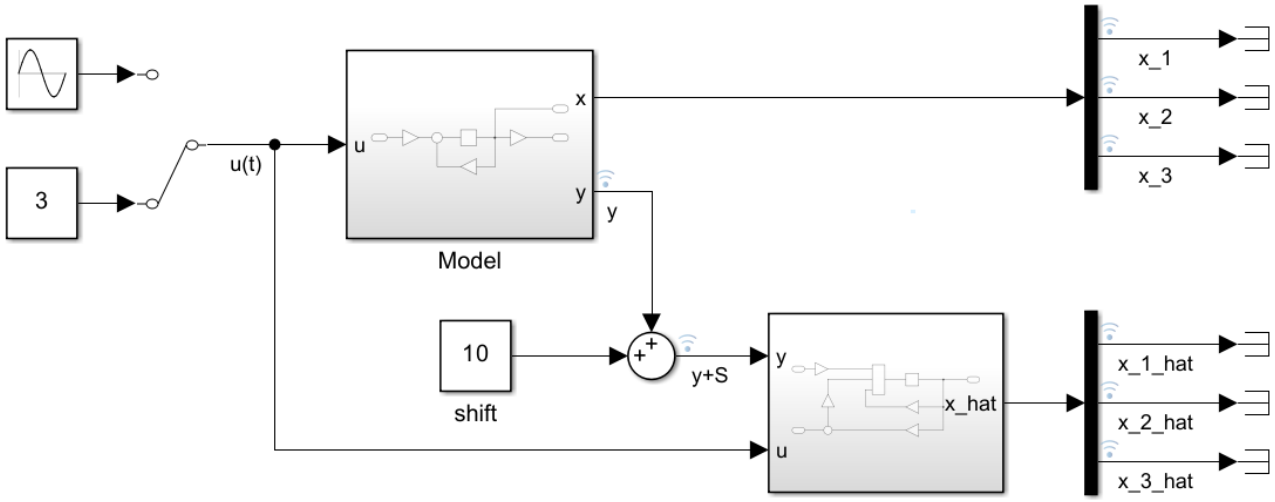

Рис. 15. Структурная схема САУ с наблюдателем Люенбергера в Simulink

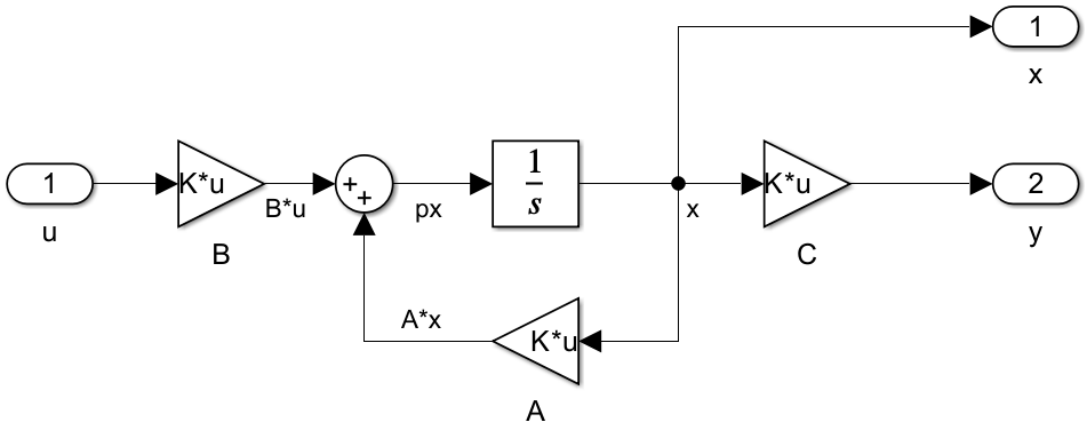

Рис. 16. Структурная схема САУ в переменных состояниях в Simulink

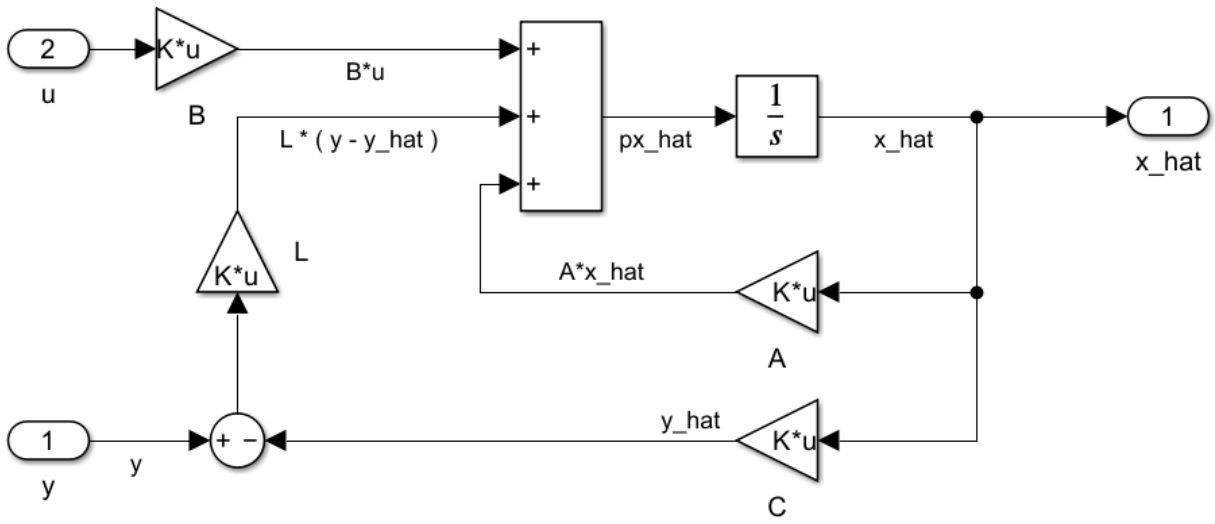

Рис. 17. Структурная схема Наблюдателя Люенбергера в Simulink

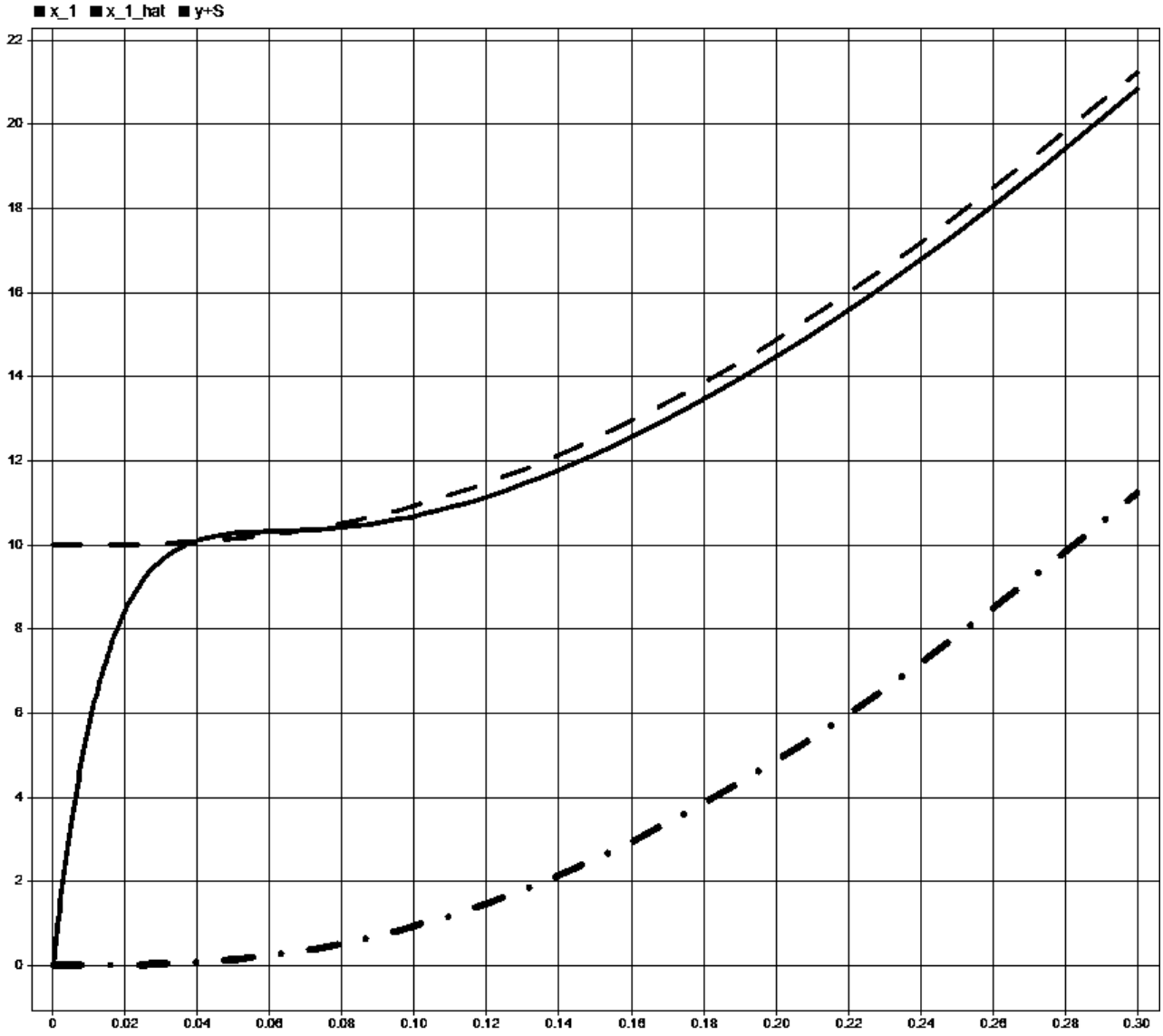

Рис. 18. Сигналы положения системы: с модели; с модели со сдвигом; с наблюдателя

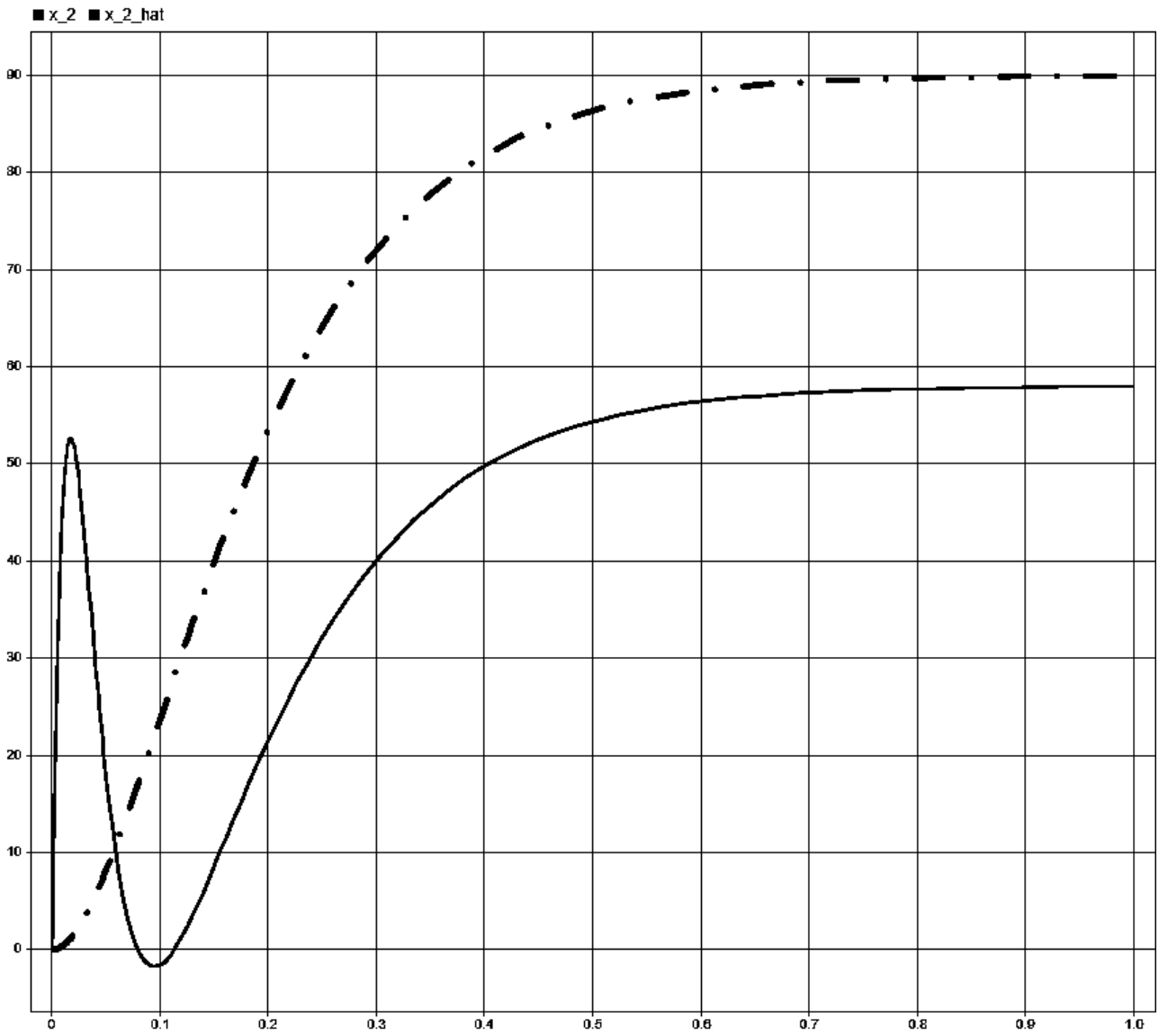

Рис. 19. Сигналы скорости системы: с модели; с наблюдателя

Как видно по графикам, наблюдатель подстраивается под возбуждающее воздействие и измеренные значения с системы. Такой инструмент может быть использован для определения параметров системы (скорости, ускорения) при наличии в структуре САУ только одного датчика (например, датчик положения) и математической модели системы.

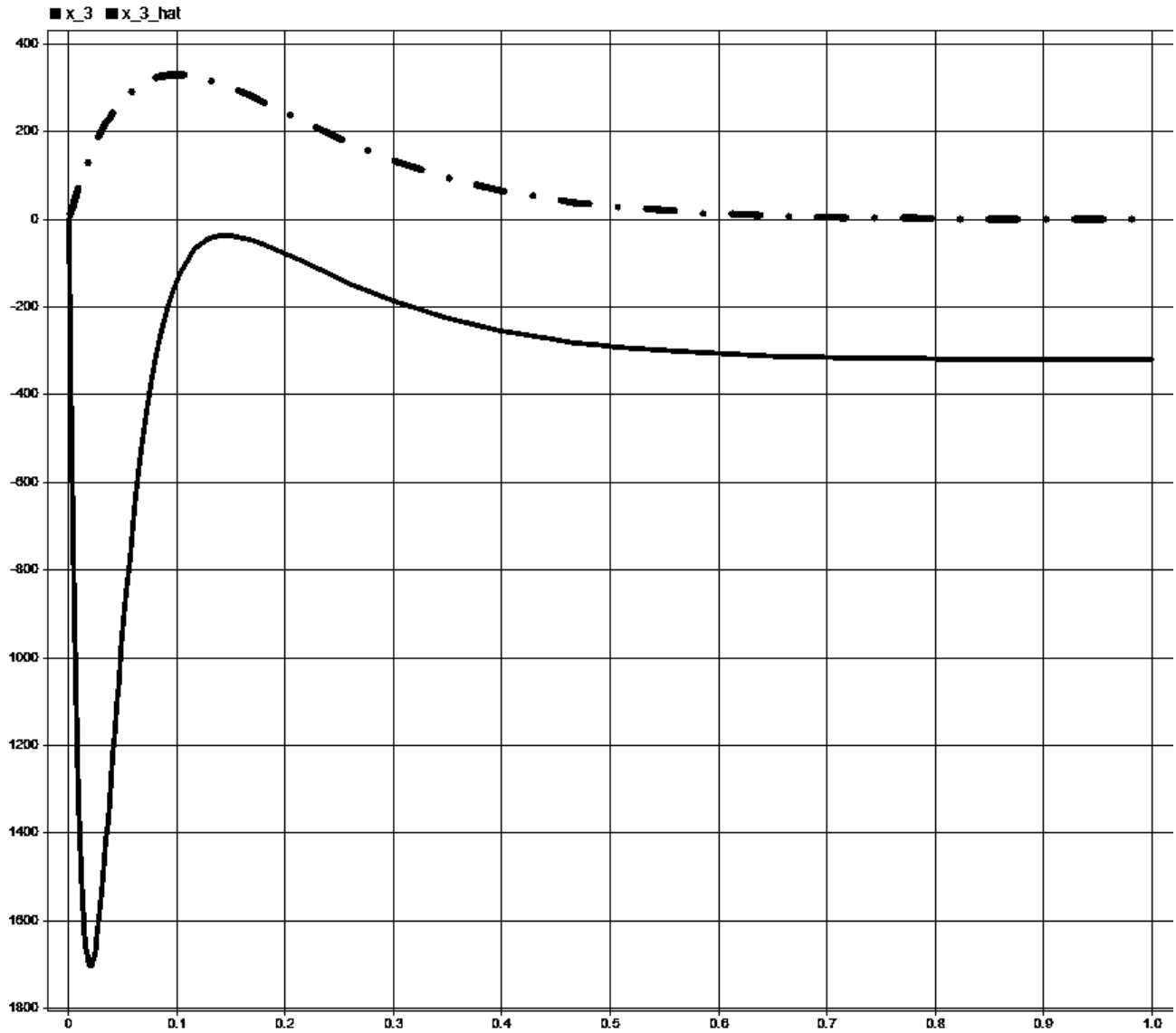

Рис. 20. Сигналы ускорения системы: с модели; с наблюдателя

4. Фильтр Калмана

Фильтр Калмана – математический алгоритм (рекурсивный фильтр), предназначенный для фильтрации внешних и внутренних шумов, действующих на и внутри системы.

«Фильтр Калмана — это мощнейший инструмент фильтрации данных. Основной его принцип состоит в том, что при фильтрации используется информация о физике самого явления. Скажем, если вы фильтруете данные со спидометра машины, то инерционность машины дает вам право воспринимать слишком быстрые скачки скорости как ошибку измерения. Фильтр Калмана интересен тем, что в каком-то смысле, это самый лучший фильтр.»[6]

Приближенная структурная схема САУ, с учетом внутренних и внешних шумов, имеет вид, представленный на рисунке 21. Plant – это идеальная математическая модель, которая не учитывает внутренние шумы модели. В общем случае фильтр Калмана описывается следующей системой уравнений:

$$\begin{cases} \dot{x} = A * x + B * u + G * w + \bar{o}(v) \\ y = Cx + Du + \bar{o}(w) + H * v \end{cases}, \quad (11)$$

где w – шум внутри системы; v – внешние шумы системы (например, шум из-за электромагнитных наводок); G и H – матрицы, описывающие влияние внутренних и внешних шумов соответственно; $\bar{o}(w)$ и $\bar{o}(v)$ – это б.м. величины внутренних шумов, перекрестно влияющих на входы/выходы.

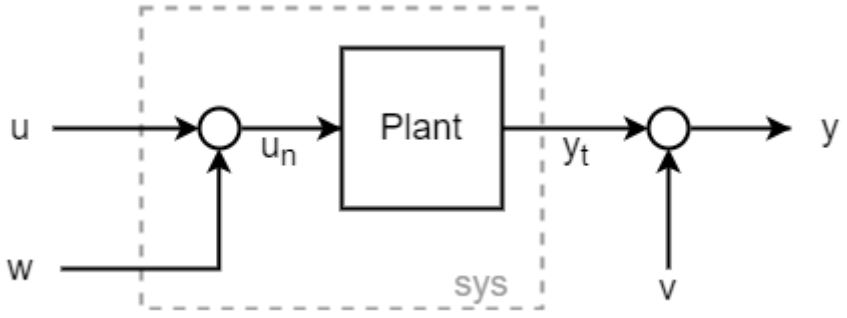

Рис. 21. Структурная схема модели с учетом шумов

Чаще всего шумы системы (11) описывают с помощью распределения Гаусса, либо иных вероятностных методов, с некоторыми из которых можно ознакомиться в статье из приложения 6. По статье из приложения 7 можно

ознакомиться с основными формулами и математическими выражениями, часто применяющимися в фильтре Калмана.

Принцип работы фильтра представлен на рисунке 22 (взят из приложения 7). В таблицах 1 и 2 можно ознакомиться с основными выражениями в фильтре Калмана и наименованиями переменных соответственно.

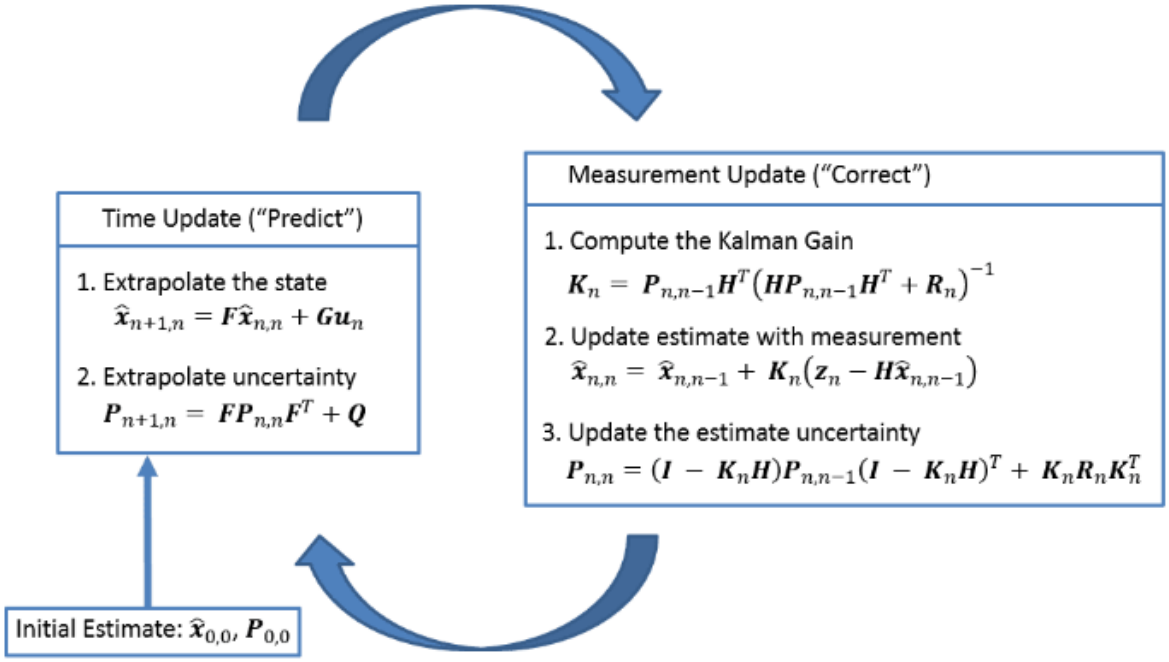

Рис. 22. Структурная схема модели с учетом шумов

	Equation	Equation Name	Alternative names
Predict	$\hat{x}_{n+1,n} = F\hat{x}_{n,n} + Gu_n$	State Extrapolation	Predictor Equation Transition Equation Prediction Equation Dynamic Model State Space Model
	$P_{n+1,n} = FP_{n,n}F^T + Q$	Covariance Extrapolation	Predictor Covariance Equation
Update (correction)	$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - H\hat{x}_{n,n-1})$	State Update	Filtering Equation
	$P_{n,n} = (I - K_nH)\,P_{n,n-1}(I - K_nH)^T + K_nR_nK_n^T$	Covariance Update	Corrector Equation
	$K_n = P_{n,n-1}H^T\left(HP_{n,n-1}H^T + R_n\right)^{-1}$	Kalman Gain	Weight Equation
Auxiliary	$z_n = Hx_n$	Measurement Equation	
	$R_n = E\left(v_nv_n^T\right)$	Measurement Covariance	Measurement Error
	$Q_n = E\left(w_nw_n^T\right)$	Process Noise Covariance	Process Noise Error
	$P_{n,n} = E\left(e_ne_n^T\right) = E\left((x_n - \hat{x}_{n,n})(x_n - \hat{x}_{n,n})^T\right)$	Estimation Covariance	Estimation Error

Табл. 1. Основные математические выражения для фильтра Калмана

Term	Name	Alternative term	Dimensions
$\boldsymbol{x}$	State Vector		$n_x \times 1$
$\boldsymbol{z}$	Measurements Vector	$\boldsymbol{y}$	$n_z \times 1$
$\boldsymbol{F}$	State Transition Matrix	$\boldsymbol{\Phi, A}$	$n_x \times n_x$
$\boldsymbol{u}$	Input Variable		$n_u \times 1$
$\boldsymbol{G}$	Control Matrix	$\boldsymbol{B}$	$n_x \times n_u$
$\boldsymbol{P}$	Estimate Covariance	$\boldsymbol{\Sigma}$	$n_x \times n_x$
$\boldsymbol{Q}$	Process Noise Covariance		$n_x \times n_x$
$\boldsymbol{R}$	Measurement Covariance		$n_z \times n_z$
$\boldsymbol{w}$	Process Noise Vector	$\boldsymbol{y}$	$n_x \times 1$
$\boldsymbol{v}$	Measurement Noise Vector		$n_z \times 1$
$\boldsymbol{H}$	Observation Matrix	$\boldsymbol{C}$	$n_z \times n_x$
$\boldsymbol{K}$	Kalman Gain		$n_x \times n_z$
$\boldsymbol{n}$	Discrete-Time Index	$\boldsymbol{k}$	

Табл. 2. Наименование и размерность переменных в фильтре Калмана

где n_x – размерность матрицы A; n_z – размерность матрицы C; n_u – размерность матрицы B

Суммирая вышесказанное, основная задача для реализации Фильтра Калмана – это расчёт матрицы усиления Калмана *K*. Обычно в случае реализации фильтра в переменных состояний эту матрицу заменяют на L (модель сильно напоминает наблюдатель Люенбергера). Её можно найти через решение

уравнения Риккати для следующего дифференциального уравнения (уравнение фильтра):

$$\frac{d\hat{x}}{dt} = A\,\hat{x} + Bu + L(y - C\,\hat{x} - Du) \qquad (12)$$

Внутри Matlab уже присутствует команда для расчёта параметра усиления L – [kalmf, L, P] = kalman(sys, Q, R, N) (см. прил. 8), где

Q – ковариант шума в системе, обычно задается как $Q = const * C^T * C$,

R – ковариант шума в датчиках, обычно принимается равным единице,

N – ковариант между шумами w и v. Принимаем, что шумы между собой никак не связаны, а значит $N = 0$.

Примем константу в ковариантe шума равной единице. Зададим переменные в Matlab

```
%% Parameters of noise

Kalman_parameters.Ts = 0.01;
Kalman_parameters.Noise_power_input = 0.05;
Kalman_parameters.Noise_power_output = 5;
Kalman_parameters.seed_input = 12357;
Kalman_parameters.seed_output = 73745;

%% Kalman parameters

Kalman_parameters.sys = ss (State_space.A, State_space.B, ...
    State_space.C, State_space.D);

Kalman_parameters.Q = 1;
Kalman_parameters.R = 1;
Kalman_parameters.N = 0;

[Kalman_parameters.kalmf, Kalman_parameters.L, Kalman_parameters.P] =
...
    kalman(Kalman_parameters.sys, Kalman_parameters.Q, ...
    Kalman_parameters.R, Kalman_parameters.N);
```

Создадим структурную схему фильтра Калмана (рис. 23) и общую структурную схему САУ с внедренным фильтром (рис. 24). Для сравнения графиков также будем снимать показания идеальной модели (без учета внешних шумов системы).

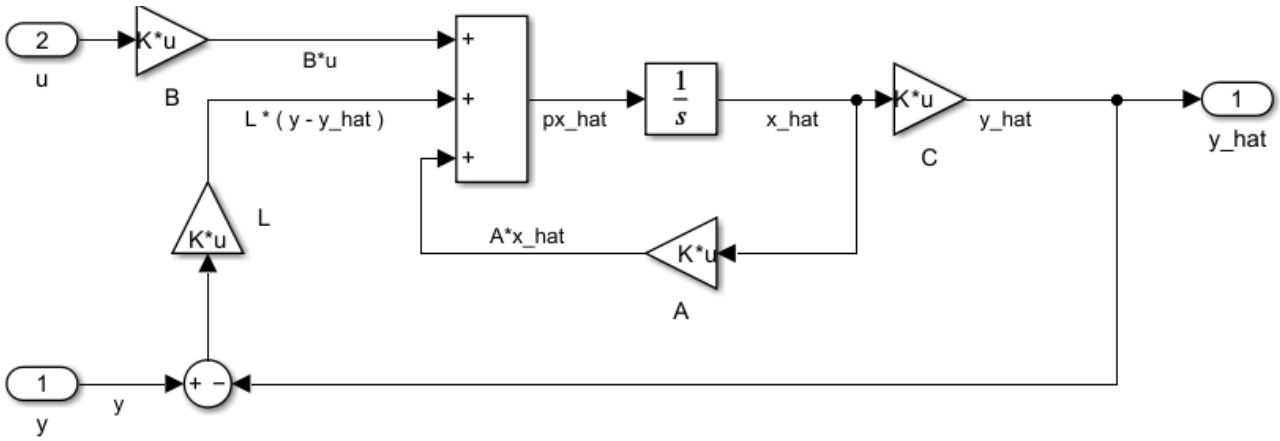

Рис. 23. Структурная схема фильтра Калмана, реализованная через переменные состояния

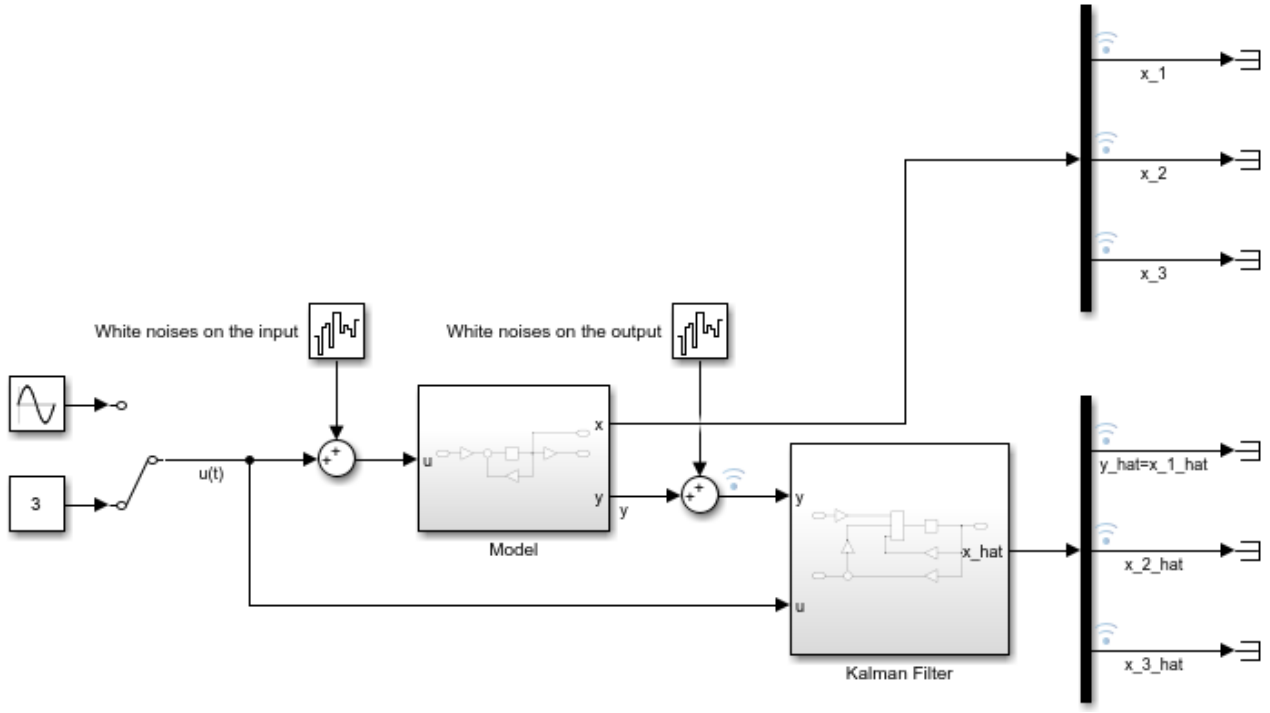

Рис. 24. Структурная схема САУ с фильтром Калмана

Перед сравнением идеальных сигналов $x(t)$ (учитываем только внутренние шумы системы) с выходными $\hat{x}(t)$ (после фильтра Калмана), посмотрим на сигнал на выходе системы с внешним шумом и выходом фильтра $\hat{y}(t)$ (рис. 25). Далее рассмотрим необходимые сигналы (рис. 26, 27, 28).

Как можно было заметить, введенные внешние шумы в систему – крайне велики, поэтому расхождение в некоторые моменты достаточно сильное, однако общая «тенденция» (сонаправленность) графиков совпадает, и фильтр Калмана держит среднее значение около требуемого графика (графика идеальной модели).

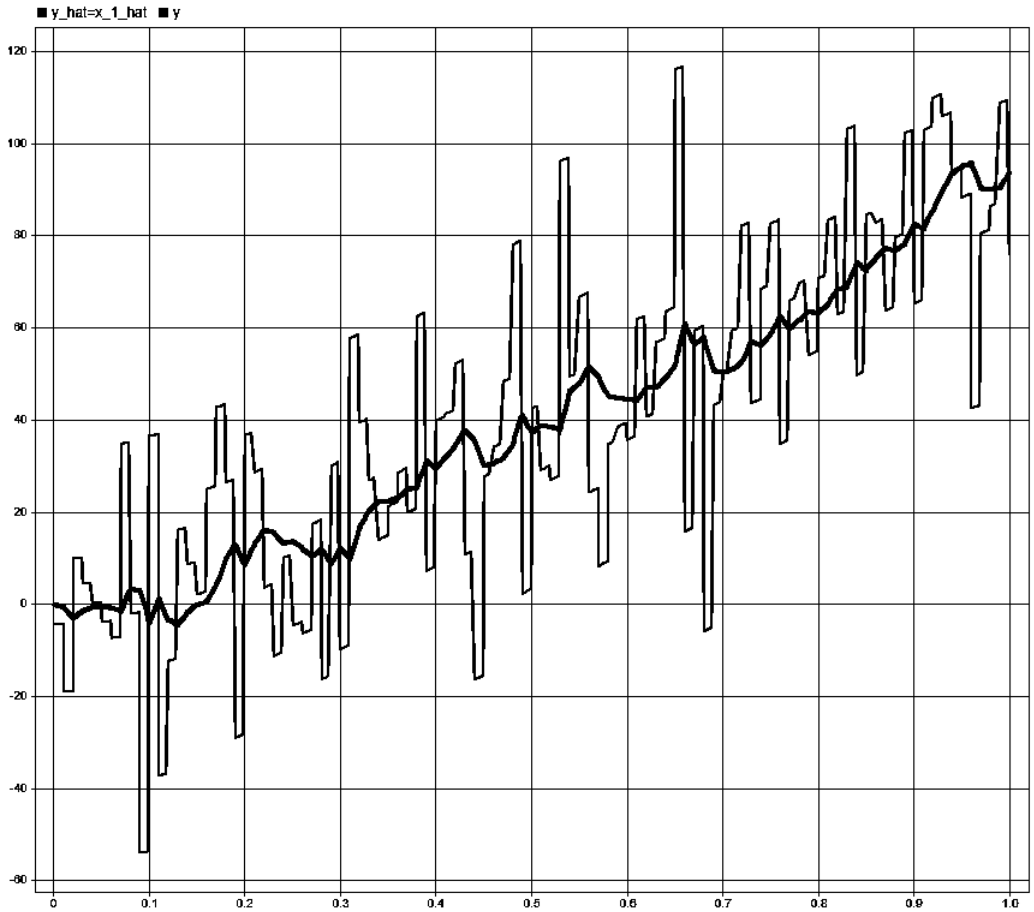

Рис. 25. Графики сигнала после фильтра Калмана и с модели (с учетом внешних шумов)

Рис. 26. Графики $x_1(t)$ и $\hat{x}_1(t)$

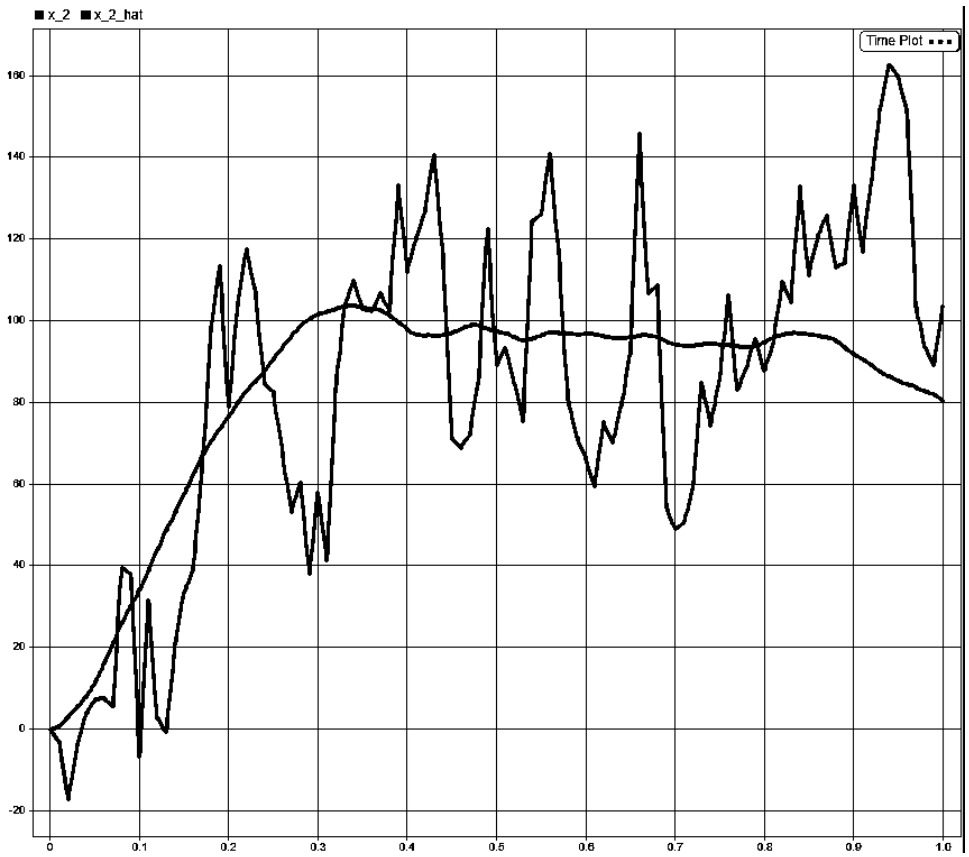

Рис. 27. Графики $x_2(t)$ и $\hat{x}_2(t)$

Рис. 28. Графики $x_3(t)$ и $\hat{x}_3(t)$

5. Логическое управление

В случае логического управления, на вход САУ подается только одно из трех значений: -1, 0, 1. Величина подаваемого значения на САУ определяется сравнением ошибки системы Error(t) и скоростью системы Speed(t). Общую структурную схему можно представить в виде, изображенном на рис. 29 (интегратор был вынесен из блока Linear part, в целях более удобного представления модели).

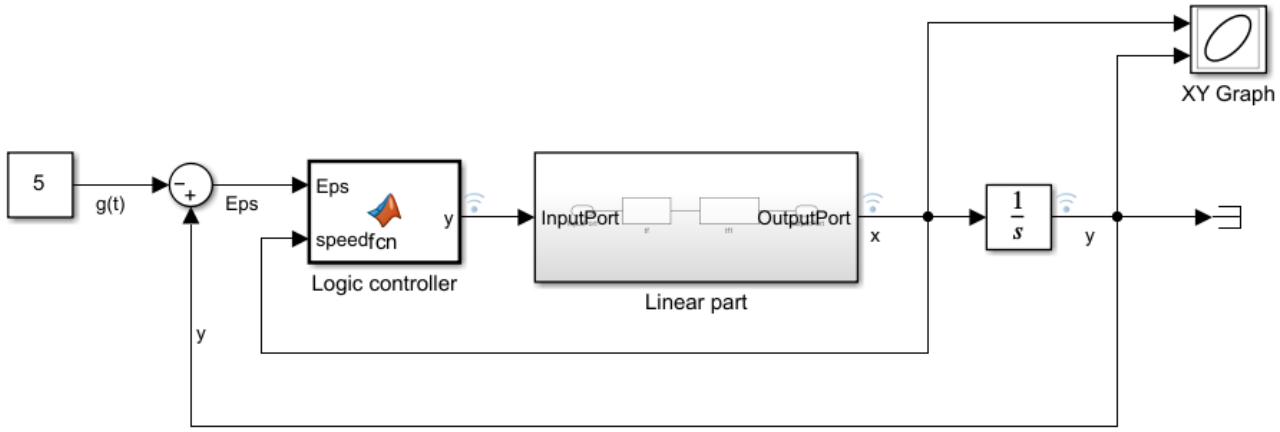

Рис. 29. Структурная схема с логическим управлением

Логическое управление определяется следующей функцией

```
function y = fcn(Eps, speed)

a_1 = 0.5;
a_2 = 2;

if ( Eps < -a_1 ) && (speed < -a_2)
    y = 1;
elseif ( Eps < -a_1 ) && (speed < a_2)
    y = 1;
elseif ( Eps > a_1 ) && (speed < a_2)
    y = -1;
elseif ( Eps > a_1 ) && (speed > a_2)
    y = -1;
else
    y = 0;
end
```

Проблема подобного управления – этого его сложная настройка для высокоточных систем. Регулятор при малых значениях скорости может крайне сильно «прыгать» от значения к значению.

Выведем графики на выходе системы y(t) (рис. 30), на выходе из логического контроллера (рис. 31), а также фазовый портрет системы (рис. 32).

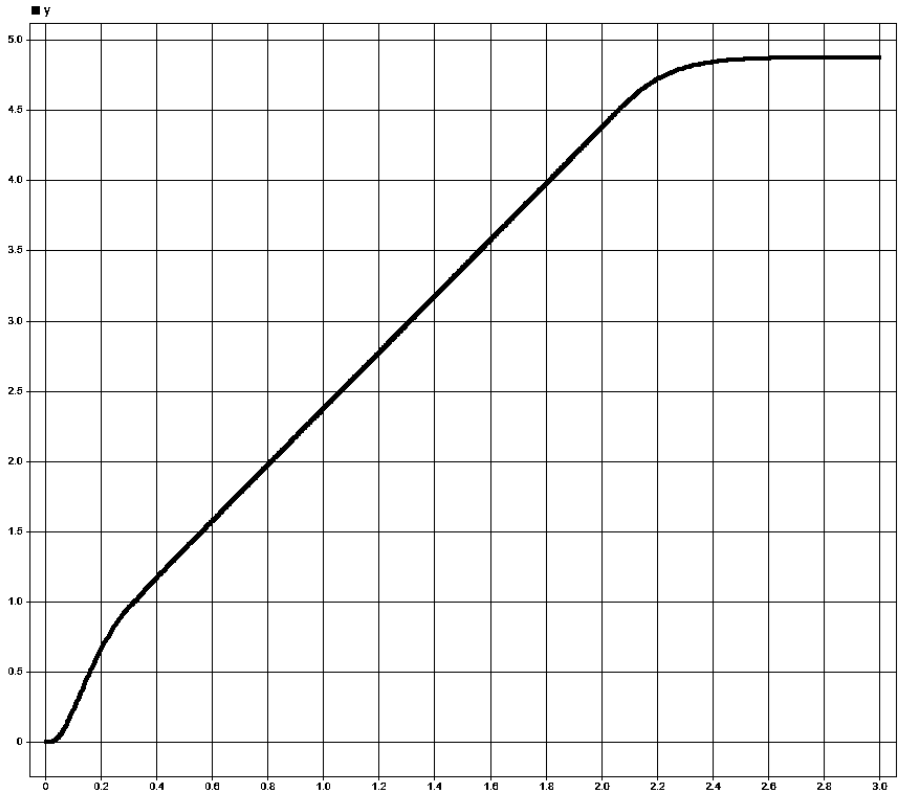

Рис. 30. Сигнал на выходе системы

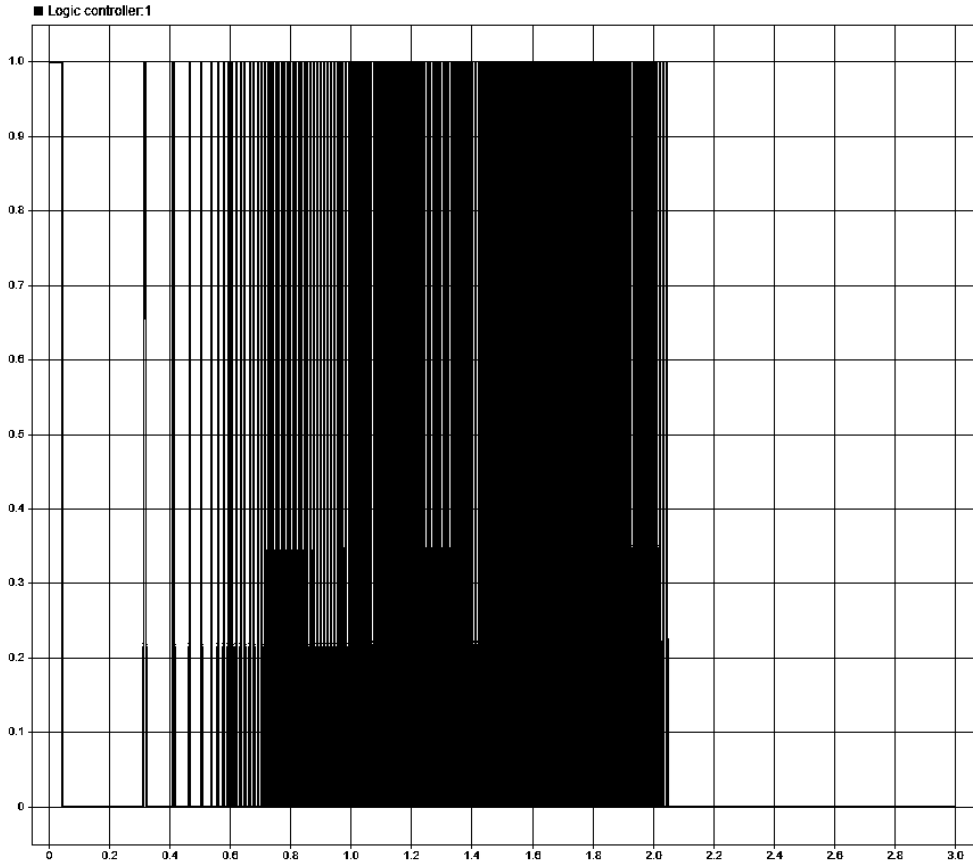

Рис. 31. Сигнал на выходе логического контроллера

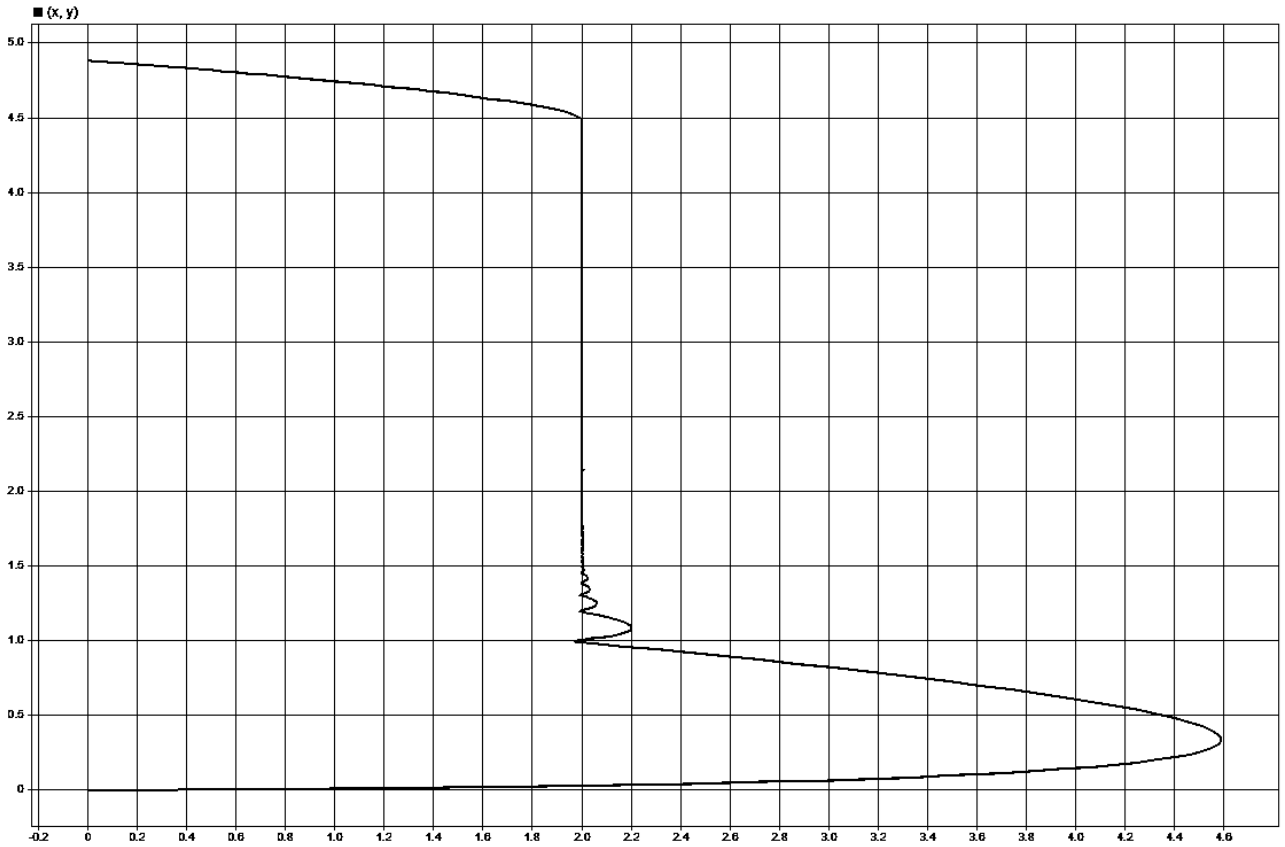

Рис. 32. Фазовый портрет системы

Приложение

1. Публичный репозиторий для лабораторных по ТАУ // GitHub URL: https://github.com/RiXenGC/Theory-of-Automatic-Control

2. Простой регулятор на базе нечеткой логики. Создание и настройка // Хабр URL: https://habr.com/ru/articles/413539/

3. Linear Quadratic Regulator (LQR) Control for the Inverted Pendulum on a Cart [Control Bootcamp] // Youtube URL: https://www.youtube.com/watch?v=1_UobILf3cc&t=3s

4. Nguyen Van Dong, Pham Quoc Thai, Phan Minh Duc, Nguyen Viet Thuan Estimation of Vehicle Dynamics States Using Luenberger Observer // International Journal of Mechanical Engineering and Robotics Research . - May 2019. - №Vol. 8, No. 3, p. 430-436.

5. G. Molnárka Design Luenberger Observer for an Electromechanical Actuator // Acta Technica Jaurinensis . - October 2014

6. Фильтр Калмана // Хабр URL: https://habr.com/ru/articles/166693/

7. Alex Becker Kalman Filter from the Ground up. - Second edition (URL: https://www.kalmanfilter.net/multiSummary.html)

8. kalman // MathWorks URL: https://www.mathworks.com/help/control/ref/ss.kalman.html?searchHighlight=kalman&s_tid=srchtitle_support_results_1_kalman#mw_bd4835a7-205f-4aab-b95e-5cf91285902b_head