Федеральное агентство по образованию РФ

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

УТВЕРЖДАЮ

Зав. кафедрой ИиУС
_____ А.В. Бушманов
«___» _____ 2010 г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ «МЕТОДЫ АНАЛИЗА ДИНАМИЧЕСКИХ СИСТЕМ»

для направления подготовки 230100.68
«Информатика и вычислительная техника»

Составитель: Шевко Д.Г.

2010

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

Шевко Д.Г. Учебно-методический комплекс дисциплины «Методы анализа динамических систем» для направления подготовки 230100.68 «Информатика и вычислительная техника». – Благовещенск: Амурский гос. ун-т, 2010.

Учебно-методическое пособие содержит: рабочую программу преподавания дисциплины; краткое изложение курса лекций; методические указания и учебные задания для выполнения курса лабораторных работ; карту обеспеченности дисциплины кадрами профессорско-преподавательского состава

РАБОЧАЯ ПРОГРАММА

по дисциплине «Методы анализа динамических систем»

для направления подготовки 230100.68 «Информатика и вычислительная техника»

Курс 7 Семестр D

Лекции 18 час. Экзамен

Практические занятия Зачет D семестр

Лабораторные занятия 54 час.

Самостоятельная работа 69 час.

Всего часов 141

Составитель: доцент Шевко Д.Г.

Факультет Математики и информатики

Кафедра Информационных и управляющих систем

1. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Наименование тем, их содержание, объем

Тема 1. МЕТОД СИНТЕЗА ГИБРИДНЫХ СИСТЕМ ПРЯМОГО АДАПТИВНОГО УПРАВЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ НЕЛИНЕЙНОГО ПРЕОБРАЗОВАНИЯ (5 час.)

- 1.1. Характеристика проблемы управления в условиях априорной неопределенности
- 1.2. Способы построения гибридных систем прямого адаптивного управления
- 1.3. Общая постановка задачи синтеза беспоисковых ГСПАУ
- 1.4. Критерий гиперустойчивости для нелинейно преобразованных гибридных систем
- 1.5. Основные этапы синтеза беспоисковых ГСПАУ на основе критерия гиперустойчивости и нелинейного преобразования

Тема 2. МАТЕМАТИЧЕСКОЕ И АЛГОРИТМИЧЕСКОЕ

ОБЕСПЕЧЕНИЯ ГИБРИДНЫХ НЕЛИНЕЙНО ПРЕОБРАЗОВАННЫХ СИСТЕМ ПРЯМОГО АДАПТИВНОГО УПРАВЛЕНИЯ С ЯВНОЙ ЭТАЛОННОЙ МОДЕЛЬЮ (6 час.)

2.1. Постановка задачи синтеза систем с явным эталоном
2.2. Синтез алгоритмов настройки для систем со скалярным управлением
2.3. Синтез алгоритмов настройки для систем с векторным управлением
2.4. Робастные алгоритмы настройки систем управления

Тема 3. АЛГОРИТМИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ГИБРИДНЫХ СИСТЕМ УПРАВЛЕНИЯ С НЕЯВНОЙ ЭТАЛОННОЙ МОДЕЛЬЮ (7 час.)

3.1. Постановка задачи синтеза систем с неявным эталоном
3.2. Синтез алгоритмов настройки для систем без запаздывания
3.3. Синтез алгоритмов настройки для систем с запаздыванием по состоянию
3.4. Робастные алгоритмы настройки для систем с неявным эталоном
3.5. Алгоритмы адаптивного управления неминимально-фазовыми объектами

Лабораторные работы

Лабораторная работа 1. ГСПАУ с ЯЭМ (n = 3, l = 3, m = 1).
Лабораторная работа 2. ГСПАУ с ЯЭМ (n = 3, l =2, m = 1)
Лабораторная работа 3. ГСПАУ с ЯЭМ (n = 3, l = 3, m = 3).
Лабораторная работа 4. ГСПАУ с ЯЭМ (n = 3, l = 3, m = 3).
Лабораторная работа 5. ГСПАУ с НЭМ (n = 3, l = 2, m = 2).
Лабораторная работа 6. ГСПАУ с НЭМ (n = 3, l = 2, m = 1).
Лабораторная работа 7. ГСПАУ с запаздыванием (n = 3, l = 2, m = 2).
Лабораторная работа 8. ГСПАУ с запаздыванием (n = 3, l = 2, m = 1).

Самостоятельная работа студентов

В качестве самостоятельной работы по дисциплине «Методы анализа динамических систем» студенты готовят рефераты по следующим темам:

1. Гибридные системы адаптивного управления
2. Характеристика проблемы управления в условиях априорной неопределенности
3. Способы построения гибридных систем адаптации
4. Формализация задачи синтеза адаптивных систем управления
5. Критерий гиперустойчивости и синтез нелинейно преобразованных гибридных систем прямого адаптивного управления

6. Пакет matlab и simulink-моделирование
7. Синтез и s-модели адаптивных систем
8. Гибридные системы со скалярным управлением
9. Гибридные системы с векторным управлением
10. SIMULINK-моделирование систем с эталонной моделью
11. Создание matlab-приложений
12. Нестационарно-нелинейные системы

Вопросы к зачету

1. МЕТОД СИНТЕЗА ГИБРИДНЫХ СИСТЕМ ПРЯМОГО АДАПТИВНОГО УПРАВЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ НЕЛИНЕЙНОГО ПРЕОБРАЗОВАНИЯ
   - 1.6. Характеристика проблемы управления в условиях априорной неопределенности
   - 1.7. Способы построения гибридных систем прямого адаптивного управления
   - 1.8. Общая постановка задачи синтеза беспоисковых ГСПАУ
   - 1.9. Критерий гиперустойчивости для нелинейно преобразованных гибридных систем
   - 1.10. Основные этапы синтеза беспоисковых ГСПАУ на основе критерия гиперустойчивости и нелинейного преобразования
2. МАТЕМАТИЧЕСКОЕ И АЛГОРИТМИЧЕСКОЕ ОБЕСПЕЧЕНИЯ ГИБРИДНЫХ НЕЛИНЕЙНО ПРЕОБРАЗОВАННЫХ СИСТЕМ ПРЯМОГО АДАПТИВНОГО УПРАВЛЕНИЯ С ЯВНОЙ ЭТАЛОННОЙ МОДЕЛЬЮ
   - 2.1. Постановка задачи синтеза систем с явным эталоном
   - 2.5. Синтез алгоритмов настройки для систем со скалярным управлением
   - 2.6. Синтез алгоритмов настройки для систем с векторным управлением
   - 2.7. Робастные алгоритмы настройки систем управления
3. АЛГОРИТМИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ГИБРИДНЫХ СИСТЕМ УПРАВЛЕНИЯ С НЕЯВНОЙ ЭТАЛОННОЙ МОДЕЛЬЮ
   - 3.6. Постановка задачи синтеза систем с неявным эталоном
   - 3.7. Синтез алгоритмов настройки для систем без запаздывания
   - 3.8. Синтез алгоритмов настройки для систем с запаздыванием по состоянию
   - 3.9. Робастные алгоритмы настройки для систем с неявным эталоном
   - 3.10. Алгоритмы адаптивного управления неминимально-

фазовыми объектами

Виды контроля

Для проверки эффективности преподавания дисциплины проводится контроль знаний студентов. При этом используются следующие виды контроля:

- *текущий контроль* за аудиторной и самостоятельной работой обучаемых осуществляется во время проведения аудиторных занятий посредством устного опроса, проведения контрольных работ;
- *промежуточный контроль* осуществляется два раза в семестр в виде анализа итоговых отчетов на аттестационные вопросы;
- *итоговый контроль* в виде зачета осуществляется после успешного прохождения студентами текущего и промежуточного контроля и сдачи отчета по самостоятельной работе.

Требования к знаниям студентов, предъявляемые на зачете

Для получения зачета студент должен посещать занятия, проявлять активность в аудитории, обязан выполнить все лабораторные работы, знать теоретический материал в объеме лекционного курса, защитить отчет по самостоятельной работе.

2. УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Основная литература

1. *Еремин Е.*Л., Самохвалова С.Г. Шевко Д.Г. Адаптивные системы управления с настройкой компенсаторов. – Благовещенск: Амурский гос. ун-т, 2006. – 155 с.

2. *Еремин Е.*Л., Галаган Т.А., Семичевская Н.П. Нелинейное робастное управление нестационарными объектами. – Благовещенск: Амурский гос. ун-т, 2006. – 185 с.

3. *Еремин Е.*Л., Теличенко Д.А., Чепак Л.В. Синтез адаптивных систем для скалярных объектов с запаздыванием по управлению. – Благовещенск: Амурский гос. ун-т, 2006.

Дополнительная литература

4. *Еремин Е.Л., Шевко Д.Г.* Синтез и упрощение технической реализации гибридной нелинейно преобразованной системы прямого адаптивного управления // Вычислительные технологии. 2003. Т. 8. № 3. С. 47-57.

5. *Куо* Б. Теория и проектирование цифровых систем управления. М.: Машиностроение, 1986.

6. Методы классической и современной теории автоматического управления. Т.2: Синтез регуляторов и теория оптимизации систем автоматического управления / Под ред. *Н.Д. Егупова*. М.: Изд-во МГТУ им. Н.Э. Баумана, 2000.

7. Методы классической и современной теории автоматического управления. Т.3: Анализ и статистическая динамика систем автоматического управления / Под ред. *Н.Д. Егупова*. М.: Изд-во МГТУ им. Н.Э. Баумана, 2000.

8. *Еремин Е.Л.* Динамические модели и S-моделирование систем. Благовещенск: Изд-во АмГУ, 2003.

9. *Черных И.В.* Simulink: Инструмент моделирования динамических систем. http://www.matlab.ru/simulink/book1/1.asp.

10. *Еремин Е.*Л., Еремина В.В., Семичевская Н.П., Шевко Д.Г. Алгоритмы и S-модели гибридных систем адаптивного управления (практикум в

среде SIMULINK). Благовещенск: Амурский гос. ун-т, 2005. 205 с.

3. УЧЕБНО-МЕТОДИЧЕСКАЯ (ТЕХНОЛОГИЧЕСКАЯ) КАРТА ДИСЦИПЛИНЫ

Номер недели	Номер темы	Лабораторные занятия	Практические занятия	Используемые наглядн. и метод. пособия	Самостоятельная Работа студентов		Форма контроля
					Содержание	Часы	
1	2	3	4	5	6	7	8
1	1.1	1	-	1–10	Выбор темы самостоятельной работы	2	
2	1.2	1	-	1–10			з.л.р.
3	1.3	2	-	1–10	Поиск литературы по теме самостоятельной работы	6	
4	1.4	2	-	1–10			собесед.
5	1.5	3	-	1–10	Работа с литературой и поиск информации в сети Интернет по новейшим достижениям в области Методов анализа динамических систем по теме самостоятельной работы	53	
6	2.1	3	-	1–10			з.л.р.
7	2.2	4	-	1–10			
8	2.2	4	-	1–10			к.р.
9	2.3	5	-	1–10			
10	2.3	5	-	1–10			з.л.р.
11	2.4	6	-	1–10			
12	3.1	6	-	1–10			собесед.
13	3.2	7	-	1–10			
14	3.2	7	-	1–10			з.л.р.
15	3.3	7	-	1–10	Подготовка отчета	6	
16	3.4	8	-	1–10			к.р.
17	3.5	8	-	1–10	Защита отчета по самостоятельной работе	2	
18	3.5	8	-	1–10			зачет

Условные обозначения:
к.р. – контрольная работа
собесед. – собеседование

з.л.р. – защита лабораторной работы

КРАТКИЙ КОНСПЕКТ ЛЕКЦИЙ

В первой лекции дается краткий обзор и анализ современного состояния проблемы, связанной с разработкой беспоисковых систем прямого адаптивного управления. Излагается общий подход к решению задачи синтеза алгоритмов адаптации беспоисковых ГСПАУ для непрерывных объектов, основанный на введении в процедуру синтеза адаптивных систем управления нелинейного преобразования фазовых координат и опирающийся на использование аппарата теории гиперустойчивости и положительности динамических систем.

Для этой цели используются результаты теории гиперустойчивых систем, полученные в работах В.М. Попова, В.А. Якубовича, развитые в работах И.Д. Ландау, А.М. Цыкунова и Е.Л. Еремина, а также метод нелинейного преобразования пространства состояния, разработанный Р.У. Брокеттом.

Рассматривается динамическая система, описываемая уравнениями

$$\begin{aligned}
&\frac{dx(t)}{dt} = \Phi\big(x(t), x(t-h), u(t), \xi\big) + f_{\xi}(t), \;\; y(t) = L^{T}(\xi)x(t), \\
&x(\theta) = \varphi(\theta), \;\; \theta \in [-h; 0], \\
&u_k = u_k(y_k, y_{k-s}, \chi_k, r_k), \\
&y_k = y(t_k), \;\; y_{k-s} = y(t_{k-s}) = y(t_k - h), \\
&u(t) = u_k \;\text{ при }\; t_k \le t < t_{k+1},
\end{aligned} \qquad (1)$$

где $x(t) \in R^n$ – вектор состояния объекта управления; $y(t) \in R^l$ – вектор выхода; $u(t) \in R^m$ – вектор управляющих воздействий; $f(t) \in R^n$ – вектор возмущающих воздействий; χ_k – матрица настраиваемых коэффициентов регулятора; $r_k \in R^m$ – вектор задающих воздействий; $t_k = k\tau$ – дискретный аналог времени; $\tau = const > 0$ – шаг дискретизации; $k = 0, 1, 2, \ldots$ – номер шага; $\Phi\big(x(t), x(t-h), u(t), \xi\big)$ – отображение, удовлетворяющее условиям существования и единственности решения уравнений (1) при заданной начальной вектор-функции $\varphi(\theta) \in C_h$; C_h – пространство непрерывных функций; $h = const \ge 0$ – запаздывание; $s = h/\tau$ – число тактов запаздывания; $L(\xi)$ – квазипостоянная матрица выхода соответствующего размера; $\xi \in \Xi$ – набор всех неизвестных параметров; Ξ – известное множество возможных значений ξ.

Требуется построить высокоэффективную адаптивную систему с эталонной моделью (АСЭМ) за счет применения метода нелинейного преобразования таким образом, чтобы при любом наборе $\xi \in \Xi$, при любых начальных условиях $x(0)$, χ_0 и возмущениях

$$\int_0^\infty \left\| f_\xi(t) \right\|^2 dt < \infty, \ \xi \in \Xi \tag{2}$$

выполнялось одно из следующих целевых условий (цель управления):

$$\begin{aligned} &\lim_{t\to\infty} (\bar{x}(t) - x(t)) = 0, \\ &\lim_{t\to\infty} (x_*(t) - x(t)) = 0, \end{aligned} \tag{3}$$

совместно с целевым условием (цель адаптации)

$$\lim_{k\to\infty} \chi_k = \chi_* = const, \tag{4}$$

где $\bar{x}(t)$ – вектор состояния явной эталонной модели; $x_*(t)$ – вектор состояния неявной ЭМ. Если же на объект управления действует неконтролируемое, ограниченное по норме возмущающее воздействие

$$\left\| f_\xi(t) \right\| \le f_0 = const, \ \xi \in \Xi, \tag{5}$$

то условия (3) заменяются на следующие:

$$\begin{aligned} &\lim_{t\to\infty} \| \bar{x}(t) - x(t) \| \le \delta_1, \ \delta_1 = const > 0, \\ &\lim_{t\to\infty} \| x_*(t) - x(t) \| \le \delta_2, \ \delta_2 = const > 0, \end{aligned} \tag{6}$$

а условие (4) – на предельное соотношение

$$\lim_{k\to\infty} \chi_k \le \chi_* = const. \tag{7}$$

Решение сформулированной задачи синтеза системы (1) включает следующие этапы: представление исходной системы в эквивалентной форме, а именно в виде линейного стационарного блока прямой цепи и нелинейного нестационарного блока обратной связи (1 этап); обеспечение строгой положительности передаточной матрицы линейной части системы (2 этап); построение дискретных алгоритмов настройки коэффициентов регулятора (3 этап); проверка выполнения целевых условий и свойств адаптивности системы (4 этап); упрощение технической реализации АСЭМ (5 этап).

При этом на втором и третьем этапах синтеза адаптивной системы управления выделяется непустой класс гиперустойчивых систем, допускающий существенное расширение за счет применения нелинейного преобразования элементов вектора состояния исходной системы. С этой целью используются результаты, полученные в рамках теории непрерывных групп Ли, позволяющие для системы

$$\frac{de(t)}{dt} = A(t)e(t), \ e(t) \in R^n \tag{8}$$

построить формы q-й степени, служащие элементами вектора $e^{[q]}(t)$ расши-

ренного пространства состояний $R^{N_n^q}$, $N_n^q = C_{n+q-1}^n = \dfrac{(n+q-1)!}{n!(q-1)!}$ и обеспечивающие выполнение условия

$$\left\| e^{[q]}(t) \right\| = \left\| e(t) \right\|^q = \left(\sqrt{e^T(t)e(t)} \right)^q, \ q = 2,3,... \tag{9}$$

при специально организованном базисе.

Рассмотренный подход используется для выделения расширенного класса гиперустойчивых систем с помощью модификации неравенства Попова, представленного следующим образом:

$$\eta(0,k_1) = -\sum_{k=0}^{k_1} \mu_k^T z_k \geq -\gamma_0^2 = const, \ \forall k_1 \geq 0, \tag{10}$$

где μ_k – преобразованный вектор управления; z_k – нелинейно преобразованный вектор выхода, явный вид которого уточняется на третьем этапе синтеза адаптивных систем управления.

Во второй лекции разрабатывается и исследуется метод синтеза беспоисковых ГСПАУ для непрерывных объектов с явной ЭМ, основанный на применении критерия гиперустойчивости.

В п.2.1 приведена постановка задачи синтеза ГСПАУ с явной эталонной моделью.

В п.2.2 осуществлялся синтез ГСПАУ со скалярным управлением. В частном случае исследовалась ГСПАУ с полным измерением вектора состояния объекта

$$\frac{dx(t)}{dt} = A(\xi)x(t) + b(\xi)u(t) + f_\xi(t), \tag{11}$$

$$\bar{x}_{k+1} = P_M \bar{x}_k + d_M r_k, \tag{12}$$

$$e_k = \bar{x}_k - x_k, \ z_k = g^T e_k \parallel e_k \parallel^q, \ x_k = x(t_k), \tag{13}$$

$$u_k = \chi_{1,k} r_k + \chi_{2,k}^T x_k, \ u(t) = u_k \text{ при } t_k \leq t < t_{k+1}, \tag{14}$$

$$\chi_{1,k} = \chi_{1,k-1} + h_1 z_k r_k, \ h_1 = const > 0, \tag{15}$$

$$\chi_{2,k} = \chi_{2,k-1} + H_2 z_k x_k, \ H_2 = diag\{h_{2i}\}, \ h_{2i} = const > 0, \ i = \overline{1,n}. \tag{16}$$

В п.2.3 решена задача синтеза алгоритмов настройки для систем с векторным управлением. Одна из синтезированных систем имела следующее математическое описание:

$$\frac{dx(t)}{dt} = A(\xi)x(t) + B(\xi)u(t) + f_\xi(t), \ y(t) = L^T x(t), \tag{17}$$

$$\bar{x}_{k+1} = P_M \bar{x}_k + D_M r_k, \ \bar{y}_k = L^T \bar{x}_k, \tag{18}$$

$$\nu_k = G^T(\bar{y}_k - y_k),\ y_k = y(t_k), \tag{19}$$

$$u_k = \chi_{1,k} r_k + \chi_{2,k} y_k,\ u(t) = u_k \text{ при } t_k \le t < t_{k+1}, \tag{20}$$

$$\chi_{1ji,k} = \chi_{1ji,k-1} + h_{1ji} r_{i,k} \nu_{j,k} \| \nu_k \|^q, \quad h_{1ji} = const > 0,\ i,j = \overline{1,m}, \tag{21}$$

$$\chi_{2ji,k} = \chi_{2ji,k-1} + h_{2ji} y_{i,k} \nu_{j,k} \| \nu_k \|^q, \quad h_{2ji} = const > 0,\ i = \overline{1,l},\ j = \overline{1,m}. \tag{22}$$

В п.2.4 исследовались случаи, когда на объект управления действовали внешние возмущения. Для всех ранее синтезированных адаптивных алгоритмов показано, что при помехах со свойствами (3) все системы управления сохраняли свою работоспособность в условиях априорной неопределенности. В тех же случаях, когда помехи являются постоянно действующими (6), возникала потребность в огрублении алгоритмов адаптации. Регуляризация алгоритмов достигалась за счет введения в контур адаптации дополнительных местных обратных связей.

В третьей лекции разрабатывается и исследуется метод синтеза гибридных адаптивных систем с неявной ЭМ для непрерывных объектов на основе применения критерия гиперустойчивости и метода нелинейного преобразования пространства состояния.

Метод построения беспоисковых ГСПАУ с неявной ЭМ опирается на результаты работ И.Д. Ландау, Р.У. Брокетта, А.Л. Фрадкова, В.А. Якубовича, А.М. Цыкунова и Е.Л. Еремина.

В п.3.1 приведена постановка задачи синтеза ГСПАУ с неявной эталонной моделью.

В п.3.2 осуществлялся синтез гибридных нелинейно преобразованных систем управления для априорно неопределенного непрерывного объекта с неполным измерением вектора состояния. Одна из синтезированных систем со скалярным управлением имела следующее математическое описание:

$$\frac{dx(t)}{dt} = A(\xi)x(t) + b(\xi)u(t) + f_\xi(t),\ y(t) = L^T(\xi)x(t), \tag{23}$$

$$\varepsilon_k = r_* - \nu_k,\ \nu_k = g^T y_k,\ y_k = y(t_k), \tag{24}$$

$$u_k = \chi_{1,k} r_* + \chi_{2,k} \nu_k,\ u(t) = u_k \text{ при } t_k \le t < t_{k+1}, \tag{25}$$

$$\chi_{1,k} = \chi_{1,k-1} + h_1 \varepsilon_k \| \varepsilon_k \|^q\ r_*,\ h_1 = const > 0, \tag{26}$$

$$\chi_{2,k} = \chi_{2,k-1} + h_2 \varepsilon_k \| \varepsilon_k \|^q\ \nu_k,\ h_2 = const > 0. \tag{27}$$

В п.3.3 решена задача синтеза ГСПАУ для объектов с запаздыванием по состоянию. В частном случае исследовалась ГСПАУ с основным контуром управления

$$\frac{dx(t)}{dt} = A(\xi)x(t) + \Gamma(\xi)x(t-h) + B(\xi)u(t) + f_{\xi}(t), \tag{28}$$

$$y(t) = L^T(\xi)x(t),\ x(\theta) = \varphi(\theta),\ \theta \in [-h;0], \tag{29}$$

$$\varepsilon_k = r_* - \nu_k,\ \nu_k = G^T y_k,\ y_k = y(t_k), \tag{30}$$

$$u_k = \chi_{1,k} r_* + \chi_{2,k}\nu_k + \chi_{3,k}\nu_{k-s},\ u(t) = u_k \text{ при } t_k \le t < t_{k+1}. \tag{31}$$

Алгоритмы настройки параметров адаптивного регулятора системы (28) – (31) были синтезированы следующим образом:

$$\chi_{1i,k} = \chi_{1i,k-1} + h_{1i}\varepsilon_{i,k} \parallel \varepsilon_k \parallel^q r_{i,*},\ h_{1i} = const > 0, \tag{32}$$

$$\chi_{2i,k} = \chi_{2i,k-1} + h_{2i}\varepsilon_{i,k} \parallel \varepsilon_k \parallel^q \nu_{i,k},\ h_{2i} = const > 0, \tag{33}$$

$$\chi_{3i,k} = \chi_{3i,k-1} + h_{3i}\varepsilon_{i,k} \parallel \varepsilon_k \parallel^q \nu_{i,k-s},\ h_{3i} = const > 0,\ i = \overline{1,m}. \tag{34}$$

В п.3.4 рассмотрены вопросы работоспособности синтезированных в п.3.2 и п.3.3 ГСПАУ в условиях действия помех, затухающих во времени или ограниченных по норме.

В п.3.5 исследовалась задача синтеза гибридной системы управления для априорно неопределенного неминимально-фазового объекта с адаптивным шунт-компенсатором (АШК) и дискретным регулятором минимальной структурной сложности

$$a(p,\xi)y(t) = b(p,\xi)u(t),\ p \overset{\Delta}{=} d/dt,$$
$$a(p,\xi) = p^n + a_{n-1}(\xi)p^{n-1} + \ldots + a_1(\xi)p + a_0(\xi), \tag{35}$$
$$b(p,\xi) = b_m(\xi)p^m + b_{m-1}(\xi)p^{m-1} + \ldots + b_1(\xi)p + b_0(\xi),$$

$$z_k = -x_k / T^2 + u_k / T,\ T = const > 0,$$
$$x_{k+1} = \exp(-\tau/T)x_k + T(1 - \exp(-\tau/T))u_k, \tag{36}$$

$$e_k = r_* - y_k - \alpha_k z_k,\ r_* = const > 0,\ y_k = y(t_k), \tag{37}$$

$$\alpha_{k+1} = \alpha_k + h_1 \parallel e_k \parallel^{q_1},\ h_1 = const > 0, \tag{38}$$

$$u_k = c_k r_*,\ u(t) = u_k \text{ при } t_k \le t < t_{k+1}, \tag{39}$$

$$c_k = c_{k-1} + h_2 e_k \parallel e_k \parallel^{q_2} r_*,\ h_2 = const > 0, \tag{40}$$

где $y(t) \in R$; $u(t) \in R$; $a(p,\xi)$ – гурвицев полином, $a_0 > 0$; p – оператор дифференцирования; $b(p,\xi)$ – полином с произвольным расположением корней, $b_0 > 0$; $z_k \in R$ – выход АШК; $\alpha_k \in R$ – параметр настройки АШК.

ЛАБОРАТОРНЫЕ РАБОТЫ

Приложения MatLab являются графическими окнами, содержащими элементы управления (кнопки, списки, переключатели, флаги, полосы скроллинга, области ввода, меню), а также оси и текстовые области для вывода результатов работы. Создание приложений включает следующие основные этапы – расположение нужных элементов интерфейса в пределах графического окна и определение действий (команд MatLab), которые выполняются при обращении пользователя к данным объектам, – например, при нажатии кнопки. Процесс работы над приложением допускает постепенное добавление элементов в графическое окно, запуск и тестирование приложения и возврат в режим редактирования. Конечным результатом является программа с графическим интерфейсом пользователя (GUI), содержащаяся в нескольких файлах, запуск ее производится указанием ее имени в командной строке MatLab или в другом приложении.

1. Графический редактор в среде GUIDE

Перейдите в среду GUIDE, для этого введите в командной строке MatLab команду guide. Появляется редактор окна приложения (рис. 1), заголовок которого untitled.fig[1] означает, что в нем открыт новый файл.

Редактор приложения содержит:

строку меню;

панель инструментов управления приложением;

заготовку окна приложения с нанесенной сеткой;

вертикальную и горизонтальную линейки;

панель инструментов для добавления элементов интерфейса на окно приложения.

[1] Здесь и далее используется руссифицированная версия MATLAB 6.5.0.

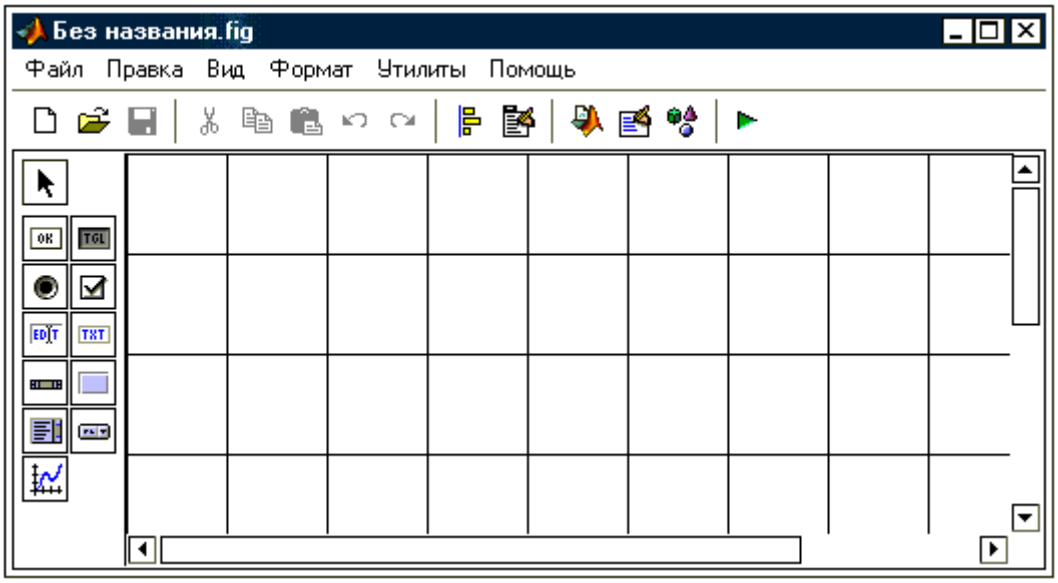

Рис. 1. Редактор приложения в версии 6.5.0.

Редактор приложения MatLab 6.x позволяет разместить различные элементы интерфейса. Для этого требуется нажать соответствующую кнопку на панели инструментов и поместить выбранный объект щелчком мыши в требуемое место заготовки окна приложения. Другой способ состоит в задании прямоугольной области объекта перемещением мыши по области заготовки окна с удержанием левой кнопки. Размер и положение добавленных объектов изменяются при помощи мыши. Перед изменением размера следует выбрать режим выделения объектов и сделать его текущим, щелкнув по нему клавишей мыши.

Любой объект можно удалить из окна при помощи <Delete>, предварительно выделив его. Запуск приложения производится кнопкой Activate Figure. Появляется диалоговое окно GUIDE, которое сообщает о необходимости сохранить приложение. Нажмите Yes и сохраните приложение в файле с расширением fig.

Приложение запускается в отдельном окне с заголовком Untitled. Пользователь может нажимать на кнопки, устанавливать флаги, переключатели, обращаться к спискам. Разумеется, при этом ничего полезного не происходит. Более того, при нажатии на кнопки в командное окно MatLab выводится сообщение следующего содержания:

Pushbutton1 Callback not implemented yet.

Данное сообщение говорит о том, что обработка события, которое происходит при нажатии на кнопку, не запрограммировано. Недостаточно разместить элементы интерфейса в окне приложения, следует позаботиться о том, чтобы каждый элемент выполнял нуж-

ные функции при обращении к нему пользователя. Например, при нажатии на кнопку производятся вычисления и строятся графики полученных результатов, переключатели позволяют установить цвет линий, полоса скроллинга изменяет толщину линии, в области ввода пользователь указывает некоторые параметры, управляющие ходом вычислений.

Программирование событий в версии 6.x. Рассмотрим принцип программирования событий в версии 6.x. Приложение MatLab 6.x хранится в двух файлах с расширением fig и m, первый из них содержит информацию о размещенных в окне приложения объектах, а второй является m-файлом с основной функцией и подфункциями. Добавление элемента интерфейса из редактора приложения приводит к автоматическому созданию соответствующей подфункции. Данную подфункцию следует наполнить содержимым – операторами, которые выполняют обработку события, возникающего при обращении пользователя к элементу интерфейса.

Начните с создания простого приложения, окно которого содержит оси и две кнопки, предназначенные для построения графика и очистки осей.

Перейдите в среду приложения командой guide. Выберите в меню Tools редактора приложений пункт Application Options, появится диалоговое окно GUIDE Application Options. Данное окно позволяет устанавливать некоторые общие свойства создаваемого прило-жения. Выберите в раскрывающемся списке Command-line accessibility строку On (recommended for plotting windows) и нажмите OK. Приложение, выводящее графики на оси, которые расположены в пределах окна приложения, требуют установки этой опции для корректной работы.

Расположите на форме оси кнопку. На кнопке автоматически размещается надпись Push Button. Следующий этап очень важный. Кнопка является элементом интерфейса, ей следует дать имя, которое уникальным образом идентифицировало бы ее среди всех объектов приложения.

Выделите кнопку Push Button и вызовите при помощи панели инструментов управления приложением редактор свойств Property Inspector. Появляется окно редактора свойств, в котором содержится таблица названий свойств кнопки и их значений. Занесите в свойство Tag значение btnPlot, щелкните мышью по строке справа от названия свойства, наберите требуемое значение и нажмите <En-

ter>. В дальнейшем будет говориться, что некоторому объекту или элементу управления следует дать имя. Например, btnPlot теперь является именем кнопки Push Button. Удобно задавать имена, часть которых определяет тип элемента управления (btn соответствует button – кнопке). Аналогичным образом дайте осям имя axMain.

Выберите в меню File редактора приложения пункт Save as, создайте папку MyFirstGui и сохраните приложение в файле mygui.fig. Обратите внимание, что открылся редактор М-файлов, содержащий файл mygui.m.

Приложение mygui содержит одну кнопку Press Button. Когда пользователь нажимает на Press Button в работающем приложении, то происходит событие Callback данного элемента управления. Вызывается подфункция btnPlot_Callback. Сейчас она содержит оператор вывода в командное окно текста с предупреждением о том, что событие Callback пока не определено. Имя подфункции образовано названием кнопки и события. Очень важно задавать имена объектам в свойстве Tag сразу после их добавления в окно приложения в редакторе приложений, иначе генерируемая подфункция получит имя, которое сохранится при последующем изменении значения Tag и повлечет ошибки при выполнении приложения.

Завершающий этап состоит в программировании действий, которые выполняются при нажатии пользователем на кнопку Press Button. Измените функцию обработки события нажатием на Press Button в соответствии с листингом 1.

Листинг 1. Обработка события Callback кнопки с именем btnPlot.

```
function varargout = btnPlot_Callback (h, eventdata, handles, varargin)
x = [-2:0.2:2];
y = exp(-x.^2);
plot(x,y)
```

Сохраните файл mygui.m в редакторе М-файлов и запустите приложение из редактора приложений, нажав кнопку Activate Figure. Нажатие на Press Button в запущенном приложении приводит к отображению графика функции на осях. Закройте окно приложения при помощи кнопки с крестиком в правом верхнем углу и продолжите работу над mygui в редакторе приложений.

Добавьте кнопку, задайте ей имя btnClear в редакторе свойств. Быстрый доступ к свойствам выделенного объекта в редакторе приложений производится из пункта Inspect Properties всплывающего

меню при нажатии правой кнопки мыши на объекте. Перейдите в подфункции обработки события Callback добавленной кнопки, для чего следует выбрать пункт Edit Callback всплывающего меню. Выбор данного пункта делает активным редактор М-файлов и выделяет заголовок соответствующей подфункции btnClear_Callback. Разместите единственный оператор очистки cla в подфункции (листинг 2).

Листинг 2. Обработка события кнопки с именем btnClear.

```
function varargout = btnClear_Callback (h, eventdata, handles, varargin)
cla
```

Запустите приложение и убедитесь, что нажатие на левую кнопку приводит к отображению графика функции, а правая служит для очистки осей.

Следует позаботиться о том, чтобы интерфейс приложения был очевиден для пользователя.

Конструирование интерфейса в версии 6.x.

Управления свойствами объектов. Разработка приложения сопряжена с изменением свойств объектов, которые они получают по умолчанию при размещении их на заголовке окна. Некоторые из свойств, например, надпись на кнопке или ее размер, устанавливаются при создании объекта в режиме редактирования. Другие свойства могут изменяться программно в работающем приложении.

Установка свойств при редактировании. Продолжите работу над приложением mygui. Очевидно, что следует надписать кнопки, – например, Построить и Очистить. Кнопки являются графическими объектами с определенными свойствами, среди которых имеется свойство, отвечающее за надпись на кнопке. Сделайте левую кнопку приложения mygui текущей и вызовите редактор свойств Property Inspector. Установите свойство String левой кнопки в значение Построить. Важно понять, что значение свойства String соответствует надписи на кнопке, а Tag – имени или тегу кнопки как объекта. Имена объектов используются для изменения их свойств в ходе работы приложения при выполнении блоков обработки событий от других элементов интерфейса. Перейдите теперь к свойствам правой кнопки и установите String в Очистить.

Следует отметить, что доступ к редактору свойств выделенного объекта производится либо из панели инструментов управления приложением, либо из меню Tools редактора приложений, либо при

помощи пункта Inspect Properties всплывающего меню.

Значение свойства String сразу отображается на кнопке приложения, находящегося в режиме редактирования. Запустите приложение, кнопки теперь снабжены надписями, которые определяются соответствующими значениями свойств String. Аналогичным образом устанавливаются и другие свойства объектов.

Программное изменение свойств. Большинство свойств объектов можно устанавливать программно прямо в ходе работы приложения для обеспечения согласованного поведения элементов управления. Усовершенствуйте приложение mygui следующим образом. Пусть при запуске доступной является только кнопка Построить. При нажатии на кнопку Построить выводится график, и она становится недоступной, зато пользователь может нажать кнопку Очистить для очистки осей, и наоборот. Итак, всегда доступна только одна из кнопок, в зависимости от состояния осей.

Решение поставленной задачи требует привлечения свойства Enable. Это свойство объекта отвечает за возможность доступа к нему пользователя, значение on разрешает доступ, a off, соответственно, запрещает. Установка значений свойствам объектов в программе производится при помощи функции set.

Функция set вызывается с тремя входными аргументами – указателем на объект, названием свойства и его значением, последние два аргумента заключаются в апострофы. Заметьте, что свойства одного объекта должны изменяться в блоке операторов обработки события Callback другого объекта. Следовательно, должна иметься возможность доступа к указателю на любой существующий объект. Аргументы h и handles подфункций обрабатывают события элементов управления, содержат требуемые указатели. В h хранится указатель на тот объект, событие которого обрабатывается в данный момент, a handles является структурой указателей. Поля структуры совпадают со значениями свойств Tag существующих элементов интерфейса. Например, handles.btnPlot является указателем на кнопку Построить с именем btnPlot.

Доступ к Очистить должен быть запрещен в начале работы приложения, пока пользователь не нажмет Построить. Установите в редакторе свойств для кнопки Очистить Enable в Off, используйте кнопку со стрелкой в строке со значением свойства. Остальные изменения значения Enable кнопок должны происходить в ходе работы приложения. Для разрешения и запрещения доступа к кнопкам

нужно внести дополнения в обработку их событий Callback.

В подфункцию обработки события Callback кнопки Построить добавьте при помощи редактора вызовов:

установку свойства Enable кнопки Очистить в значение on (после вывода графика следует разрешить доступ к Очистить);

установку свойства Enable кнопки Построить в значение off (после вывода графика следует запретить доступ к Построить).

Аналогичные изменения произведите в обработке события Callback кнопки Очистить, а именно:

установку свойства Enable кнопки Построить в значение on (после вывода графика следует разрешить доступ к Построить);

установку свойства Enable кнопки Очистить в значение off (после вывода графика следует запретить доступ к кнопке).

Подфункции btnPlot_Callback и btnClear_Callback должны быть запрограммированы так, как показано на листинге:

```
function varargout = btnPlot_Callback (h, evendata, handles, varargin)
% Построение графика функции
x = [-2:0.2:2];
y = exp(-x.^2);
plot (x,y)
% Кнопка Построить должна стать недоступной после вывода
% графика
set (h, ‘Enable’, ‘off’)
% Кнопка Очистить должна быть доступной
set (handles.btnClear, ‘Enable’, ‘on’)
function varargout = btnClear_Callback (h, evendata, handles, varargin)
cla % очистка осей
% Кнопка Очистить должна стать недоступной после очистки осей
set (h, ‘Enable’, ‘off’)
Кнопка Построить должна быть доступной
set (handles.btnPlot, ‘Enable’, ‘on’)
```

Сохраните изменения в редакторе m-файлов. Запустите приложение mygui и убедитесь, что всегда доступной является только одна из кнопок – Построить или Очистить, что является хорошей подсказкой для пользователя о возможных действиях. Закройте окно приложения и редактор приложений. Следующий раздел посвящен запуску приложения из командной строки и переходу в режим редактирования.

Работа над приложением. Запуск приложения осуществляется не только из редактора приложений. Возникают вопросы: как работать с уже созданным приложением с графическим интерфейсом и вносить в него требуемые изменения? Для запуска приложения достаточно в качестве команды задать его имя в командной строке.

Появляется окно приложения, обращение к элементам интерфейса окна приводит к соответствующим действиям.

Заметим, что каталог с приложением должен содержаться в путях поиска MatLab или являться текущим.

Очень часто сразу не удается написать законченное приложение, необходимое усовершенствование появляется только в ходе работы с приложением. В любой момент можно продолжить редактирование двумя способами:

задать guide в командной строке, что приводит к отображению редактора приложений с заготовкой для нового приложения untitled.fig, и открыть соответствующий файл с расширением fig, выбрав пункт Open… в меню File редактора (закройте ненужный редактор приложений с untitled.fig);

указать имя приложения через пробел после команды guide, – например, guide mygui появляется редактор приложений, в котором открыто mygui.

Перейдите в режим редактирования приложения mygui любым из перечисленных способов и продолжите работу над ним. Измените название окна приложения на «Визуализация функций».

Заголовок окна задается свойством Name графического окна. Сделайте графическое окно текущим, щелкнув мышью по области заготовки окна в редакторе приложений. Установите в редакторе свойств Name в «Визуализация функций». Запустите mygui, появляется окно GUIDE с предупреждением, что продолжение повлечет сохранение приложения. Установите флаг Don’t tell me again для подавления данного предупреждения при дальнейшей работе и нажмите OK. Убедитесь, что приложение имеет нужный заголовок.

Обобщая изложенные выше сведения, можно сказать, что процесс программирования приложения с графическим интерфейсом включает этапы:

размещение элементов интерфейса в окне приложения;

программирование событий данных элементов в подфункциях m-файла приложения;

сохранение приложения;

запуск и работа над приложением;

переход в режим редактирования для дальнейшего усовершенствования приложения.

Желательно располагать элементы интерфейса в порядке, обеспечивающем удобную работу пользователя с приложением. Приложение хорошо выглядит, если однотипные элементы, – например, кнопки, флаги и т.д., определенным образом выровнены в окне приложения. MatLab 6.x предоставляет разработчику приложений простые способы выравнивания добавляемых объектов – сетку и линейку.

Оформление интерфейса. Редактор приложений содержит заготовку окна приложения с нанесенной сеткой, а также и вертикальную и горизонтальную линейки. Линейка позволяет размещать элементы интерфейса в позиции с любыми координатами в пикселях, отсчитываемыми от левого нижнего угла заготовки окна приложения. Следует выделить объект щелчком мыши и перемещать его по окну, следя за указателями его положения на линейке. Происходит непрерывное движение объекта, что не всегда удобно.

Часто требуется, чтобы небольшое перемещение мыши вызывало изменение положения объекта на некоторый фиксированный шаг. Сетка редактора приложений позволяет осуществить такое дискретное движение. Выбор пункта Grid and Rules меню Layout приводит к появлению диалогового окна Grid and Rules.

Флаги Show rulers и Show grid соответствуют отображению линеек и сетки в редакторе приложений, а раскрывающийся список Grid Size позволяет выбрать размер ячеек сетки. Минимально допустимый размер десять пикселей позволяет достаточно точно располагать элементы управления в окне приложений. Привязка перемещения к линиям сетки происходит при установленном флаге Snap to grid. Привязка разрешает разместить объект и измерить его размеры только при условии прохождения границы объекта по линиям сетки. Выбор мелкого шага сетки в сочетании с привязкой предоставляет разработчику возможность быстро оформить приложение. Плавно изменять положение выделенного объекта можно при помощи клавиш со стрелками. Одновременное удержание <Ctrl> приводит к перемещению с учетом привязки к сетке.

Размещение объектов в ряд по вертикали или горизонтали требует предварительного определения некоторой вспомогательной

линии. Можно, конечно, использовать линии сетки, но проще добавить горизонтальные или вертикальные линии выравнивания. Следует навести курсор мыши на соответствующую линейку (курсор меняет форму на двустороннюю стрелку), и движение объектов к линии выравнивания вызывает автоматическое расположение их границ на данной линии. Сами линии выравнивания можно убирать с окна приложения, перетаскивая их мышью обратно на линейку. Snap Show guides окна Grid and Rules отвечает за отображение линий выравнивания в области окна приложения.

Отметим, что привязка действует даже при выключенных флагах Show rulers, Show guides и Show grid.

Размер окна приложения изменяется при помощи перетаскивания мышью квадратика, расположенного в правом нижнем углу заготовки окна приложения.

Отредактируйте вид приложения mygui, используя описанные возможности. Следующие разделы посвящены размещению и программированию основных элементов интерфейса.

Программирование элементов интерфейса.

Флаги и рамки. Флаги позволяют произвести одну или несколько установок, определяющих ход работы приложения. Продолжите работу над mygui, предоставьте пользователю возможность наносить линии сетки на график. Окно приложения должно содержать два флага с названиями Сетка по x и Сетка по y. Если пользователь нажимает кнопку Построить, то на оси наносится сетка по выданным координатам. Нажатие на Очистить должно приводить не только к исчезновению графика функции, но и скрытию сетки.

Обычно несколько элементов управления со схожими назначениями группируются и помещаются внутри рамки. Измените размеры осей, освободив справа место для рамки. Нанесите рамку на окно приложения при помощи соответствующей кнопки. В рамку добавьте два флага.

Разместите поясняющие подписи рядом с флагами и дайте им имена. Задайте свойству Tag верхнего флага значение chbxGridX, а свойству String, отвечающему за подпись флага, значение Сетка по x. Аналогичным образом определите свойства нижнего флага, установите свойство Tag в chbxGridУ, String – в Сетка по y. Если текст не помещается рядом с флагом, увеличьте ширину области флага при помощи мыши, удерживая нажатой левую кнопку. Со-

храните приложение в редакторе приложений для автоматического создания в редакторе m-файлов заготовок для подфункций обработки событий добавленных объектов.

Осталось сделать так, чтобы при нажатии пользователем кнопки Построить происходило отображение линий сетки в зависимости от установленных флагов, а нажатие на Очистить приводило к скрытию сетки. Блок обработки события Callback кнопки Построить следует дополнить проверкой состояния флагов. Свойство флага Value принимает значение логической единицы при включении флага пользователем и, соответственно, равно нулю, если флаг выключен. Указатели на флаги содержатся на полях chbxGridX и chbxGridУ структуры handles.

Состояние флагов определяет значение свойств Xgrid и Ygrid осей.

Произведите необходимые изменения в подфункции обработки события Callback кнопки Построить с именем btnPlot.

Листинг. Обработка события кнопки btnPlot с учетом состояния флагов.

```
function varargout = btnPlot_Callback (h, eventdata, handles, varargin)
% Построение графика функции
x = [-2:0.2:2];
y = exp(-x.^2);
plot (x,y)
% проверка флага Сетка по x
if get (handles.chbxGridX, ‘Value’)
% Флаг включен, следует добавить линии по сетки
set (gca, ‘Xgrid’, ‘on’)
else
% Флаг выключен, следует убрать линии сетки
set (gca, ‘Xgrid’, ‘off’)
% проверка флага Сетка по y
if get (handles.chbxGridY, ‘Value’)
% Флаг включен, следует добавить линии по сетки
set (gca, ‘Ygrid’, ‘on’)
else
%Флаг выключен, следует убрать линии сетки
set (gca, ‘Ygrid’, ‘off’)
end
% Кнопка Построить должна стать недоступной после вывода гра-
```

фика
set (h, ‘Enable’, ‘off’)
% Кнопка Очистить должна стать доступной
set (handles.btnClear, ‘Enable’, ‘on’)

Запустите приложение mygui и убедитесь, что установка флагов влияет на отображение сетки при нажатии на кнопку Построить.

Смена состояния флагов сетки не приводит к немедленным изменениям на графике. Пользователь должен перестроить график, нажимая последовательно кнопки Очистить и Построить. Для немедленного реагирования приложения на состояние флагов следует определить их события Callback. Программирование данных событий заключается в проверке состояния флага и отображении или скрытии соответствующих линий сетки.

Сделайте текущим флаг Сетка по x в редакторе приложений и выберите опцию Edit Callback во всплывающем меню данного объекта. Запрограммируйте событие Callback флага. Используйте аргумент h соответствующих подфункций, содержащий указатель на объект, событие которого обрабатывается в текущий момент времени. Аналогичным образом обработайте событие Callback второго флага Сетка по y.

Листинг. Обработка событий Callback флагов сетки.

```
function varargout = cnbxGridX_Callback (h, eventdata, handles, varargin)
% Проверка флага Сетка по x
if get (h, ‘Value’)
% Флаг включен, следует добавить линии сетки
set (gca, 'Xgrid', ‘on’)
else
% Флаг выключен, следует убрать линии сетки
set (gca, ‘Xgrid’, ‘off’)
end
function varargout = cnbxGridY_Callback (h, eventdata, handles, varargin)
% Проверка флага Сетка по x
if get (h, ‘Value’)
% Флаг включен, следует добавить линии сетки
set (gca, 'Ygrid', ‘on’)
else
% Флаг выключен, следует убрать линии сетки
set (gca, ‘Ygrid’, ‘off’)
```

end

Внесенные дополнения позволяют пользователю наносить и убирать сетку по каждой координате при помощи флагов, без перестроения графика функции.

Флаги предоставляют пользователю возможность выбора одной или сразу нескольких опций. Выбор только одной опции осуществляется при помощи переключателей.

Переключатели. Переключатели обычно группируются по их предназначению, и пользователь может выбрать только одну опцию. Всегда установлен единственный переключатель из группы. Обработка событий переключателя должна влиять на состояние остальных переключателей группы. Модернизируйте интерфейс приложения mygui, предоставьте пользователю возможность выбирать тип маркера (кружок, квадрат ли отсутствие маркера).

Добавьте в окно приложения новую рамку и нанесите на нее три переключателя, установите свойствам Tag значения rbMarkCirc, rbMarkSq, rbMarkNone, a String – соответственно маркеры-круги, маркеры-квадраты, без маркеров.

Состояние переключателя определяется его свойством Value. Если Value равно единице, то переключатель включен, ноль – нет. Задайте в редакторе свойств значение единицы свойству Value переключателя с надписью «без маркеров», он будет включен при запуске программы. Значение свойства Value в версии MatLab 6.x устанавливается следующим образом. Выделите переключатель и перейдите к его свойствам. В редакторе свойств нажмите кнопку в строке с Value. Появляется окно Value.

Выделите при помощи мыши строку со значением 0.0 и перейдите в режим редактирования значения двойным щелчком мыши. Измените 0.0 на единицу и нажмите ОК. Обратите внимание, что в редакторе свойств значение Value изменилось на единицу и включился переключатель без маркеров на окне приложения в редакторе приложений. Описанным способом устанавливаются значения Value в редакторе свойств. Дальнейшее управление значением Value переключателей должно осуществляться программно в ходе работы приложения mygui.

Предположим, что пользователь установил один из переключателей. Происходит обращение к соответствующей подфункции обработки события Callback переключателя, которая обеспечивает:

изменение типа маркеров линии;

выключение двух остальных переключателей.

Второе действие программируется достаточно просто – следует занести в Value значение ноль при помощи функции set и указателей на нужные переключатели, хранящихся в структуре handles. Изменение типа маркеров линии не представляет труда: если известен указатель на линию, то достаточно обратиться к свойству линии Marker. Указатель на линию возвращает set в выходном аргументе, его следует записать в некоторую переменную, – например, Hline. Использовать указатель на линию придется в других подфункциях, обрабатывающих событие Callback переключателей. Очень важно понять, что переменная Hline инициализируется при вызове подфункции btnPlot_Callback, выполняются операторы подфункции, и она заканчивает работу. Все переменные, определенные в подфункции, являются локальными и по окончании работы подфункции не определены.

Обмен данными между подфункциями проще всего осуществить при помощи структуры handles. Подфункция, передающая данные, должна содержать запись данных в новое поле и сохранение структуры командой guidata. Входной аргумент – структура handles всех подфункций теперь содержит обновленное поле, в которое занесено соответствующее значение. Например, в некоторой подфункции можно сохранить массив в поле dat1 структуры handles

```
handles.dat1 = [1.2 3.2 0.1];
guidata (gcbo, handles)
```

Будем использовать его в другой подфункции

```
(handles.dat1)
```

аналогичным образом сохраняются указатели на объекты, создаваемые в разделах подфункции. Внесите необходимые изменения в подфункцию btnPlot_Callback и запрограммируйте обработку событий callback переключателей в подфункциях rbMarkCirc_Callback, rbMarkSq_Callback, rbMarkNone_Callback:

сохраните указатель на линию в поле Line структуры handles при построении графика командой plot;

добавьте блоки обработки событий Callback переключателей, каждый из которых устанавливает остальные два переключателя в положение «выключено» и наносит на линию соответствующие маркеры.

Листинг. Обработка событий переключателя в myguiprog.

```
function varargout = btnPlot_Callback (h, eventdata, handles, varargin)
% Построение графика функции
x = [-2:0.2:2];
y = exp(-x.^2);
% Запись указателя в поле line структуры handles
handles.line = plot(x,y);
% Сохранение структуры handles для использования
% в других подфункциях
guidata (gcbo, handles);
…
function varargout = rbMarkCirc_Callback (h, eventdata, handles, varargin)
% выбран переключатель маркеры-круги
set (handles.line, 'Marker', 'o') % размещение маркеров-кругов
% на линии
% Установка остальных переключателей в положение выключено
set (handles.rbMarkSq, 'Value', 0)
set (handles.rbMarkNone, 'Value', 0)

function varargout = rbMarkSq_Callback (h, eventdata, handles, varargin)
% выбран переключатель маркеры-квадраты
set (handles.line, 'Marker', 's') % размещение маркеров-квадратов
% на линии
% Установка остальных переключателей в положение выключено
set (handles.rbMarkCirc, 'Value', 0)
set (handles.rbMarkNone, 'Value', 0)

function varargout = rbMarkNone_Callback (h, eventdata, handles, varargin)
% выбран переключатель без маркеров
set (handles.line, 'Marker', 'none') % удаление маркеров с линии
% Установка остальных переключателей в положение выключено
set (handles.rbMarkCirc, 'Value', 0)
set (handles.rbMarkSq, 'Value', 0)
```

Запустите приложение mygui, отобразите график функции, нажав кнопку Построить, и убедитесь в том, что возможна установка только одного из переключателей и она приводит к появлению соответствующих маркеров на графике функции. Однако пока еще интерфейс mygui имеет ряд недостатков.

Если пользователь использует кнопку Очистить для очистки осей, а затем устанавливает переключатель, то производится обра-

щение к несуществующему объекту линии (сообщение об ошибке выводится в командном меню).

Нажатие на кнопку Построить приводит к получению линии без маркеров вне зависимости от установленного переключателя.

Повторный щелчок по области переключателя вызывает его выключение, но всегда один из переключателей должен быть установлен.

Конечно, первый недостаток является существенным – приложение должно работать без ошибок! Проще всего запретить доступ к переключателям, если нет линии на графике, и разрешить после ее появления. Очевидно, что следует внести изменения в соответствующие подфункции myguiprog, обрабатывающие события Ca;;back кнопок. Нажатие на кнопку Построить должно открыть доступ к группе переключателей, а очистка осей кнопкой Очистить – запрещать доступ. Итак, следует найти указатели на переключатели и установить их свойство Enabled в нужное значение on или off. При запуске приложения все переключатели должны быть недоступны, т.к. пользователь не построил график функции.

Редактор свойств позволяет одновременно установить значение общих свойств (например, Enable) целой группы объектов. Выделите щелчком мыши один из переключателей, остальные добавляйте в группу щелчком мыши, удерживая клавишу <Ctrl>. В результате должны быть выделены все три переключателя. Теперь перейдите в редактор свойств при помощи всплывающего меню. Вверху окна редактора размещена надпись Multiply objects selected, означающая, что проделываемые установки произойдут для свойств сразу всех выделенных объектов. Установите Enable off, при запуске приложения mygui переключатели недоступны. Осталось дополнить подфункции обработки события Callback кнопок Построить и Очистить для программного управления свойством Enable. Обратитесь к листингу, содержащему требуемые операторы.

Листинг. Разрешение и запрещение доступа к группе переключателей.

```
function varargout = btnPlot_Callback (h, eventdata, handles, varargin)
% Все переключатели должны стать доступными после
% появления графика
set (handles.rbMarkCirc, ‘Enable’, ‘on’)
set (handles.rbMarkSq, ‘Enable’, ‘on’)
```

```
set (handles.rbMarkNone, 'Enable', 'on')
%-------------------------------------------------------------------------
--
function varargout = btnClear_Callback (h, eventdata, handles, varargin)
% Все переключатели должны стать недоступными после
% очистки осей
set (handles.rbMarkCirc, 'Enable', 'off')
set (handles.rbMarkSq, 'Enable', 'off')
set (handles.rbMarkNone, 'Enable', 'off')
```

Первый недостаток интерфейса приложения mygui устранен. Теперь необходимо сделать так, чтобы при построении графика тип маркера отвечал установленному переключателю. Очевидно, что после вывода графика следует найти переключатель со значением Value, равным единице, и установить соответствующий тип маркера, согласно листингу.

Листинг. Изменение маркеров при построении графика.

```
function varargout = btnPlot_Callback (h, eventdata, handles, varargin)
…
if get (handles.rbMarkCirc, 'Value')
% Установлен переключатель маркеры-круги
set (handles.line, 'Marker', 'o')
end
if get (handles.rbMarkSq, 'Value')
% Установлен переключатель маркеры-квадраты
set (handles.line, 'Marker', 's')
end
```

Сохраните изменения в М-файле и запустите приложение mygui. Тип маркеров определяется установленным переключателем при построении графика.

Осталось нерешенной одна проблема. При повторном щелчке по области переключателя он выключается, но всегда должен быть установлен единственный переключатель. Данный недостаток устраняется с привлечением еще одного возможного значения inactive свойства Enable. Переключатель со значением inactive является неактивным, он выглядит в работающем приложении как доступный переключатель (со значением on), но попытка изменить состояние данного переключателя не приводит к успеху. Усовершенствуйте обработку событий согласно следующему алгоритму.

Свойство Enable переключателя, событие которого обрабатывается, должно иметь значение inactive, а для остальных двух – on. Если не задать on для других переключателей, то в результате все они станут неактивными.

При нажатии на кнопку Построить свойству Enable всех переключателей присваивается значение on, а затем определяется установленный в данный момент переключатель и в Enable заносится inactive.

Выполните подфункции btnPlot_Callback, rbMarkCirc_Callback, rbMarkSq_Callback и rbMarkNone_Callback необходимыми операторами, согласно листингу.

Листинг.

```
function varargout = btnPlot_Callback (h, eventdata, handles, varargin)
…
% Поиск установленного переключателя и определение его,
% как неактивного
if get (handles.rbMarkCirc, ‘Value’)
set (handles.rbMarkCirc, ‘Enable’, ‘inactive’)
end
if get (handles.rbMarkSq, ‘Value’)
set (handles.rbMarkSq, ‘Enable’, ‘inactive’)
end
if get (handles.rbMarkNone, ‘Value’)
set (handles.rbMarkNone, ‘Enable’, ‘inactive’)
end

function varargout = rbMarkCirc_Callback (h, eventdata, handles, varargin)

% Переключатель, событие которого обрабатывается, должен стать
% неактивным, а остальные – активными
set (h, ‘Enable’, ‘inactive’)
set (handles.rbMarkSq, ‘Enable’,’on’)
set (handles.rbMarkNone, ‘Enable’, ‘on’)

function varargout = rbMarkSq_Callback (h, eventdata, handles, varargin)

% Переключатель, событие которого обрабатывается, должен стать
% неактивным, а остальные – активными
set (h, ‘Enable’, ‘inactive’)
set (handles.rbMarkCirc, ‘Enable’,’on’)
set (handles.rbMarkNone, ‘Enable’, ‘on’)
```

```
function varargout = rbMarkNone_Callback (h, eventdata, handles, varargin)

% Переключатель, событие которого обрабатывается, должен стать
% неактивным, а остальные – активными
set (h, 'Enable', 'inactive')
set (handles.rbMarkSq, 'Enable','on')
set (handles.rbMarkCirc, 'Enable', 'on')
```

Правильная обработка переключателей (см. листинги 12.4–12.7) требует учета всех ситуаций, которые могут возникнуть при взаимодействии пользователя с приложением. Раскрывающиеся списки реализуют альтернативный способ выбора пользователем только одной из прилагаемых опций.

Списки. Модернизируйте интерфейс приложения mygui, предоставьте пользователю возможность выбрать цвет линии графика из раскрывающегося списка (синий, красный, зеленый). Перейдите в режим редактирования и добавьте при помощи панели управления раскрывающийся список. В редакторе свойств установите свойство Tag в значение 'pmColor'.

Элементами раскрывающегося списка являются строки, которые вводятся в редакторе свойств. Нажмите кнопку в строке со свойством String раскрывающегося списка, появляется окно String. Наберите в нем строки 'синий', 'красный', 'зеленый' (без кавычек), разделяя их при помощи клавиши <Enter>.

Запустите mygui и убедитесь, что раскрывающийся список содержит требуемые строки. Выбор различных строк пока не приводит к изменению цвета линии – требуется запрограммировать событие Callback раскрывающегося списка.

Обработка события Callback раскрывающегося списка состоит в определении выбора пользователя и соответствующем изменении цвета линии. Свойство списка Value содержит номер выбранной строки (строки списка нумеруются с единицы). Перейдите к подфункции pmColor_Callback и запрограммируйте обработку выбора пользователя. Используйте оператор switch для установки цвета линии в зависимости от номера выбранной строки списка, согласно листингу.

Листинг. Обработка выбора пользователя из раскрывающегося списка.

```
function varargout = pmColor_Callback (h, eventdata, handles, varargin)
```

```
% Определение номера выбранной строки
Num = get(h, ‘Value’)
switch Num
case 1
% выбрана первая строка, следует сделать линию синей
set (handles.line, ‘Color’, ‘b’);
case 2
% выбрана вторая строка, следует сделать линию красной
set (handles.line, ‘Color’, ‘r’);
case 3
% выбрана третья строка, следует сделать линию зеленой
set (handles.line, ‘Color’, ‘g’);
end
```

Запустите приложение, постройте график, нажав на кнопку Построить, и убедитесь в том, что раскрывающийся список позволяет изменять цвет линии графика функции. Несложно заметить, что интерфейс mygui имеет ряд недостатков.

Повторное построение графика не учитывает текущий выбор цвета в раскрывающемся списке.

Выбор цвета при отсутствии линии на графике приводит к ошибке (handles.line указывает на несуществующий объект).

Рядом со списком требуется разместить текст, поясняющий назначение списка.

Устраните первый недостаток, поместите в подфункции btnPlot_Callback обработки нажатия кнопки Построить блок switch для задания цвета построенной линии в зависимости от выбора опции раскрывающегося списка, согласно листингу.

Листинг. Учет выбора опции раскрывающегося списка при построении графика.

```
function varargout = btnPlot_Callback (h, eventdata, handles, varargin)
…
% Определение номера выбранной строки
case 1
set (handles.line, ‘Color’, ‘b’);
case 2
set (handles.line, ‘Color’, ‘r’);
case 3
set (handles.line, ‘Color’, ‘g’);
```

end

Изменение цвета линии при отсутствии графика лишено смысла, поэтому следует запретить доступ пользователя к раскрывающемуся списку и, напротив, разрешить при построении графика. В начале работы приложения список должен быть недоступен для пользователя. Установите в редакторе свойств для раскрывающегося списка Enable в off. Внесите необходимые изменения в подфункции btnPlot_Callback и btnClear_Callback, соответствующие нажатию на кнопки, используйте свойство списка Enable, согласно листингу.

Листинг. Разрешение и запрещение доступа к раскрывающемуся списку.

```
function varargout = btnPlot_Callback (h, eventdata, handles, varargin)
…
% Разрешение доступа к раскрывающемуся списку
set (handles.pmColor, ‘Enable’, ‘on’)
function varargout = btnClear_Callback (h, eventdata, handles, varargin)
…
% Запрещение доступа к раскрывающемуся списку
set (handles.pmColor, ‘Enable’, ‘off’)
```

Многие элементы интерфейса, в частности раскрывающиеся списки, следует сопровождать поясняющим текстом. Перейдите в режим редактирования и при помощи панели управления разместите текстовую область над списком. Установите в редакторе свойств string в значение цвет линии, HorisontalAlignment – в значение left для добавленного объекта, используйте кнопки в строках с названиями свойств. Теперь работающее приложение имеет более наглядный интерфейс.

Программирование событий обычных списков производится практически аналогично. Отличие состоит в том, что в обычных списках может быть выделено несколько элементов. Свойство Value содержит вектор номеров выбранных элементов. Разрешение выбора нескольких элементов определяется значениями свойств Max и Min. Если разность Max — Min больше единицы, то пользователь может выделить несколько строк.

Полосы скроллинга. Усовершенствуйте интерфейс приложения mygui, предоставив пользователю возможность устанавливать ширину линии при помощи полосы скроллинга. Добавьте полосу

скроллинга в окно приложения и задайте название scrWidth в свойстве Tag полосы. Снабдите полосу скроллинга текстовым пояснением "Толщина линии" так же, как и раскрывающийся список.

Заметим, что простой щелчок мышью в окне приложения при добавлении полосы скроллинга приводит к появлению вертикально расположенной полосы. Вертикальное или горизонтальное направление зависит от соотношения ширины и высоты полосы скроллинга. Измените размер для получения горизонтально расположенной полосы. Проще всего сразу нарисовать прямоугольник полосы скроллинга. удерживая нажатой левую кнопку мыши.

Теперь следует определить соответствие между положением бегунка полосы числовым значением свойства Value.

Проделайте следующие установки из редактора свойств. В Max занесите десять, а в Min — единицу. Свойства Max и Min полосы скроллинга отвечают за границы значений, записываемых в Value, при перемещении бегунка.

Определите начальное положение, записав в Value единицу. Нажмите кнопку в строке с названием свойства и в появившемся окне Value измените значение на единицу.

Обратитесь к свойству SliderStep. Его значением является вектор из двух компонентов, первый из которых определяет относительное изменение Value при нажатии на кнопки со стрелками полосы скроллинга, а второй — при перетаскивании бегунка мышью. Следует установить значение [0.1 0.2] свойства SliderStep для того, чтобы нажатие на кнопки полосы изменяло Value на десять процентов, а щелчок мыши справа или слева от бегунка – на двадцать. Раскройте строку SliderStep щелчком мыши по знаку плюс слева от названия свойства и в появившихся строках x и y введите 0.1 и 0.2.

Удалось запрограммировать событие Callback полосы скроллинга с именем scrwidth, которое состоит в задании ширины линии, равной округленному значению Value. Перейдите к подфункции scrWidth_Callback и добавьте в ней оператор установки ширины линии, согласно листингу.

Листинг. Обработка события Callback полосы скроллинга.

```
function varargout = scrWidth__Callback(h, eventdata, handles, varargin)
% Получеие ширины линии в зависимости от бегунка
% на полосе скроллинга width = get(h, 'Value');
% Установка толщины линии
```

```
set(handles.line, 'LineWidth', round(width))
```

Запустите mygui и убедитесь, что полоса скроллинга позволяет легко изменять толщину линии построенного графика. Устраните самостоятельно некоторые недочеты интерфейса. Полоса скроллинга должна быть недоступной после очистки осей кнопкой Очистить, построение графика при помощи кнопки Построить произведите с учетом установленной ширины линии. Данные недостатки исправляются внесением соответствующих изменений в обработку событий callback перечисленных кнопок так же, как и в предыдущих разделах.

Область ввода текста. Обычные текстовые области, использовавшиеся на протяжении предыдущих разделов, позволяют лишь вывести некоторый текст в окно приложения. Обмен текстовой информацией между пользователем и приложением осуществляется при помощи областей ввода текста. Предоставьте пользователю возможность размещать заголовок на графике. Текст заголовка пользователь вводит в соответствующей строке.

Добавьте в окно приложения область ввода текста, установите значение editTitle свойству Tag области ввода и снабдите ее пояснением в текстовой области, расположенной выше. В редакторе свойств удалите из String строку Edit Text, для чего нажмите кнопку в строке с названием свойства и сотрите текст в окне String.

Запрограммируйте событие callback области ввода, перейдите к подфункции editTitle_Callback и поместите операторы, которые считывают значения свойства string области ввода и наносят заголовок на график, согласно листингу.

Листинг. Обработка события Callback области ввода текста.

```
function varargout = editTitle_Callback(h, eventdata, handles, varargin)
txt = get(h, 'String') ;
title(txt)
```

Запустите приложение mygui и нанесите заголовок на график, набрав его в области ввода и нажав клавишу <Enter>. Дополните обработку события, соответствующего нажатию кнопки Очистить**,** очисткой заголовка при помощи команды title ('').

Описанный пример демонстрирует использование области ввода, состоящей из одной строки. Разность значений свойств Max и Min области ввода определяет, позволяет ли данная область ввод многострочного текста.

Если разность больше единицы, то заносимый пользователем текст записывается в массив ячеек текстовых строк, который хранится в свойстве String.

2. Задания и варианты лабораторных работ

Для адаптивной системы управления общего вида

$$\frac{dx(t)}{dt} = \Phi\big(x(t), x(t-h), u(t), \xi\big) + f_{\xi}(t), \;\; y(t) = L^{T}(\xi)x(t),$$

$$x(\theta) = \varphi(\theta), \;\; \theta \in [-h; 0],$$

$$u_k = u_k(y_k, y_{k-s}, \chi_k, r_k), \qquad (1)$$

$$y_k = y(t_k), \;\; y_{k-s} = y(t_{k-s}) = y(t_k - h),$$

$$u(t) = u_k \text{ при } t_k \le t < t_{k+1},$$

где $x(t) \in R^n$ – вектор состояния объекта управления; $y(t) \in R^l$ – вектор выхода; $u(t) \in R^m$ – вектор управляющих воздействий; $f(t) \in R^n$ – вектор возмущающих воздействий; χ_k – матрица настраиваемых коэффициентов регулятора; $r_k \in R^m$ – вектор задающих воздействий; $t_k = k\tau$ – дискретный аналог времени; $\tau = const > 0$ – шаг дискретизации; $k = 0, 1, 2, \ldots$ – номер шага; $\Phi\big(x(t), x(t-h), u(t), \xi\big)$ – отображение, удовлетворяющее условиям существования и единственности решения уравнений (1) при заданной начальной вектор-функции $\varphi(\theta) \in C_h$; C_h – пространство ограниченных непрерывных функций; $h = const \ge 0$ – запаздывание; $s = h/\tau$ – число тактов запаздывания; $L(\xi)$ – квазистационарная матрица выхода соответствующего размера; $\xi \in \Xi$ – набор всех неизвестных параметров; Ξ – известное множество возможных значений ξ, требуется:

1. Построить Simulink-модель (S-модель) гибридной системы адаптивного управления с помощью пакета MATLAB (согласно заданному варианту, используя структурную схему, математическое описание и исходные данные).

Обеспечить возможность наблюдения следующих функций времени S-модели: $x(t) \in R^n$; $y(t) \in R^l$; $e_k \in R^n$; $u_k \in R^m$; $f(t) \in R^n$; χ_k; $r_k \in R^m$. S-модель представить в виде файла с именем – N_НомерВарианта_НомерСхемы_1.mdl.

2. Сформировать блок S-функции из заданного состава элементов S-модели (см. соответствующую структурную схему) и повторно выполнить все задания п.1. S-модель совместно с S-функцией представить в файле – N_НомерВарианта_НомерСхемы_2.mdl.

3. Объединить S-модели, построенные в п.1 и 2, представив их в виде схемы рассогласования для выходов систем управления. S-модель представить в файле N_НомерВарианта_НомерСхемы_3.mdl.

4. Разработать графический интерфейс для адаптивный системы управления в виде S-модели (см. результаты п.2), используя возможности подсистемы Guide пакета MatLab.

5. Интерфейс должен позволять: просматривать результаты моделирования и справочную информацию; изменять время моделирования, параметры объекта управления и эталонной модели, значения коэффициентов контура адаптации, а также шаг дискретизации.

Лабораторная работа 1. ГСПАУ с ЯЭМ (n = 3, l = 3, m = 1).

$$\frac{dx(t)}{dt} = Ax(t) + bu(t) + f(t), \;\; y(t) = x(t), \tag{2}$$

$$\bar{x}_{k+1} = P_M \bar{x}_k + d_M r_k, \;\; \bar{y}_k = \bar{x}_k, \tag{3}$$

$$e_k = \bar{x}_k - x_k, \;\; z_k = g^T e_k \,\|\, e_k \,\|^q, \;\; x_k = x(t_k), \tag{4}$$

$$u_k = \chi_{1,k} r_k + \chi^T_{2,k} x_k, \tag{5}$$

$$\chi_{1,k} = \chi_{1,k-1} + h_1 z_k r_k, \;\; \chi_{2,k} = \chi_{2,k-1} + H_2 z_k x_k, \tag{6}$$

$$h_1 = const > 0, \; H_2 = diag\{h_{2i}\}, \;\; h_{2i} = const > 0, \;\; i = \overline{1,n}, \tag{7}$$

$$u(t) = u_k \text{ при } t_k \le t < t_{k+1}. \tag{8}$$

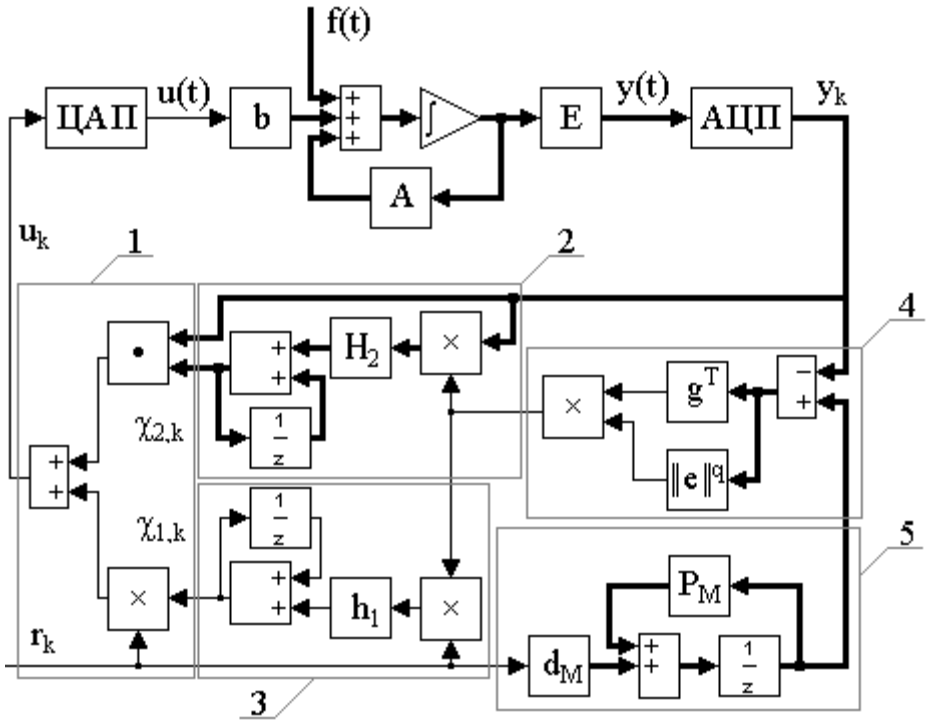

Рис. 2. Схема системы (2) – (8).

$$A=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & -23 & -9\end{pmatrix},\ b=\begin{pmatrix}0\\0\\1\end{pmatrix},\ \mathrm{f(t)}=\begin{pmatrix}0\\0\\ \exp(-t)\end{pmatrix},\ x(0)=\begin{pmatrix}0\\0\\0\end{pmatrix},\ g=\begin{pmatrix}8\\6\\1\end{pmatrix},$$

$$P_M=\begin{pmatrix}0.6301 & 0.2966 & 0.0344\\ -0.5157 & -0.1607 & -0.0129\\ 0.1928 & -0.2200 & -0.0450\end{pmatrix},\ d_M=\begin{pmatrix}0.0247\\0.0344\\-0.0129\end{pmatrix},\ \bar{x}_0=\begin{pmatrix}0\\0\\0\end{pmatrix}, \qquad (9)$$

$$r_k=0.8\sin(k\tau)-1.1,\ \tau=1,\ q=2,\ h_1=320,$$

$$H_2=diag\{120000\quad 50000\quad 200000\}.$$

Лабораторная работа 2. ГСПАУ с ЯЭМ (n = 3, l =2, m = 1).

$$\frac{dx(t)}{dt}=Ax(t)+bu(t)+f(t),\ \ y(t)=L^Tx(t), \qquad (10)$$

$$\bar{x}_{k+1}=P_M\bar{x}_k+d_Mr_k,\ \ \bar{y}_k=L^T\bar{x}_k, \qquad (11)$$

$$\nu_k=g^T(\bar{y}_k-y_k),\ \ y_k=y(t_k), \qquad (12)$$

$$u_k=\chi_{1,k}r_k+\chi_{2,k}^Ty_k, \qquad (13)$$

$$\chi_{1,k}=\chi_{1,k-1}+h_1\nu_k\|\nu_k\|^q\,r_k,\ \chi_{2,k}=\chi_{2,k-1}+H_2\nu_k\|\nu_k\|^q\,y_k, \qquad (14)$$

$$h_1=const>0,\qquad H_2=diag\{h_{2i}\},\ h_{2i}=const>0,\ i=\overline{1,l}, \qquad (15)$$

$$u(t)=u_k\ \text{при}\ t_k\le t<t_{k+1}. \qquad (16)$$

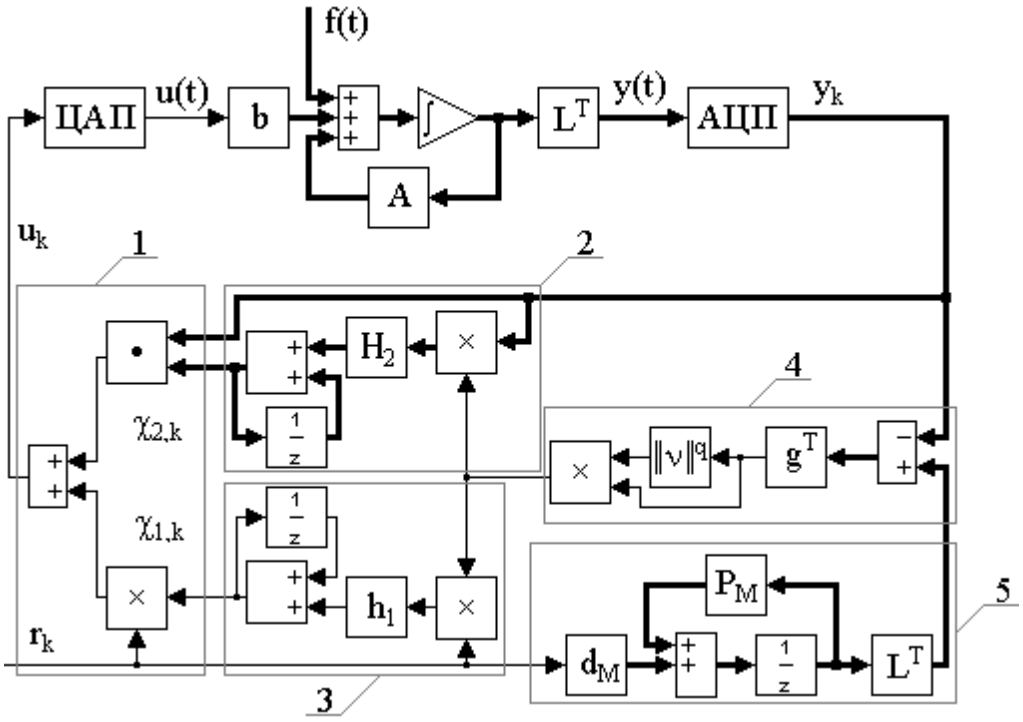

Рис. 3. Схема системы (10) – (16).

$$A=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & -23 & -9\end{pmatrix},\ b=\begin{pmatrix}0\\0\\1\end{pmatrix},\ L=\begin{pmatrix}4 & 0\\ 4 & -2\\ 0 & 1\end{pmatrix},\ f(t)=\begin{pmatrix}0\\0\\ \exp(-t)\end{pmatrix},\ x(0)=\begin{pmatrix}0\\0\\0\end{pmatrix},$$

$$P_M=\begin{pmatrix}0.6301 & 0.2966 & 0.0344\\ -0.5157 & -0.1607 & -0.0129\\ 0.1928 & -0.2200 & -0.0450\end{pmatrix},\ d_M=\begin{pmatrix}0.0247\\0.0344\\-0.0129\end{pmatrix},\ \bar{x}_0=\begin{pmatrix}0\\0\\0\end{pmatrix},\quad (17)$$

$$g=\begin{pmatrix}2\\1\end{pmatrix},\ r_k=0.1\sin(k\tau)+1,\ \tau=1,\ q=2,\ h_1=110,\ H_2=diag\{3450\quad 0.6\}.$$

Лабораторная работа 3. ГСПАУ с ЯЭМ ($n = 3$, $l = 3$, $m = 3$).

$$\frac{dx(t)}{dt}=Ax(t)+Bu(t)+f(t),\ \ y(t)=x(t), \quad (18)$$

$$\bar{x}_{k+1}=P_M\bar{x}_k+D_M r_k,\ \ \bar{y}_k=\bar{x}_k, \quad (19)$$

$$e_k=\bar{x}_k-x_k,\ \ x_k=x(t_k),\ \ z_k=G^T e_k\,\|\,e_k\,\|^q, \quad (20)$$

$$u_k=\chi_{1,k}r_k+\chi_{2,k}y_k,\ \ y_k=y(t_k), \quad (21)$$

$$\chi_{1ji,k}=\chi_{1ji,k-1}+h_{1ji}r_{i,k}z_{j,k},\ \ h_{1ji}=const>0,\ \ i,j=\overline{1,m}, \quad (22)$$

$$\chi_{2ji,k}=\chi_{2ji,k-1}+h_{2ji}y_{i,k}z_{j,k},\ \ h_{2ji}=const>0,\ \ i=\overline{1,l},\ \ j=\overline{1,m}, \quad (23)$$

$$u(t)=u_k\ \text{при}\ t_k\le t<t_{k+1} \quad (24)$$

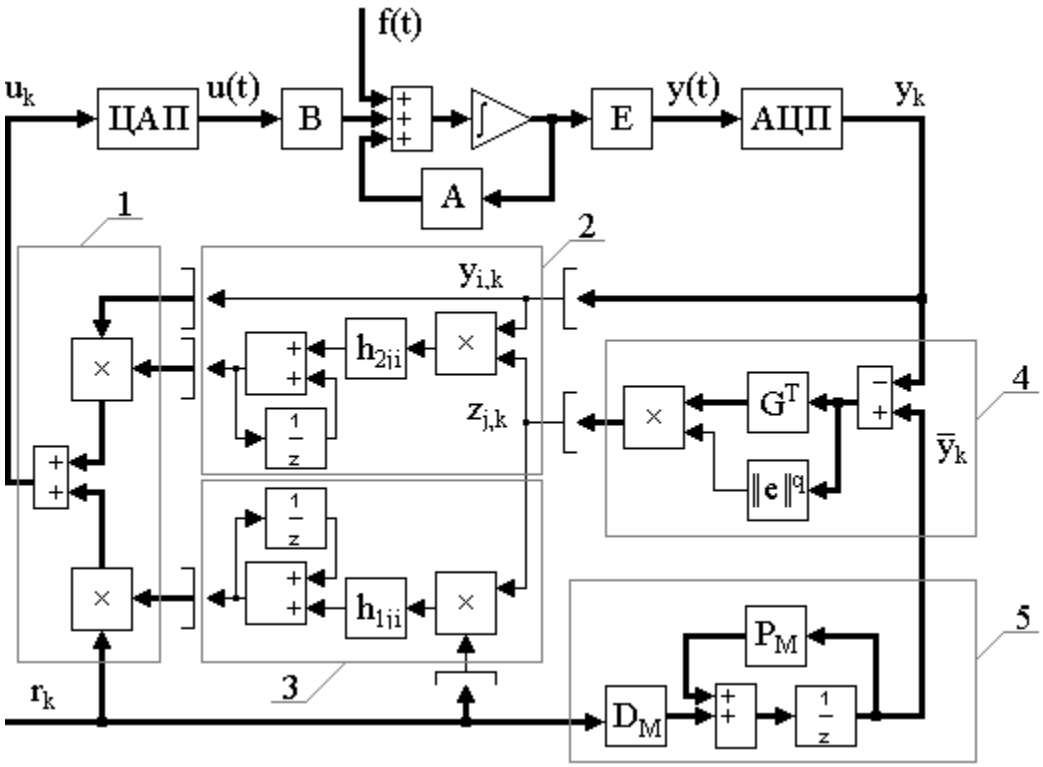

Рис. 4. Схема системы (18) – (24).

$$A=\begin{pmatrix}0&1&0\\0&0&1\\1&1&1\end{pmatrix},\ B=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},\ f(t)=\begin{pmatrix}\exp(-t)\\\exp(-t)\\\exp(-t)\end{pmatrix},\ x(0)=\begin{pmatrix}0\\0\\0\end{pmatrix},\ \bar{x}_0=\begin{pmatrix}0\\0\\0\end{pmatrix},$$

$$P_M=\begin{pmatrix}0.9940&0.1888&0.0135\\-0.0807&0.8461&0.1081\\-0.6483&-1.2693&0.1977\end{pmatrix},D_M=\begin{pmatrix}0.1078&0.0388&0.0020\\-0.0032&0.3775&0.0269\\-0.0436&-0.3078&0.2161\end{pmatrix},\quad(25)$$

$$G=\begin{pmatrix}1&0.5&0.5\\0.5&1&0.5\\0.5&0.5&1\end{pmatrix},\ r_k=\begin{pmatrix}0.3\sin(k\tau)\\0\\0.2\sin(k\tau)\end{pmatrix},\ \tau=0.2,\ q=4,\ h_{1ji}=1,\ h_{2ji}=1.$$

Лабораторная работа 4. ГСПАУ с ЯЭМ ($n = 3$, $l = 3$, $m = 3$).

$$\frac{dx(t)}{dt}=Ax(t)+Bu(t)+f(t),\ \ y(t)=L^Tx(t),\qquad(26)$$

$$\bar{x}_{k+1}=P_M\bar{x}_k+D_Mr_k,\ \ \bar{y}_k=L^T\bar{x}_k,\qquad(27)$$

$$v_k=G^T(\bar{y}_k-y_k),\ \ y_k=y(t_k),\qquad(28)$$

$$u_k=\chi_{1,k}r_k+\chi_{2,k}y_k,\qquad(29)$$

$$\chi_{1ji,k}=\chi_{1ji,k-1}+h_{1ji}r_{i,k}v_{j,k}\,\|v_k\|^q,\quad h_{1ji}=const>0,\ \ i,j=\overline{1,m},\qquad(30)$$

$$\chi_{2ji,k}=\chi_{2ji,k-1}+h_{2ji}y_{i,k}v_{j,k}\,\|v_k\|^q,\ h_{2ji}>0,\ \ i=\overline{1,l},\ \ j=\overline{1,m},\qquad(31)$$

$$u(t)=u_k\ \text{при}\ t_k\le t<t_{k+1}.\qquad(32)$$

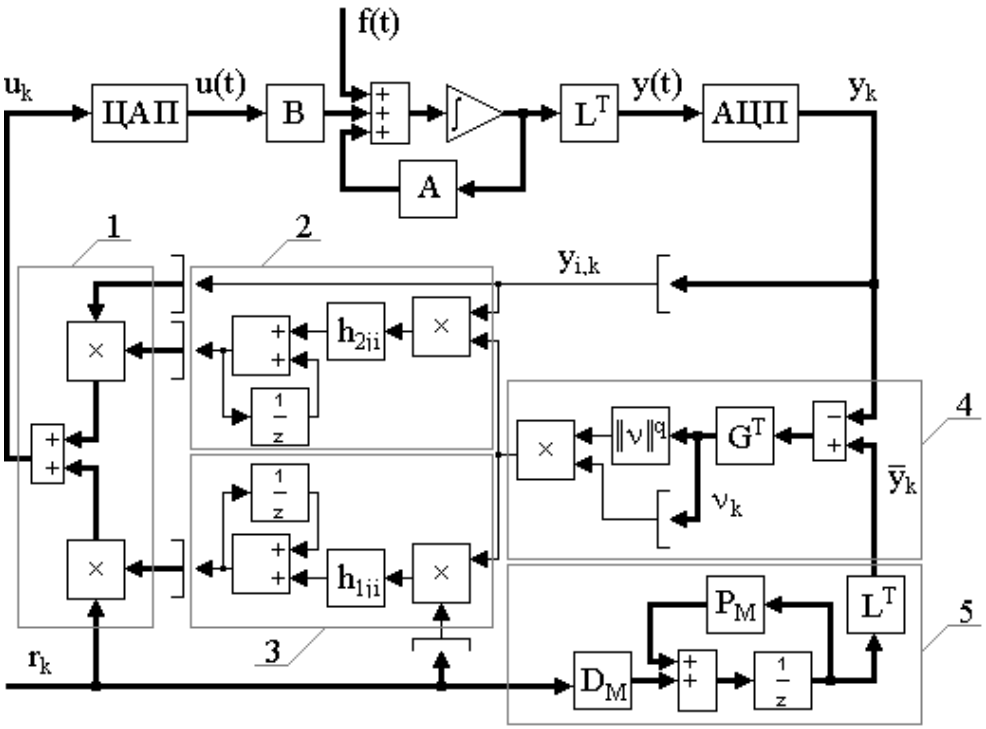

Рис. 5. Схема системы (26) – (32).

$$A=\begin{pmatrix}0&1&0\\0&0&1\\1&1&1\end{pmatrix},\ B=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},\ L=\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix},\ f(t)=\begin{pmatrix}\exp(-t)\\\exp(-t)\\\exp(-t)\end{pmatrix},\ x(0)=\begin{pmatrix}0\\0\\0\end{pmatrix},$$

$$P_M=\begin{pmatrix}0.9642&0.3303&0.0364\\-0.2186&0.5635&0.1117\\-0.6702&-1.4473&-0.1068\end{pmatrix},\ D_M=\begin{pmatrix}0.2138&0.1445&0.0119\\-0.0193&0.6606&0.0729\\-0.1180&-0.8731&0.2234\end{pmatrix},\quad (33)$$

$$\bar{x}_0=\begin{pmatrix}0\\0\\0\end{pmatrix},\ G=\begin{pmatrix}1&0.5&0.5\\0.5&1&0.5\\0.5&0.5&1\end{pmatrix},\ r_k=\begin{pmatrix}0.1\\0\\0.1\end{pmatrix},\ \tau=0.4,\ q=4,\ h_{1ji}=1,\ h_{2ji}=1.$$

Лабораторная работа 5. ГСПАУ с НЭМ ($n = 3$, $l = 2$, $m = 2$).

$$\frac{dx(t)}{dt}=Ax(t)+Bu(t)+f(t),\ \ y(t)=L^Tx(t), \qquad (34)$$

$$\varepsilon_k=r_*-\nu_k,\ \nu_k=G^Ty_k,\ y_k=y(t_k), \qquad (35)$$

$$u_k=\chi_{1,k}r_*+\chi_{2,k}\nu_k,\ u(t)=u_k\ \text{при}\ t_k\le t<t_{k+1}, \qquad (36)$$

$$\chi_{1i,k}=\chi_{1i,k-1}+h_{1i}\varepsilon_{i,k}\,\|\,\varepsilon_k\,\|^q\ r_{i,*},\ h_{1i}=const>0, \qquad (37)$$

$$\chi_{2i,k}=\chi_{2i,k-1}+h_{2i}\varepsilon_{i,k}\,\|\,\varepsilon_k\,\|^q\ \nu_{i,k},\ h_{2i}=const>0,\ i=\overline{1,m}. \qquad (38)$$

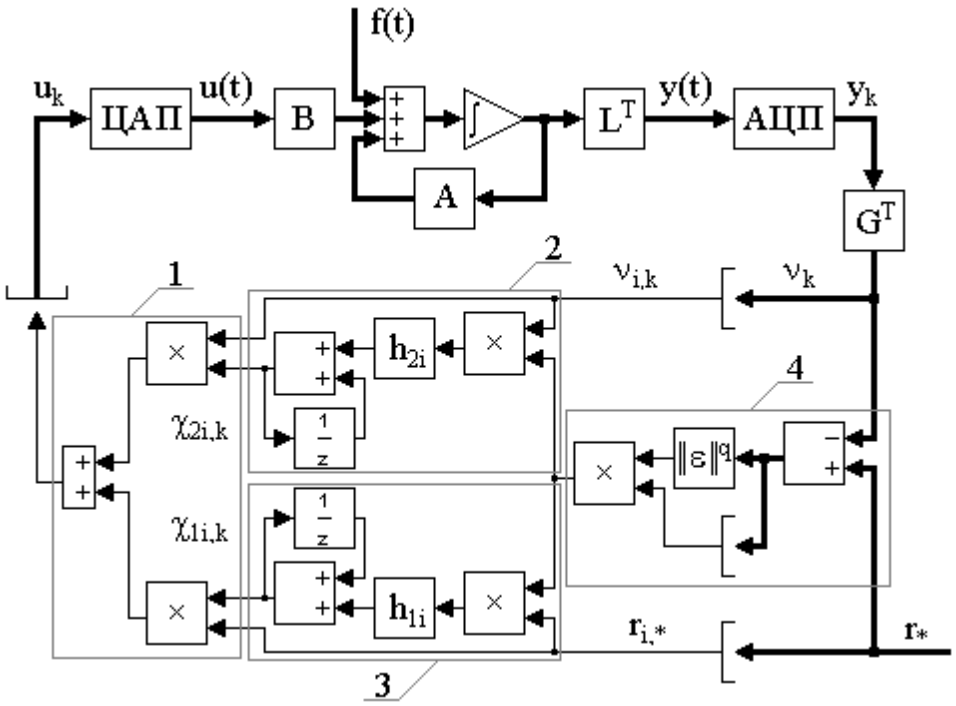

Рис. 6. Схема системы (34) – (38).

$$A = \begin{pmatrix} 4.8 & 0.28 & 2.82 \\ 2.9 & -1.33 & 1.82 \\ 0.04 & 4.77 & -2.22 \end{pmatrix}, \quad B = \begin{pmatrix} 2.8 & 0.99 \\ 0.125 & 1.28 \\ 0.63 & 2.17 \end{pmatrix},$$

$$f(t) = \begin{pmatrix} 0 \\ 0 \\ -0.05\exp(-t) \end{pmatrix}, \quad x(0) = \begin{pmatrix} 0 \\ 0 \\ 0.01 \end{pmatrix},$$

$$G = \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix}, \quad L^T = \begin{pmatrix} 1.56 & 0.98 & 0.55 \\ 0.79 & 3.42 & 0.95 \end{pmatrix}, \quad r_* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{39}$$

$$h_{21} = 3000, \quad h_{22} = 160,$$

$$\tau = 0.2, \quad q = 4.$$

Лабораторная работа 6. ГСПАУ с НЭМ ($n = 3$, $l = 2$, $m = 1$).

$$\frac{dx(t)}{dt} = Ax(t) + bu(t) + f(t), \quad y(t) = L^T x(t), \tag{40}$$

$$\varepsilon_k = r_* - \nu_k, \quad \nu_k = g^T y_k, \quad y_k = y(t_k), \tag{41}$$

$$u_k = \chi_{1,k} r_* + \chi_{2,k} \nu_k, \quad u(t) = u_k \text{ при } t_k \le t < t_{k+1}, \tag{42}$$

$$\chi_{1,k} = \chi_{1,k-1} + h_1 \varepsilon_k \,\|\, \varepsilon_k \,\|^q \, r_*, \quad h_1 = const > 0, \tag{43}$$

$$\chi_{2,k} = \chi_{2,k-1} + h_2 \varepsilon_k \,\|\, \varepsilon_k \,\|^q \, \nu_k, \quad h_2 = const > 0. \tag{44}$$

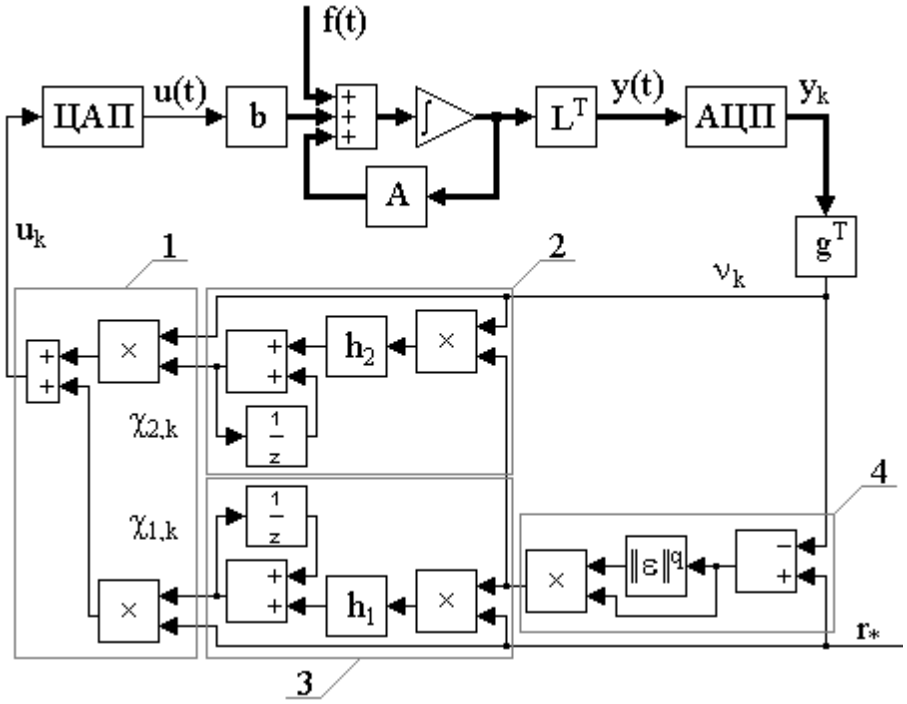

Рис. 7. Схема системы (40) – (44).

$$A=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & -26 & -9\end{pmatrix},\ b=\begin{pmatrix}0\\0\\1\end{pmatrix},\ L=\begin{pmatrix}4 & 0\\ 4 & -2\\ 0 & 1\end{pmatrix},$$

$$f(t)=\begin{pmatrix}0\\0\\0.1\sin(t)\end{pmatrix},\ x(0)=\begin{pmatrix}0\\0\\0\end{pmatrix}, \tag{45}$$

$$g^T=(2\quad 1),\ r_*=1,\ h_1=1,\ h_2=0.9,$$

$$\tau=5,\ q=3.$$

Лабораторная работа 7. ГСПАУ с запаздыванием ($n = 3$, $l = 2$, $m = 2$).

$$\frac{dx(t)}{dt}=Ax(t)+\Gamma x(t-h)+Bu(t)+f(t),\ y(t)=L^Tx(t), \tag{46}$$

$$x(\theta)=\varphi(\theta),\ \theta\in[-h;0], \tag{47}$$

$$u_k=\chi_{1,k}r_*+\chi_{2,k}v_k+\chi_{3,k}v_{k-s},\ u(t)=u_k \text{ при } t_k\le t<t_{k+1}, \tag{48}$$

$$v_k=v(t_k),\ v_{k-s}=v(t_{k-s})=v(t_k-h). \tag{49}$$

$$\chi_{1i,k}=\chi_{1i,k-1}+h_{1i}\varepsilon_{i,k}\,\|\,\varepsilon_k\,\|^q\ r_{i,*},\ h_{1i}=const>0,\ i=\overline{1,m}, \tag{50}$$

$$\chi_{2i,k}=\chi_{2i,k-1}+h_{2i}\varepsilon_{i,k}\,\|\,\varepsilon_k\,\|^q\ v_{i,k},\ h_{2i}=const>0,\ i=\overline{1,m}, \tag{51}$$

$$\chi_{3i,k} = \chi_{3i,k-1} + h_{3i}\varepsilon_{i,k} \parallel \varepsilon_k \parallel^q \nu_{i,k-s},\ h_{3i} = const > 0,\ i = \overline{1,m}, \qquad (52)$$

$$\varepsilon_k = r_* - \nu_k,\ \nu_k = G^T y_k,\ y_k = y(t_k). \qquad (53)$$

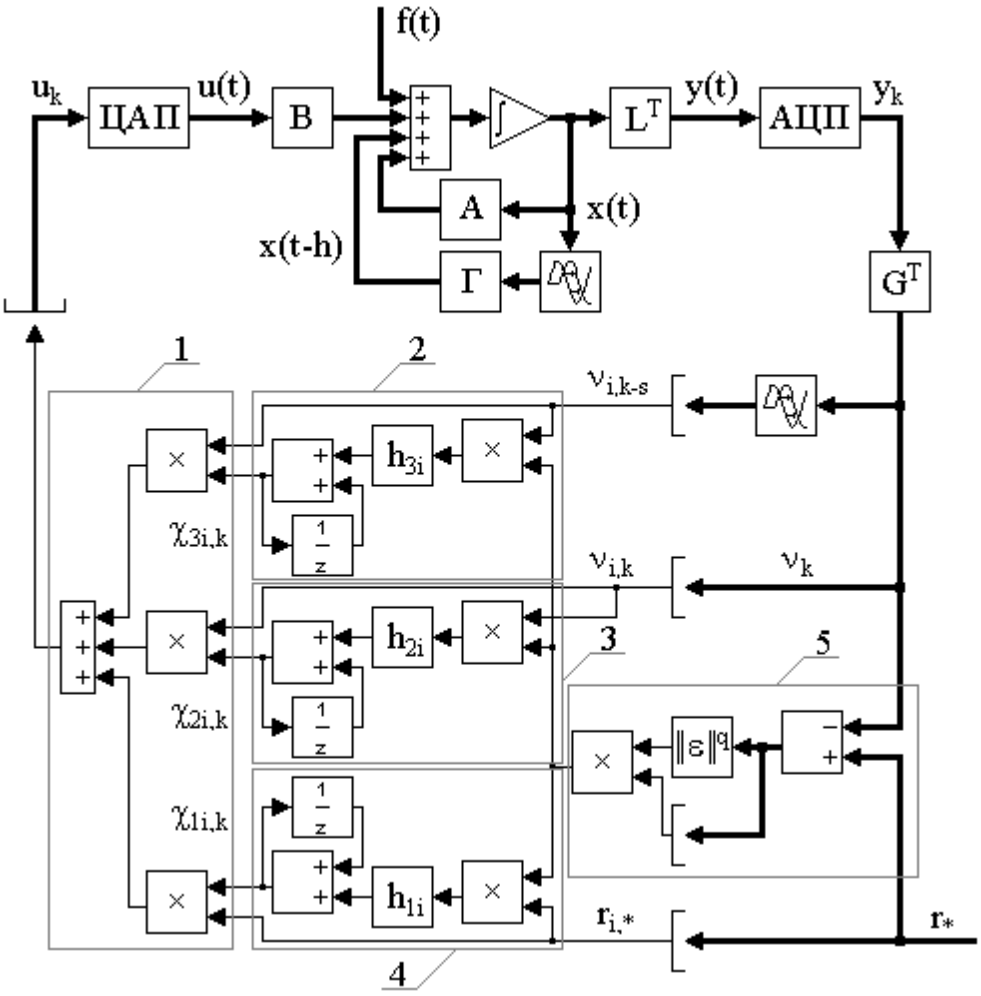

Рис. 8. Схема системы (46) – (53).

$$A = \begin{pmatrix} 4.8 & 0.28 & 2.82 \\ 2.9 & -1.33 & 1.82 \\ 0.04 & 4.77 & -2.22 \end{pmatrix},\ B = \begin{pmatrix} 2.8 & 0.99 \\ 0.125 & 1.28 \\ 0.63 & 2.17 \end{pmatrix},$$

$$\tilde{A} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1.3 & -0.3 & 1.4 \end{pmatrix},\ f(t) = \begin{pmatrix} 0 \\ 0 \\ -0.05\exp(-t) \end{pmatrix},\ x(0) = \begin{pmatrix} 0 \\ 0 \\ 0.01 \end{pmatrix}, \qquad (54)$$

$$G = \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix},\ L^T = \begin{pmatrix} 1.56 & 0.98 & 0.55 \\ 0.79 & 3.42 & 0.95 \end{pmatrix},\ r_* = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

$h = 2,\ q = 4,$

$h_{21} = 1750,\ h_{22} = 150,\ h_{31} = 125000,\ h_{32} = 35000,$

$\tau = 0.2.$

Лабораторная работа 8. ГСПАУ с запаздыванием ($n = 3$, $l = 2$, $m = 1$).

$$\frac{dx(t)}{dt} = Ax(t) + \Gamma x(t-h) + bu(t) + f(t),\ y(t) = L^T x(t), \tag{55}$$

$$x(\theta) = \varphi(\theta),\ \theta \in [-h; 0], \tag{56}$$

$$u_k = \chi_{1,k} r_* + \chi_{2,k} \nu_k + \chi_{3,k} \nu_{k-s},\ u(t) = u_k \text{ при } t_k \le t < t_{k+1}, \tag{57}$$

$$\nu_k = \nu(t_k),\ \nu_{k-s} = \nu(t_{k-s}) = \nu(t_k - h), \tag{58}$$

$$\chi_{1,k} = \chi_{1,k-1} + h_1 \varepsilon_k \,\|\, \varepsilon_k \,\|^q\, r_*,\ h_1 = const > 0, \tag{59}$$

$$\chi_{2,k} = \chi_{2,k-1} + h_2 \varepsilon_k \,\|\, \varepsilon_k \,\|^q\, \nu_k,\ h_2 = const > 0, \tag{60}$$

$$\chi_{3,k} = \chi_{3,k-1} + h_3 \varepsilon_k \,\|\, \varepsilon_k \,\|^q\, \nu_{k-s},\ h_3 = const > 0, \tag{61}$$

$$\varepsilon_k = r_* - \nu_k,\ \nu_k = g^T y_k,\ y_k = y(t_k) \tag{62}$$

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -23 & -9 \end{pmatrix},\ b = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},\ \tilde{A} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.1 & 0.3 & 0.3 \end{pmatrix},$$

$$f(t) = \begin{pmatrix} 0 \\ 0 \\ \exp(-t) \end{pmatrix},\ L = \begin{pmatrix} 4 & 0 \\ 4 & -2 \\ 0 & 1 \end{pmatrix},\ x(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},\ g = \begin{pmatrix} 2 \\ 0.9 \end{pmatrix},$$

$$r_* = 1,\ h = 1,\ q = 4, \tag{63}$$

$$h_1 = 0.5,\ h_2 = 1,\ h_3 = 0.03,$$

$$\tau = 6.$$

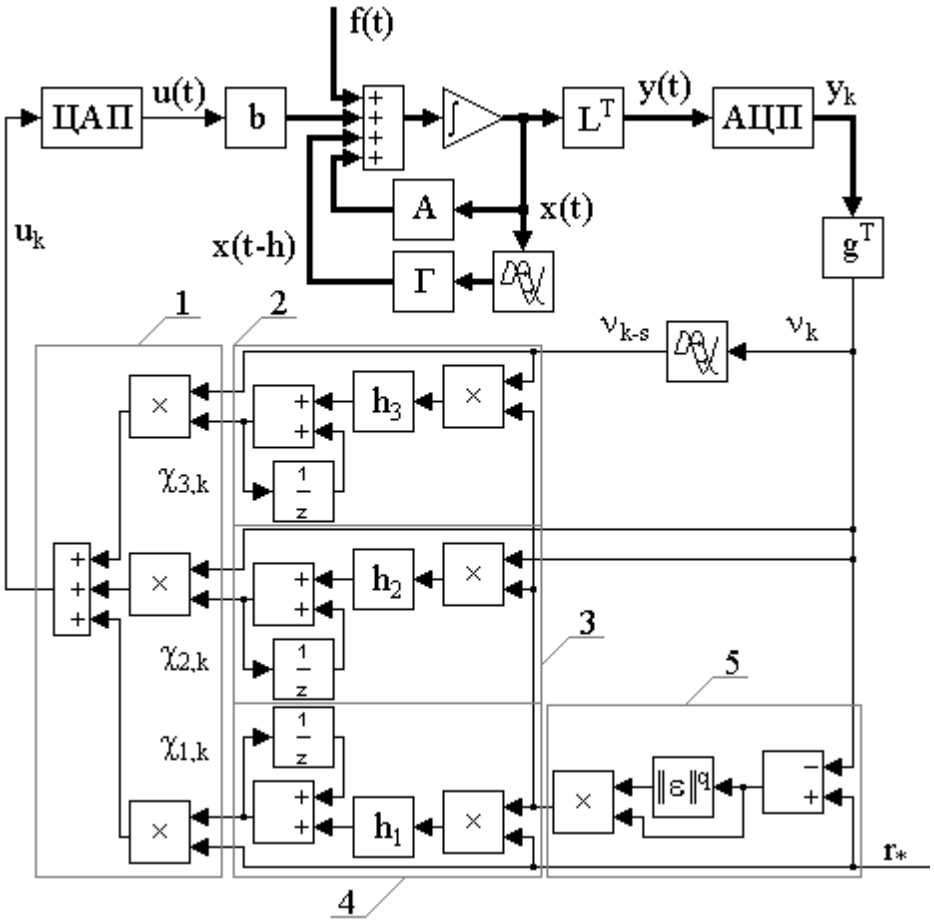

Рис. 9. Схема системы (55) – (62).

3. Пример выполнения лабораторных работ

Рассматривается динамическая система, описываемая уравнениями

$$\frac{dx(t)}{dt} = Ax(t) + bu(t),\ \ y(t) = x(t),\ \ \nu(t) = g^T y(t), \tag{64}$$

$$\frac{d\bar{x}(t)}{dt} = A_M \bar{x}(t) + b_M r_*,\ \ y_M(t) = x_M(t),\ \ \nu_M(t) = g^T y_M(t), \tag{65}$$

$$u(t) = \chi_1(t) r_* + \chi_2(t)\nu(t),\ \ e(t) = \nu_M(t) - \nu(t), \tag{66}$$

$$\frac{d\chi_1(t)}{dt} = h_1 r_* e(t),\ \ h_1 = const > 0, \tag{67}$$

$$\frac{d\chi_2(t)}{dt} = h_2 \nu(t) e(t), \tag{68}$$

$$h_2 = const > 0,$$

$$A=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 1 & -23 & -9\end{pmatrix},\ b=\begin{pmatrix}0\\0\\1\end{pmatrix},\ x(0)=\begin{pmatrix}0\\0\\0\end{pmatrix},$$

$$A_M=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -15 & -23 & -9\end{pmatrix},\ b_M=\begin{pmatrix}0\\0\\1\end{pmatrix},\ x_M(0)=\begin{pmatrix}0\\0\\0\end{pmatrix}, \qquad (69)$$

$$g^T=(8\quad 6\quad 1),\ r_*=1,\ h_1=1,\ h_2=1.$$

S-модель адаптивной системы управления (64) – (69) представлена на рис. 10.

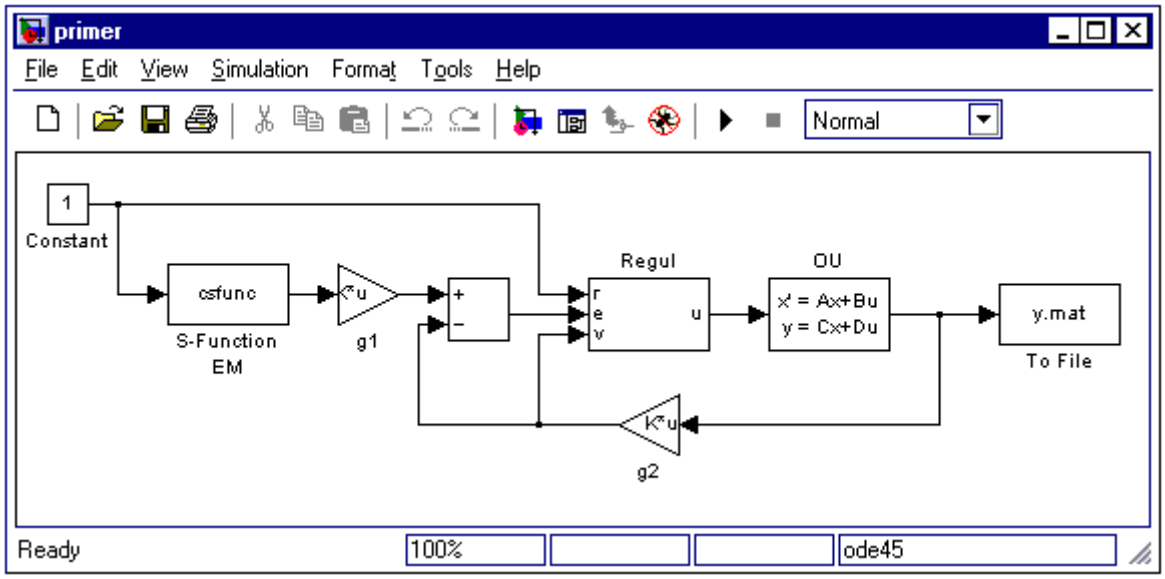

Рис. 10. S-модель системы (64) – (69).

Далее приведены два листинга программ, составляющих основу примера выполнения курсовой работы.

Листинг реализации блока эталонной модели в виде S-функции:

```
function [sys,x0,str,ts] = csfunc(t,x,u,flag)
% Задание матриц:
A=[ 0 1 0 ; 0 0 1 ; -15 -23 -9 ]; % Матрица состояния.
B=[ 0 ; 0 ; 1 ];                  % Матрица входа.
C=[ 1 0 0 ; 0 1 0 ; 0 0 1 ];      % Матрица выхода.
D=[ 0 ; 0 ; 0 ];                  % Матрица обхода.
switch flag,    % В зависимости от значения переменной flag происходит
            % вызов того или иного метода:
%===============%
% Инициализация %
%===============%
```

```
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);
%====================%
% Расчет производных %
%====================%
  case 1,
    sys=mdlDerivatives(t,x,u,A,B,C,D);
%============================================%
% Расчет значений вектора выходных сигналов %
%============================================%
  case 3,
    sys=mdlOutputs(t,x,u,A,B,C,D);
%=========================================%
% Неиспользуемые значения переменной flag %
%=========================================%
  % В примере не используются методы для завершения работы S-функции,
  % нет дискретных переменных состояния,
  % поэтому значения переменной flag = 2, 4, 9 не используются.
  % Результатом S-функции в этом случае является пустая матрица.
  case { 2, 4, 9 }
    sys=[];
%====================================%
% Неизвестное значение переменной flag %
%====================================%
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
% Окончание csfunc
%
%=============================================================
=%
% mdlInitializeSizes                                        %
% Функция инициализации                                     %
% Расчет начальных условий, значений вектора шагов модельного   %
% времени, размерности матриц                               %
%=============================================================
=%
%
function [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D)
sizes = simsizes;
sizes.NumContStates  = 3; % Число непрерывных переменных состояния.
sizes.NumDiscStates  = 0; % Число дискретных  переменных состояния.
sizes.NumOutputs     = 3; % Число выходных переменных.
sizes.NumInputs      = 1; % Число входных переменных.
sizes.DirFeedthrough = 0; % Прямой проход (матрица D пустая).
sizes.NumSampleTimes = 1; % Размерность вектора шагов модельного времени.
sys = simsizes(sizes);
x0 = zeros(3,1); % Задание вектора начальных значений переменных состояния.
```

```
            % Начальные условия нулевые
str = [];       % Параметр зарезервирован для будущего использования.
ts = [0 0];     % Матрица из двух колонок, задающая шаг модельного
            % времени и смещение.
% Окончание mdlInitializeSizes
%
%==========================================================================%
% mdlDerivatives                                                           %
% Функция для расчета значений производных вектора состояния непрерывной
% части системы
%==========================================================================%
%
function sys=mdlDerivatives(t,x,u,A,B,C,D)
sys = A*x + B*u;
% Окончание mdlDerivatives
%
%==========================================================================%
% mdlOutputs                                                               %
% Функция для расчета значений вектора выходных сигналов                   %
%==========================================================================%
%
function sys=mdlOutputs(t,x,u,A,B,C,D)
sys = C*x + D*u;
% Окончание mdlOutputs
```

Листинг графического интерфейса адаптивной управления с явной эталонной моделью вида (64) – (69):

```
function varargout = mygui(varargin)
if nargin == 0
        fig = openfig(mfilename,'reuse');
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1})
        try
                [varargout{1:nargout}] = feval(varargin{:});
        catch
                disp(lasterr);
        end
end

% Обработка события Callback кнопки с именем btnPlot
% --------------------------------------------------------------------
```

```
function varargout = btnPlot_Callback(h, eventdata, handles, varargin)
sim('primer');
s=load('y.mat');
subplot(222)
plot(s.ans(1,:),s.ans(2,:),s.ans(1,:),s.ans(3,:),s.ans(1,:),s.ans(4,:))
title('Выход объекта')
if get(handles.chbxGridX,'Value') set(gca,'XGrid','on')
else set(gca,'XGrid','off')
set(h,'Enable','off')
end
if get(handles.chbxGridY,'Value') set(gca,'YGrid','on')
else set(gca,'YGrid','off')
set(handles.btnClear,'Enable','on')
end
s=load('ka.mat');
subplot(224)
plot(s.ans(1,:),s.ans(2,:),s.ans(1,:),s.ans(3,:))
title('Настройки КА')
if get(handles.chbxGridX,'Value') set(gca,'XGrid','on')
else set(gca,'XGrid','off')
set(h,'Enable','off')
end
if get(handles.chbxGridY,'Value') set(gca,'YGrid','on')
else set(gca,'YGrid','off')
set(handles.btnClear,'Enable','on')
end
set(h,'Enable','off')
set(handles.btnClear,'Enable','on')

% Обработка события Callback кнопки с именем btnClear
% --------------------------------------------------------------------
function varargout = btnClear_Callback(h, eventdata, handles, varargin)
subplot(222), cla
subplot(224), cla
set(h,'Enable','off')
set(handles.btnPlot,'Enable','on')

% Обработка событий Callback флагов сетки
% --------------------------------------------------------------------
function varargout = chbxGridX_Callback(h, eventdata, handles, varargin)
subplot(222)
if get(h,'Value') set(gca,'XGrid','on')
else set(gca,'XGrid','off')
end
subplot(224)
if get(h,'Value') set(gca,'XGrid','on')
else set(gca,'XGrid','off')
end
```

```
% ---------------------------------------------------------------------
function varargout = chbxGridY_Callback(h, eventdata, handles, varargin)
subplot(222)
if get(h,'Value') set(gca,'YGrid','on')
else set(gca,'YGrid','off')
end
subplot(224)
if get(h,'Value') set(gca,'YGrid','on')
else set(gca,'YGrid','off')
end

% Ввод матрицы состояния A
% ---------------------------------------------------------------------
function varargout = editA_Callback(h, eventdata, handles, varargin)
A=get(h,'String');
set_param('primer/OU','A',A);

% Ввод вектора управления b
% ---------------------------------------------------------------------
function varargout = editB_Callback(h, eventdata, handles, varargin)
B=get(h,'String');
set_param('primer/OU','B',B);

% Ввод матрицы выхода L
% ---------------------------------------------------------------------
function varargout = editL_Callback(h, eventdata, handles, varargin)
L=get(h,'String');
set_param('primer/OU','C',L);

% Ввод матрицы обхода D
% ---------------------------------------------------------------------
function varargout = editD_Callback(h, eventdata, handles, varargin)
D=get(h,'String');
set_param('primer/OU','D',D);

% Ввод значений линейного компенсатора g
% ---------------------------------------------------------------------
function varargout = editG_Callback(h, eventdata, handles, varargin)
G=get(h,'String');
set_param('primer/g1','Gain',G);
set_param('primer/g2','Gain',G);

% Ввод коэффициентов алгоритмов контура адаптации
% ---------------------------------------------------------------------
function varargout = editH1_Callback(h, eventdata, handles, varargin)
h1=get(h,'String');
set_param('primer/Regul/h1','Gain',h1);
% ---------------------------------------------------------------------
```

```
function varargout = editH2_Callback(h, eventdata, handles, varargin)
h2=get(h,'String');
set_param('primer/Regul/h2','Gain',h2);
```

На рис. 11 показано MATLAB-приложение для имитационного моделирования адаптивной системы управления (64) – (69).

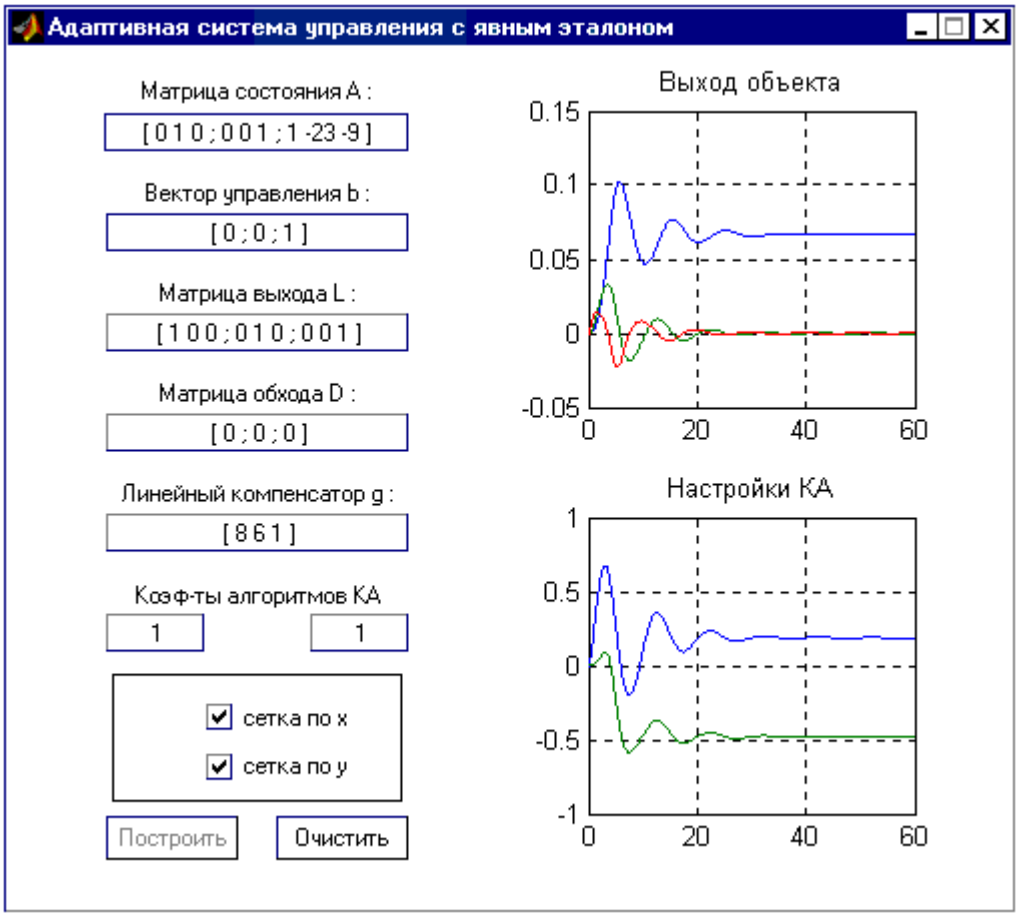

Рис. 11. Диалоговое окно для исследования системы (64) – (69).

КАРТА ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА

Виды учебных занятий	ФИО преподавателя, ученая степень, должность
Лекции	Шевко Денис Геннадьевич, к.т.н., доцент кафедры ИиУС
Лабораторные работы	Шевко Денис Геннадьевич, к.т.н., доцент кафедры ИиУС

СОДЕРЖАНИЕ

Денис Геннадьевич ШЕВКО

доцент кафедры Информационных и управляющих систем АмГУ, кандидат технических наук

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ

«МЕТОДЫ АНАЛИЗА ДИНАМИЧЕСКИХ СИСТЕМ»

для направления подготовки 230100.68

«Информатика и вычислительная техника»

Издательство АмГУ