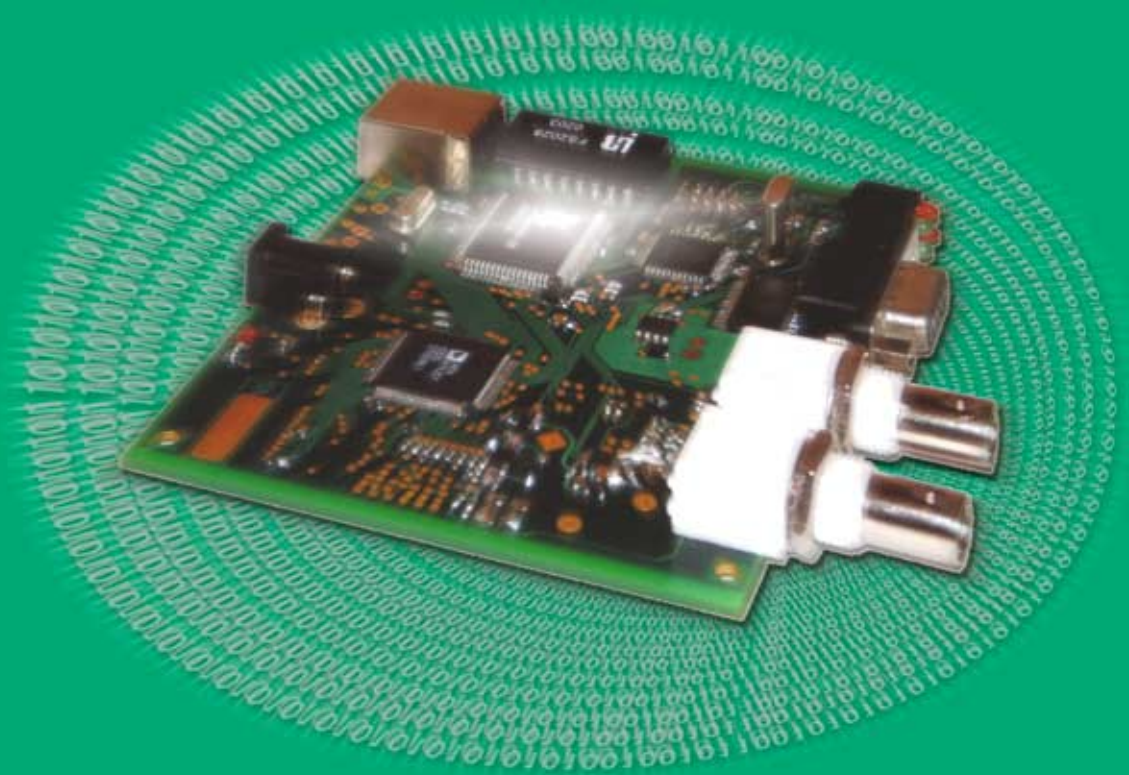
ISSN 1684–8853

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ



6/2003

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

6/2003

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

Учредитель и издатель:

Издательство «Политехника»

Адрес редакции: 191023, Санкт-Петербург, Инженерная ул., д. 6
Тел./факс: (812) 312-53-90
E-mail: asklab@aanet.ru

Журнал зарегистрирован в Министерстве РФ по делам печати, телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-12412 от 19 апреля 2002 г.

Журнал распространяется по подписке. Подписку можно оформить в любом отделении связи по каталогу агентства «Роспечать». Индекс 15385.

ЛР № 010292 от 18.08.98.
Сдано в набор 28.11.2003. Подписано в печать 30.12.2003. Формат 60×90 1/8.
Бумага офсетная. Гарнитура Pragmatica. Печать офсетная.
Усл. печ. л. 12,0. Уч.-изд. л. 13,3. Тираж 1000 экз. Заказ 835.

Оригинал-макет изготовлен
в отделе электронных публикаций и библиографии ГУАП.
190000, Санкт-Петербург, Б. Морская ул., 67.

Отпечатано с готовых диапозитивов в ООО «Политехника-сервис».
191023, Санкт-Петербург, Инженерная ул., д. 6.

УДК 681.1

МОДЕЛЬ УПРАВЛЕНИЯ ДЕЙСТВИЯМИ НАБЛЮДАТЕЛЯ ПРИ ВТОРИЧНОМ ПОИСКЕ

А. Н. Прокаев,
адъюнкт
Военно-Морская академия им. Н. Г. Кузнецова

Рассмотрено решение задачи нахождения оптимального алгоритма вторичного поиска (поиска подвижного объекта после потери контакта с ним) на основе теоретико-игрового подхода при различном характере распределения положения объекта в области неопределенности.

This article is devoted to solution secondary search (the mobile object search after losing) optimal algorithm problem on the search games basis with the different law of object position distribution in the uncertainty area.

Введение

Актуальность развития методов и моделей поиска подвижных объектов вытекает из наличия и объективного роста ситуаций, определяемых необходимостью обнаружения объектов, характеризующихся неопределенностью текущих координат в пространстве (аварийный самолет в спасательной операции, косяк рыбы при координируемом лове рыбы, подводная лодка противника в территориальных водах и пр.). При этом координация и распределение усилий сил поиска есть целенаправленный процесс, требующий эффективного управления. В современных условиях управление поиском осуществляется на базе широкого класса информационно-управляющих систем, позволяющих в реальном масштабе времени получать и обосновывать решения руководства поисковыми действиями на применение сил поиска. Предлагаемая модель управления действиями наблюдателя (поисковой системой) является дальнейшим развитием соответствующего класса задач теории поиска, используемых при разработке математического и программного обеспечения указанных информационно- управляющих систем.

Влияние основных ограничений теории поиска на эффективность поисковых действий в современных условиях

Как известно, основные положения теории поиска подвижных объектов были разработаны в период Второй мировой войны учеными группы оценки операций под руководством Б. О. Купмана [6–8] в 1956–1957 гг. Идеи, сформулированные в отчетах Б. О. Купмана, получили дальнейшее развитие как за рубежом, так и в трудах отечественных ученых. Однако развитие технических средств поиска и их носителей привело к появлению ограничений, затрудняющих или вовсе исключающих использование теории поиска подвижных объектов (ТППО) в ее классической редакции для решения ряда практических задач поиска. Приведем здесь ряд наиболее очевидных ограничений:

1) объект поиска оказывает активное противодействие наблюдателю;

2) зона обнаружения наблюдателя и (или) объекта поиска соизмерима с размерами района поиска;

3) форма и (или) размеры района поиска в процессе поиска изменяются;

4) объект поиска имеет возможность выйти из района поиска в течение времени поиска.

Традиционный подход исследования операций предполагает учет указанных ограничений путем наращивания математических моделей поиска неподвижного объекта элементами, играющими роль факторов, ограничивающих поисковые усилия [2]. В конечном счете это приводит к тому, что *тактика поиска цели, активно уклоняющейся от обнаружения, не имеет принципиальных отличий от тактики поиска неподвижной цели*, что непосредственно следует из анализа используемых показателей эффективности.

В качестве показателя эффективности поиска в большинстве случаев принято использовать вероятность обнаружения цели. Вероятность обнаружения одиночной цели при самостоятельном поиске в районе группой наблюдателей определяется выражением

$$P(t) = 1 - \exp(-\gamma_{\Sigma} t), \qquad (1)$$

где t – время поиска;

γ_{Σ} – суммарная интенсивность поиска группы из n наблюдателей.

Модель равновероятного распределения координат цели предполагает равномерное распределение поисковых усилий по району поиска, что достигается выделением каждому наблюдателю участка поиска в пределах района, площадь которого пропорциональна поисковой производительности данного наблюдателя. В результате этого интенсивность поиска цели на всех участках поиска становится одинаковой.

Для определения вероятности обнаружения цели, уклоняющейся от обнаружения, в качестве интенсивности поиска принимается величина

$$\gamma_{\text{н}} = \frac{2 d_{\text{ц}i} V_{\text{н}} \sin q_i}{S_i}, \qquad (2)$$

где $d_{\text{ц}i}$ – эффективная дальность обнаружения целью i-го наблюдателя;

$$q_i = \arcsin\frac{d_{нi}}{d_{цi}} - \arcsin\frac{V_{ц.у}}{V_н}, \qquad (3)$$

где $V_{ц.у}$ – скорость цели при уклонении от обнаружения.

Анализ выражений (1) –(3) позволяет сделать следующие выводы.

1. Распределение наблюдателей по участкам поиска, площадь которых пропорциональна их поисковой производительности, приводит к тому, что каждый наблюдатель осуществляет поиск не в составе единой поисковой системы, а самостоятельно, независимо от остальных наблюдателей, вследствие чего преимущество группового поиска в значительной степени утрачивается. Взаимодействие при поиске осуществляется только в пределах участка (если наблюдатель групповой). Взаимодействие между наблюдателями, ведущими поиск на смежных участках, заключается, как правило, только в организации смены участков.

2. Если

$$d_{цi} > d_{нi}\frac{V_{ц.у}}{V_{нi}}, \qquad (4)$$

что чаще всего и имеет место в современных условиях, особенно в ситуации, когда цель получает информацию о поисковых силах от внешних источников, интенсивность поиска для случая идеально уклоняющейся цели, определяемая выражением (2), равна нулю. Если условие (4) выполняется для всех наблюдателей группы, общая вероятность обнаружения цели в районе будет равна нулю независимо от количества наблюдателей в группе, так как все наблюдатели осуществляют поиск независимо друг от друга. Очевидно, что решить задачу оптимального поиска или хотя бы оценить его эффективность в этом случае можно только опосредованно.

Оптимизация поиска объектов на основе теории дифференциальных игр

Разрешение указанного противоречия возможно как за счет расширения собственных положений ТППО, так и путем привлечения элементов других теорий к решению вновь возникающих задач. Вышеуказанные ограничения ТППО переводят ситуацию поиска в разряд конфликтных, т. е. таких, где участвующие стороны имеют несовпадающие интересы. Математической теорией конфликтных ситуаций является теория игр. Привлечение аппарата теории игр к решению задач поиска привело к появлению целого класса задач на стыке теории игр и теории поиска – игр поиска. Значительная часть поисковых ситуаций в условиях противодействия цели может быть представлена в виде непрерывной бесконечной антагонистической игры (иначе – дифференциальной игры) с неполной информацией.

Метод оптимизации поиска подвижных объектов на основе теории дифференциальных игр базируется на следующих основных положениях.

1. Оптимальная смешанная стратегия объекта поиска определяется целью его действий в конкретной поисковой ситуации. Цель действий объекта поиска всегда противоположна цели действий наблюдателя. В любой момент времени поиска объект поиска действует наименее выгодным для наблюдателя образом.

2. Игра поиска подвижного объекта $\bar{\Gamma}$ может быть сведена к эквивалентной ей игре поиска неподвижного объекта Γ.

3. Игра $\bar{\Gamma}$ является эквивалентной игре Γ, если:

– значения игры равны, т. е. val Γ = val $\bar{\Gamma}$;

– множества оптимальных стратегий игроков совпадают, т. е. $u(t) = u^*(t)$ и $\nu = \nu^*$, где $u(t)$, ν – оптимальные стратегии наблюдателя и цели в игре поиска подвижного обьекта; $u^*(t)$, ν^* – соответствующие им оптимальные стратегии в эквивалентной игре поиска неподвижного обьекта.

4. Оптимальной стратегией цели в игре поиска неподвижного объекта является размещение в области G по равновероятному закону. В игре поиска подвижного объекта оптимальная стратегия цели будет определяться условиями конкретной задачи.

5. Система фазовых координат при решении задачи поиска должна строиться таким образом, чтобы пространство поиска G' в данной системе координат при условии применения наблюдателем оптимальной смешанной стратегии u^* включало в себя все пространства параметров цели G_i', $i = 1...\infty$, соответствующие всем возможным стратегиям цели, иначе, чтобы выполнялось условие $G_i' \subset G'$.

Основная идея метода заключается в нахождении решений игр поиска подвижных объектов путем сведения последних к эквивалентным играм поиска неподвижных объектов, имеющим более простое решение. Таким образом, решение задачи оптимального поиска подвижной цели на плоскости сводится к решению задачи оптимального поиска неподвижного «образа» цели в пространстве параметров.

Последовательность применения метода для решения задач оптимизации поиска подвижных объектов такова:

1 этап. Постановка задачи поиска. Включает в себя:

а) определение цели действий объекта поиска;

б) определение закона распределения вероятности нахождения цели в районе поиска (на рубеже, в полосе);

в) определение характеристик движения наблюдателя и цели и возможности по их взаимному обнаружению;

г) определение дополнительных условий задачи (возможность выхода цели из района поиска и др.).

2 этап. Представление задачи оптимизации поиска в виде дифференциальной игры (формальное определение дифференциальной игры поиска подвижного объекта). Включает в себя:

а) определение множеств стратегий (оптимальных смешанных стратегий) наблюдателя и цели, соответствующих поисковой ситуации;

б) выбор системы фазовых координат, позволяющей преобразовать игру поиска подвижного объекта в эквивалентную игру поиска неподвижного объекта;

в) определение динамики игроков и области поиска в выбранной системе фазовых координат.

3 этап. Решение дифференциальной игры поиска неподвижного объекта в фазовых координатах. Включает в себя:

а) определение значения игры, т. е. значения функции выигрыша в ситуации равновесия;

б) определение условия оптимальности стратегии наблюдателя;

в) определение аналитических зависимостей, характеризующих траектории наблюдателя и цели в фазовых координатах.

4 этап. Преобразование полученных аналитических зависимостей из системы фазовых координат в систему декартовых (полярных) координат на плоскости; нахождение числовых характеристик траектории наблюдателя на основе полученных выражений.

В качестве критерия оптимальности поиска принимается цена игры – минимальное время обследования области возможного положения цели при заданной вероятности ее обнаружения или максимум обследуемой площади за определенное время. Решением дифференциальной игры поиска является оптимальная траектория наблюдателя.

Рассмотрим некоторые решения задачи вторичного поиска на основе теоретико-игрового подхода.

Вторичный поиск («поиск по вызову»)

В работах по теории поиска [1, 4, 6] широко освещен так называемый вторичный поиск, когда установлен факт присутствия цели в районе, но ее место определено с ошибкой. При этом предполагается, что скорость цели известна достоверно, а направление движения распределяется равновероятно по всему горизонту. Плотность вероятности места цели относительно исходной точки обнаружения через время t после потери контакта с ней, соответствующее указанной выше гипотезе о характере движения, цели имеет вид:

$$w(r,t) = \frac{1}{2\pi\sigma^2}\mathbf{e}^{-\frac{r^2+V_ц^2t^2}{2\sigma^2}} I_0\left(\frac{rV_ц t}{\sigma^2}\right), \qquad (5)$$

где σ – среднеквадратическая погрешность места цели; $I_0\left(\frac{rV_ц t}{\sigma^2}\right)$ – функция Бесселя нулевого порядка от мнимого аргумента.

Оптимальной стратегией наблюдателя, позволяющей ему осуществлять поиск, постоянно находясь в области максимума плотности вероятности места цели $w(r, t)$, как показано в вышеуказанных работах, является расходящаяся логарифмическая спираль, описываемая уравнением

$$R(t) = V_ц t_з \mathbf{e}^{\frac{\varphi}{\sqrt{\frac{1}{m^2}-1}}}, \qquad (6)$$

где $R(t)$ – расстояние от исходной точки до точки максимума величины $w(r, t)$;

$t_з$ – время от момента обнаружения до момента начала поиска;

ц – координата наблюдателя в полярной системе при его движении по логарифмической спирали.

Способу обследования водного пространства по расходящейся логарифмической спирали присущи следующие существенные недостатки [5]:

– эффективный поиск возможен только лишь при соблюдении гипотезы о постоянстве курса и скорости уклоняющегося объекта в течение всего времени поиска, а также соответствии фактических параметров движения объекта, принятым в гипотезе;

– учитывает только случайный характер курса цели, но не учитывает случайное распределение места цели в зависимости от средства и способа ее первичного обнаружения;

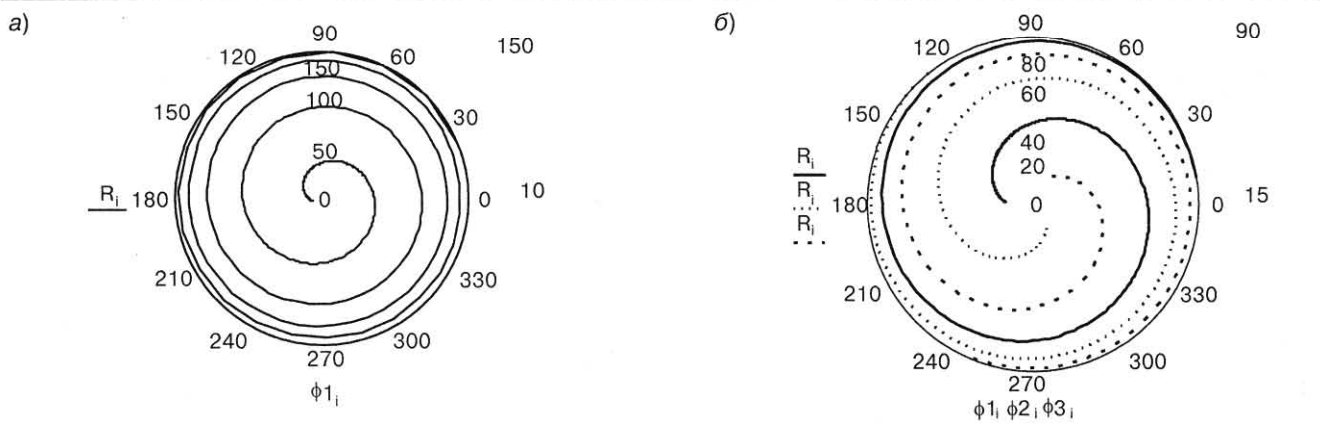
– невозможен поиск объектов, скорость которых при уклонении больше скорости наблюдателя.

С целью преодоления указанных противоречий сформулируем задачу вторичного поиска следующим образом: в области неопределенности, имеющей форму круга радиусом R_0, находится цель, максимальная скорость цели $V_ц$. Характер задач, выполняемых целью в районе поиска, позволяет ей покинуть его в любой момент времени поиска. Наблюдатель осуществляет поиск цели на скорости $V_н > V_ц$, имея при этом дальность действия средств обнаружения цели, равную $d_н < d_ц$, где $d_ц$ – дальность, на которой цель обнаруживает наблюдателя. Задача наблюдателя состоит в том, чтобы обнаружить цель в кратчайшее время.

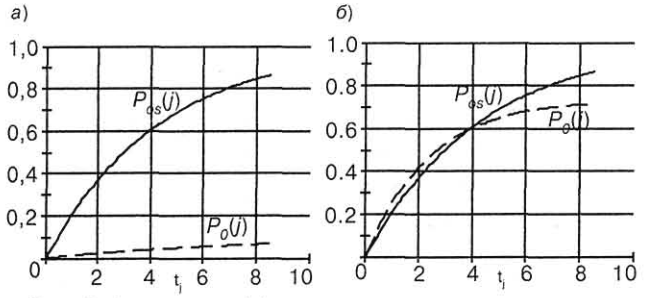
Применение метода оптимизации на основе теории дифференциальных игр к решению задачи вторичного поиска позволяет сделать следующие выводы.

1. Если место цели в момент начала поиска распределено в области неопределенности по равновероятному закону, оптимальная траектория наблюдателя, являющаяся результатом решения данной задачи на основе теоретико-игрового подхода, представляет собой сходящуюся спираль, результаты компьютерного моделирования которой представлены на рис. 1, *а*, *б*. Вектор скорости цели в каждой точке спирали направлен под углом $\mu(t)$ к радиусу спирали. Значение $\mu(t)$ определяется выражением:

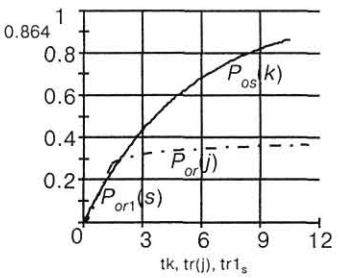
$$\mu(t) = \arccos\frac{k(t)\sqrt{k(t)^2+1-m^2}-m}{k(t)^2+1}, \qquad (7)$$



■ ***Рис. 1.*** *Поиск по сходящейся спирали одиночным наблюдателем (а) и группой из трех наблюдателей (б)*



■ ***Рис. 2.*** *Сравнение эффективности поиска по расходящейся логарифмической ($P_0(j)$) и по сходящейся ($P_{os}(j)$) спирали*



■ ***Рис. 4.*** *Сравнение эффективности вторичного поиска по сходящейся (P_{os}) и расходящейся (P_{or}) спирали при равновероятном распределении цели*

где

$$k(t)=\frac{nd_{\text{н}}}{\pi R(t)}; \qquad (8)$$

$R(t)$ – текущий радиус спирали; m – соотношение скоростей цели и наблюдателя, $m=\frac{V_{\text{ц}}}{V_{\text{н}}}$.

Способ поиска, реализующий траекторию поиска по сходящейся спирали, будем именовать способом (стратегией) ПСС. Условие оптимальности стратегии ПСС определяется выражением:

$$R_{\max}\le\frac{nd_{\text{н}}}{m\pi}, \qquad (9)$$

где $R_{\max}$ - предельное значение радиуса;
n – число наблюдателей.

Число наблюдателей, необходимое для реализации стратегии ПСС, определяется выражением:

$$n_{\text{необх}}\ge\frac{m\pi R_0}{d_{\text{н}}}, \qquad (10)$$

где R_0 – радиус района поиска. Сравнительная оценка эффективности поиска традиционным способом (по расходящейся логарифмической спирали) и поиска по сходящейся спирали приведена на рис.2, *а*, *б*.

Анализ результатов математического моделирования данной поисковой ситуации позволяет сделать вывод о том, что эффективность ПСС тем выше эффективности поиска по расходящейся логарифмической спирали, чем большее упреждение в обнаружении имеет цель над наблюдателем (на рис. 2, *б* изображен случай, когда цель практически не имеет такого упреждения), т. е. чем меньшее значение имеет интенсивность поиска уклоняющейся цели, определяемая выражениями (2) – (3).

2. Если место цели в момент начала поиска распределено в области неопределенности по нормальному закону, оптимальной траекторией наблюдателя является рассмотренная выше сходящаяся спираль или расходящаяся спираль, характеристики которой также определяются выражениями (7) – (8). Способ поиска по расходящейся спирали будем называть способом (стратегией) ПРС. Его отличие от ПСС заключается только в направлении движения наблюдателя – не от периферии области неопределенности к ее центру, а наоборот. Предельный радиус области неопределенности, которая может быть обследована с использованием стратегии ПРС, определяется выражением

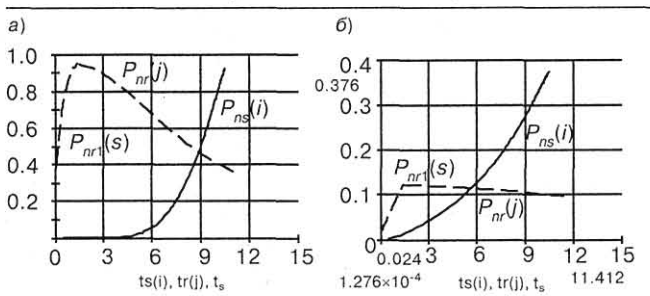
$$R_{\max}\le\frac{nd_{\text{н}}}{m_1\pi}, \qquad (11)$$

где

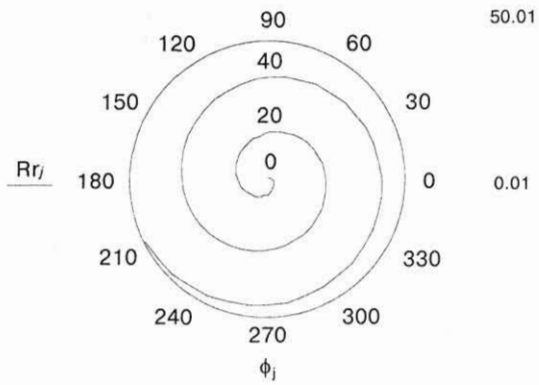
$$m_1=\frac{2m}{\sqrt{1-m^2}}. \qquad (12)$$

3. Эффективность вторичного поиска при нормальном законе распределения координат цели определяется рядом факторов. Кроме величин, от которых эффективность поиска зависит в любой поисковой ситуации (скорость цели и наблюдателя, дальность обнаружения наблюдателя и др.), при вторичном поиске большую роль играет точность знания места цели в момент начала поиска, определяемая величиной СКО σ_0. Если значение σ_0 мало, то более эффективным можно считать способ ПРС (рис. 3, *а*). Уменьшение вероятности обнаружения цели с течением времени объясняется расширением ОВПЦ, сопровождающимся уменьшением плотности вероятности нахождения цели в необследованной области. Если значение σ_0 велико (рис. 3, *б*), способ ПСС имеет преимущество над ПРС.

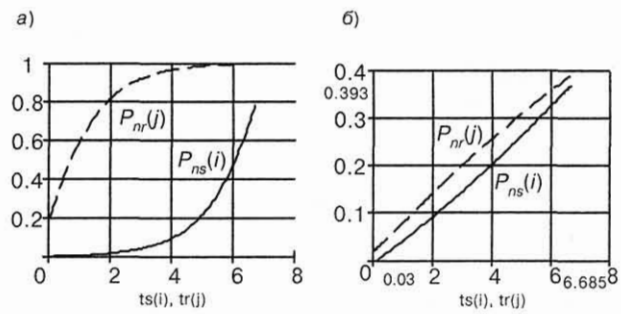
4. При равновероятном характере распределения цели в районе поиска ПСС имеет бесспорное преимущество над ПРС (рис. 4). Линия излома графика функции $P_{or}(t)$ (вероятность обнаружения цели при ПРС) соответствует моменту достижения наблюдателем расстояния $R_{\max}$ от центра района поиска. На графике видно, что даже в том случае, когда цель находится поблизости от центра района по-



■ ***Рис. 3.*** *Сравнение эффективности вторичного поиска по сходящейся (P_{ns}) и расходящейся (P_{nr}) спирали при нормальном распределении цели при σ_0 = 10 миль (а) и σ_0 = 50 миль (б)*



■ ***Рис. 5.*** *Траектория наблюдателя при поиске неподвижной (малоподвижной) цели*



■ ***Рис. 6.*** *Сравнение эффективности поиска неподвижной цели по сходящейся (P_{ns}) и расходящейся (P_{nr}) спирали при нормальном распределении*

иска, динамика роста вероятности обнаружения практически одинакова.

5. *Если цель неподвижна ($m=0$) или малоподвижна ($m<<1$)*, траектория наблюдателя будет иметь вид спирали Архимеда (рис. 5).

Если распределение места цели в районе характеризуется равновероятным законом, способы ПСС и ПРС имеют одинаковую эффективность. Если цель распределена по нормальному закону, оптимальным способом поиска будет ПРС независимо от значения σ_0 (рис. 6).

Заключение

Представленный в статье подход позволяет решать задачу оптимизации поиска для случаев, не имеющих прямых аналитических решений в рамках ТППО. При этом полученное решение позволяет решить задачу не только для однородных наблюдателей, но и для разнородных сил поиска.

Литература

1. **Абчук В. А., Суздаль В. Г.** Поиск объектов. – М.: Сов. радио, 1977. – 333 с.
2. **Динер И. Я.** Исследование операций. – Л.: ВМА, 1969. – 604 с.
3. **Хеллман О.** Введение в теорию оптимального поиска. Пер. с англ. / Под ред. Моисеева Н. П. – М.: Наука, 1985. –284 с.
4. **Попович В. В.** Моделирование, оценка эффективности и оптимизация систем наблюдения ВМФ (теория поиска подвижных объектов). – СПб.: ВМА, 2000. – 424 с.
5. **Чаусов Ф. С., Михайлов В. А.** Способ поиска объектов по сходящейся архимедовой спирали (Материалы изобретения). – СПб.: ВМА, 2002. – 33 с.
6. **Koopman B. O.** The theory of search. 1. / Operat.Res. – 1956. – 4. – P. 324–346.
7. **Koopman B. O.** The theory of search. 2. / Operat.Res. – 1956. – 4. – P. 503–531.
8. **Koopman B. O.** The theory of search. 3. / Operat.Res. – 1956. – 5. – P. 613–626.

УДК 621.865.8

МЕТОДЫ КОРРЕКЦИИ ПРОСТРАНСТВЕННЫХ ИСКАЖЕНИЙ ИЗОБРАЖЕНИЙ ПЛОСКИХ ОБЪЕКТОВ В УСЛОВИЯХ ДЕЙСТВИЯ ПОЛНОЙ АФФИННОЙ ГРУППЫ ПРЕОБРАЗОВАНИЙ

Н.В.Соловьев,
старший преподаватель
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Рассмотрены применяемые в системах распознавания робототехнических комплексов методы коррекции пространственных искажений изображений, описываемых полной аффинной группой преобразований. Основное внимание уделено методам, позволяющим распознавать объекты в процессе компенсации искажений. Предложен метод определения параметров преобразования, основанный на последовательном переборе ограниченного числа характерных точек предварительно центрированного изображения объекта.

The methods of a correction of spatial aliasings of images are considered. The distortions are circumscribed by full affine group of transformations. The methods allow to recognize objects during indemnification of distortions. Area of application algorithm – robot vision.

Введение

Задача распознавания объектов и определения параметров их положения в пространстве по изображениям возникает при разработке адаптивных промышленных роботов и робототехнических комплексов различного назначения. Подобные устройства состоят из системы распознавания, системы управления, приводов и исполнительных механизмов. Система распознавания робота после получения, обработки и анализа сенсорной информации поставляет системе управления данные об окружающей среде, на основании анализа которых последняя вырабатывает управляющие сигналы для приводов исполнительной системы. Можно сказать, что именно система распознавания, являясь непременной составляющей адаптивного робота (робототехнического комплекса), превращает программно управляемое устройство в робота, совершающего целесообразные действия в условиях изменяющейся внешней среды.

Следует отметить, что первые попытки решения проблемы машинного видения, создания систем технического зрения и компьютерного анализа изображений сцен относятся к концу 60-х годов XX века, что связано с появлением возможности ввода в ЭВМ изображений. За прошедшее время появилось огромное количество литературы по проблемам технического зрения вообще и роботов в частности, как фундаментального характера, так и посвященных решению отдельных задач. Выпущено множество тематических сборников, обзоров, материалов конференций, посвященных различным аспектам технического зрения и смежным областям. Однако, несмотря на достигнутые успехи в решении отдельных проблем, системы технического зрения по своей универсальности все еще далеки от аналогичных систем в живой природе [1].

Основные трудности в создании систем распознавания объектов по их изображениям связаны с тем, что разные изображения одного и того же объекта в подавляющем большинстве случаев существенно отличаются друг от друга. Данное обстоятельство практически не позволяет подобрать для проведения классификации распознаваемого образа признаки, инвариантные к этим отличиям. Причины, вызывающие указанные отличия (или искажения, если считать одно из изображений объекта эталонным), можно разделить на три группы:

1) помехи, обусловленные несовершенством аппаратуры для получения, передачи, хранения и обработки изображений;

2) изменение с течением времени освещенности объектов сцены, изображение которой анализируется;

3) изменение взаимного положения объектов сцены относительно друг друга и устройства получения изображения (камеры).

Методы компенсации технических помех хорошо разработаны. Применение тестовых изображений и методов статистической обработки позволяет в большинстве случаев добиться вполне приемлемых результатов.

Компенсация искажений, вызванных второй причиной, более трудна, так как изменение освещенности приводит не только к глобальным изменениям яркости изображения, но и к изменению яркости изображений отдельных объектов, появлению и смещению бликов и теней. Однако применительно к промышленным роботам трудности компенсации данной причины можно существенно снизить, подобрав соответствующее освещение сцены, например, используя бестеневые матовые светильники для освещения объектов сцены.

По признанию большинства авторов [2, 3], основные сложности вызывают именно пространственные искажения. Действительно, даже при отсутствии технических помех и постоянном освещении сцены любое изменение взаимного расположения объектов и камеры приводит к изменению изображения.

Известно [2], что если в результате перемещения объектов или камеры не происходит перекрытие их изображений при изменении изображения сцены, то это изменение может быть описано групповым преобразованием. При центральном проецировании, когда расстояние от объектов до камеры сравнимо с фокусным расстоянием объектива камеры, преобразование описывается проективной группой. При параллельном проецировании, когда объекты находятся на значительном удалении от камеры, преобразование описывается аффинной группой, являющейся подгруппой проективной группы преобразований.

Постановка задачи

Пусть в пространстве U действует транзитивная группа преобразований G, каждый элемент $g \in G$ которой может быть представлен в виде произведения некоторых элементов ее подгрупп G_1, G_2, ..., G_n, т. е. $g = g_1 * g_2 * ... * g_n$; $g_i \in G_i$.

Тогда каждая точка X с координатами (x,y) исходного изображения (эталонного образа) преобразуется элементом g:

$$X' = g * X = g_1 * g_2 * ... * g_n * X,$$

где X':(x',y') – координаты соответствующей точки преобразованного или искаженного изображения. Таким образом, при наличии только пространственных искажений яркость точки X' искаженного изображения полагается равной яркости точки X исходного изображения (рис.1).

Если из каких-либо соображений будет найден элемент g_i (или g_i^{-1}), тогда, подвергнув искаженное изображение преобразованию элементом g_i^{-1}, мы обеспечим «сближение» искаженного и эталонного изображений. Продолжая процесс последовательной нейтрализации подгрупп можно либо полностью устранить искажения (распознать все объекты и определить их параметры положения), либо существенно упростить задачу и свести ее к известному решению.

Аффинная группа имеет вид:

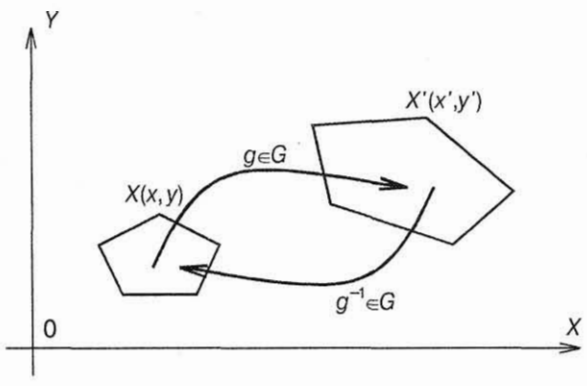


Рис. 1. *Представление пространственного искажения групповым преобразованием*

$$\begin{aligned} x' &= a_1 x + a_2 y + b_1, \\ y' &= a_3 x + a_4 y + b_2, \end{aligned} \qquad (1)$$

где x, y, x', y' – исходные и преобразованные аффинной группой координаты точки изображения; $a_1, ..., a_4, b_1, b_2$ – параметры аффинной группы. Очевидно, что, зная параметры, можно легко осуществить обратное преобразование $X = g^{-1} * X'$, тем самым приведя изображение к эталонному виду. Дальнейшее распознавание не представляет сложностей и может быть выполнено, например, методом маски.

Из уравнения (1) видно, что для нахождения шести параметров аффинной группы необходимо знать как минимум координаты трех не лежащих на одной прямой точек на исходном и искаженном изображениях и соответствие между ними. Тогда, решив легко получающуюся из (1) систему шести уравнений с параметрами $a_1, ..., a_4, b_1, b_2$ в качестве неизвестных и координатами соответствующих точек изображений, можно получить искомое преобразование g^{-1}. Рассмотренные далее методы в той или иной степени используют данный подход.

Метод последовательной компенсации подгрупп аффинной группы

Разработанный И. Л. Ерошем [4] метод позволяет совместить процесс компенсации пространственных искажений с собственно распознаванием, т. е. отнесением объекта к одному из известных эталонов.

Элемент g аффинной группы может быть представлен в виде:

$$g = g_1 * g_2, \qquad (2)$$

где $g_1 \in G_1$ – группа сдвигов вдоль координатных осей вида $\begin{cases} x' = x + b_1 \\ y' = y + b_2 \end{cases}$; $g_2 \in G_2$ – аффинная группа без сдвига с матрицей преобразования вида $\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$.

Если изображения эталонов распознаваемых объектов центрированы (центр формы совпадает с началом координат, что всегда можно сделать на этапе обучения системы автоматического распознавания), то, вычислив центр формы искаженного изображения, получим параметры b_1 и b_2. После переноса начала координат в этот центр будут нейтрализованы параметры подгруппы G_1. Следовательно, частично преобразованное, распознаваемое изображение будет связано с неизвестным пока эталоном преобразованием элементом группы G_2. Этот элемент g_2 может быть представлен в виде

$$g_2 = g_2' * g_2'',$$

где g_2' – элемент группы масштабного преобразования с некоторой матрицей, например, вида $\begin{pmatrix} r & 0 \\ 0 & 1 \end{pmatrix}$; g_2'' – элемент аффинной унимодулярной группы с матрицей вида $\begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}$,

$$c_1c_4 - c_2c_3 = 1. \quad (3)$$

Значения параметров r, а также c_1, c_2, c_3, c_4 могут быть найдены через параметры аффинной группы без сдвига:

$$rc_1 = a_1,\ rc_2 = a_2,\ c_3 = a_3,\ c_4 = a_4,\ \frac{a_1a_4 - a_2a_3}{r} = 1,$$

откуда $r = a_1a_4 - a_2a_3$.

Обратная матрица, которая устраняет масштабную часть искажений, имеет вид $\begin{pmatrix} \frac{1}{a_1a_4 - a_2a_3} & 0 \\ 0 & 1 \end{pmatrix}$. Значение r неизвестно, однако его можно определить из сравнения площадей распознаваемого изображения и изображения эталонов. Для этого все эталонные изображения необходимо отмасштабировать так, чтобы их площади были равны между собой и имели бы величину S_0. Вычислив величину S распознаваемого изображения, можно найти коэффициент масштаба $r = S/S_0$.

После преобразования распознаваемого изображения элементом вида

$$(g_2')^{-1} = \begin{pmatrix} 1/(S/S_0) & 0 \\ 0 & 1 \end{pmatrix}$$

оно оказывается связанным с одним из эталонов (по-прежнему неизвестным) преобразованием вида (3) или

$$\begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ c_3/c_1 & 1 \end{pmatrix}\begin{pmatrix} c_1 & c_2 \\ 0 & 1/c_1 \end{pmatrix}.$$

Отношение c_3/c_1 можно определить следующим образом. Предположим, что известна одна характерная точка на каждом эталоне, и можно определить ее на исследуемом изображении. Под характерной точкой будем понимать такую точку, положение которой определяется однозначно независимо от заданных преобразований (в данном случае – аффинных). К сожалению, общий метод для выбора характерных точек на произвольных изображениях неизвестен. Для многоградационных растровых изображений в качестве характерной точки можно выбрать центр масс или центр яркости изображения, который вычисляется следующим образом:

$$x_c = \frac{1}{N}\sum_{i=1}^{N} x_i n_i,\ y_c = \frac{1}{N}\sum_{i=1}^{N} y_i n_i,$$

где n_i – числовое значение (вес), приписанное пикселу i, например, код яркости пиксела; N – число пикселов изображения. Для таких изображений при переносе начала координат в центр формы центр яркости оказывается «сдвинутым», т. е. может не совпадать с началом координат.

Для изображений каждого из эталонов поместим на этапе обучения центр яркости на ось X (путем поворота вокруг начала координат). Тогда, вычислив центр яркости распознаваемого изображения, получим

$$\begin{pmatrix} x_c' \\ y_c' \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}\begin{pmatrix} x_c \\ 0 \end{pmatrix} = \begin{pmatrix} c_1x_c \\ c_3x_c \end{pmatrix}.$$

Из полученного выражения находим:

$$\frac{c_3}{c_1} = \frac{y_c'}{x_c'}.$$

Тогда можно вычислить матрицу

$$\begin{pmatrix} 1 & 0 \\ c_3/c_1 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 \\ -c_1/c_3 & 1 \end{pmatrix}.$$

После преобразования распознаваемого изображения этой матрицей получим, что некоторый неизвестный эталон связан с распознаваемым изображением групповым преобразованием с матрицей

$$\begin{pmatrix} c_1 & c_2 \\ 0 & 1/c_1 \end{pmatrix}. \quad (4)$$

Таким образом, от полной аффинной группы с шестью неизвестными параметрами мы перешли к группе с двумя неизвестными параметрами путем нейтрализации некоторых групповых параметров.

Теперь предположим, что нам известен эталон с номером p, которому соответствует распознаваемое изображение. Его характерная точка имеет координаты $\{x_c^p, 0\}$. После преобразования матрицей (4) координаты этой точки будут иметь значения:

$$\begin{pmatrix} x_c' \\ y_c' \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ 0 & 1/c_1 \end{pmatrix}\begin{pmatrix} x_c^p \\ 0 \end{pmatrix} = \begin{pmatrix} c_1x_c^p \\ 0 \end{pmatrix},$$

откуда

$$c_1 = \frac{x_p'}{x_p}. \quad (5)$$

Представим матрицу (4) в следующем виде:

$$\begin{pmatrix} c_1 & c_2 \\ 0 & 1/c_1 \end{pmatrix} = \begin{pmatrix} 1 & c_1c_2 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} c_1 & 0 \\ 0 & 1/c_1 \end{pmatrix}.$$

Если известно, какому эталону соответствует распознаваемое изображение, то, преобразуя изображение эталона матрицей вида $\begin{pmatrix} c_1 & 0 \\ 0 & 1/c_1 \end{pmatrix}$, получим изображение, связанное с распознаваемым матрицей вида $\begin{pmatrix} 1 & c_1c_2 \\ 0 & 1 \end{pmatrix}$, в которой только один параметр c_2 неизвестен.

Для каждого эталона, используя соотношение (5), найдем значения c_1^p; $p = 1, 2, \ldots, P$, где P – число эталонов. Каждый p-й эталон преобразуем с помощью матрицы $\begin{pmatrix} c_1^p & 0 \\ 0 & 1/c_1^p \end{pmatrix}$, тогда получим, по-прежнему, P искаженных эталонов, один из которых (пока неизвестный) связан с распознаваемым изображением матрицей вида

$$\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}, \quad (6)$$

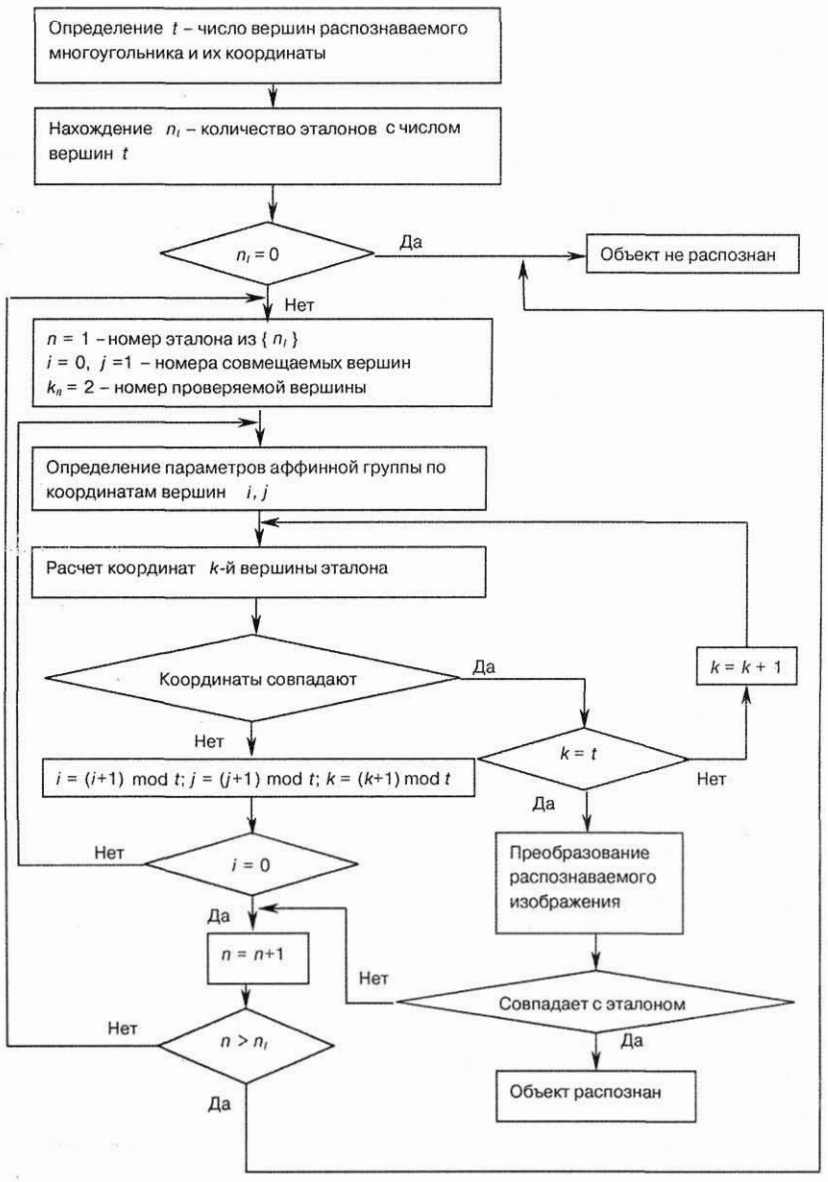
где $a = c_1^pc_2$.

Преобразование матрицей (6) не меняет координаты y любой точки, а координата x меняется линейно $x' = x + ay$. При таком преобразовании расстояние между любыми двумя точками, имеющими одинаковые значения y, не меняется. Из этих соображений легко находятся значения p и c_2.

Таким образом, можно узнать эталон, которому соответствует искаженное изображение, и определить все параметры неизвестного преобразования. Тогда

$$g_2 = \begin{pmatrix} S/S_0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ c_1/c_3 & 1 \end{pmatrix} \begin{pmatrix} 1 & c_1^p c_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c_1^p & 0 \\ 0 & 1/c_1^p \end{pmatrix}.$$

Приведенный алгоритм позволяет распознать объект по его искаженному изображению и определить все шесть параметров аффинного преобразования. Основное ограничение области применения алгоритма – необходимость несовпадения центра формы с центром яркости на изображении каждого эталона. На бинарных изображениях они всегда совпадают. Для многоградационных изображений необходимо практически точное совпадение яркостей всех точек изображения эталона и распознаваемого объекта, что на практике наблюдается крайне редко. Кроме этого, желательно, чтобы координаты центров яркости эталонов существенно отличались друг от друга.



■ ***Рис. 2.*** *Алгоритм распознавания перебором вершин контура*

Метод последовательного перебора характерных точек изображения

Аналогично рассмотренному методу на первом этапе предполагается провести центрирование изображения по координатам центра формы, тем самым компенсировав подгруппу g_1 из (2).

Для нахождения параметров аффинной группы без сдвига g_2: $\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$ достаточно выбрать две характерные точки на изображении эталона, лежащие на прямой, не проходящей через начало координат, и найти соответствующие им точки на изображении распознаваемого объекта. Если представить контур изображения распознаваемого объекта и каждого эталона в виде замкнутой ломаной линии, то в качестве характерных точек можно использовать ее вершины. Пусть число таких вершин на изображении p-го эталона $s_p \in S$; $S = \{s_1, ..., s_P\}$, где S – множество вершин эталонов, P – число эталонов. Представим контур изображения распознаваемого объекта также в виде ломаной линии. Как известно [2], аффинное преобразование переводит прямую линию в прямую, следовательно, число вершин на изображении распознаваемого объекта $s_x \in S$ и последовательность вершин сохраняется. Если в S нет одинаковых элементов, то, очевидно, проблема распознавания решена. Остается найти параметры элемента группы g_2. Дальнейший алгоритм, представленный на рис. 2, предполагает перебор вершин контура изображения распознаваемого объекта при конечном числе возможных сочетаний.

Возьмем две любые последовательно расположенные вершины $\{i, j\}$ на контуре изображения эталона. В предположении, что им соответствуют две последовательно рас-

положенные вершины на контуре изображения распознаваемого объекта $\{l, k\}$, составим систему уравнений

$$\begin{cases} x_l' = a_1 x_i + a_2 y_i; \\ y_l' = a_3 x_i + a_4 y_i; \\ x_k' = a_1 x_j + a_2 y_j; \\ y_k' = a_3 x_j + a_4 y_j, \end{cases} \quad (7)$$

где $x_i, y_i, x_j, y_j, x_l', y_l', x_k, y_k'$ – координаты соответствующих точек. Решив систему (7), определим параметры $a_1, \ldots, a_4$ и преобразуем эталонное изображение группой g_2. Если после сравнения преобразованного и распознаваемого изображений совпадение будет отсутствовать, то следует повторить описанные действия для следующей пары вершин контура изображения эталона. Рано или поздно будет получено совпадение изображений, так как число вершин контура конечно.

Если в S есть одинаковые элементы, т. е. среди эталонов есть совпадающие по числу вершин, то следует разбить S на непересекающиеся подмножества эталонов с одинаковым числом вершин контура и выполнять описанные действия по перебору вершин контура последовательно для всех эталонов подмножества до тех пор, пока не будет получено совпадение изображений. Естественно предполагается, что распознаваемый объект соответствует одному из имеющихся эталонов.

Практически при переборе пар вершин эталона не требуется подвергать преобразованию g_2 все эталонное изображение (см. рис. 2). Достаточно преобразовывать только координаты вершин контура, прерывая процесс при первом несовпадении и переходя к следующей паре вершин или эталону, если вершины данного эталона закончились. Преобразовывать все изображение следует только в случае полного совпадения вершин. Последнее связано с тем, что эталоны могут иметь одинаковый контур, но отличаться внутренней структурой, например, иметь различное число и форму отверстий.

Основное ограничение рассмотренного выше алгоритма – изображения эталонов не должны преобразовываться друг в друга аффинной группой. На практике это означает, что треугольные эталоны произвольной формы и размеров должны отличаться внутренней структурой, а четырехугольные эталоны должны иметь хотя бы две непараллельные стороны. Другое ограничение – эталоны с плавным криволинейным контуром практически невозможно представлять в виде соответствующих многоугольников на изображениях как эталонов, так и распознаваемого объекта.

Естественно, при компьютерной реализации рассмотренного алгоритма условие точного равенства координат и совпадения изображений необходимо заменить некоторой мерой близости с экспериментально подобранным порогом, определяющим наличие совпадения. Данное требование вызвано дискретным (пиксельным) принципом хранения и обработки изображения в компьютере, что приводит к погрешностям при округлении координат.

При компьютерной реализации алгоритма возникает еще одна проблема – нахождение координат вершин многоугольника. Существующие алгоритмы поиска вершин [5] наиболее эффективны для углов линий контура в искомой вершине, близких к прямым углам. При приближении угла контура при вершине к 180° или к 0° точность определения координат вершины резко падает, так как в таких случаях трудно решить, что найдено – вершина линии контура или случайное отклонение от прямой, обусловленное помехами предварительной обработки изображения. Компьютерное моделирование данного алгоритма показало, что с учетом этих ограничений алгоритм достаточно эффективен для многоугольников, угол контура которых при вершине находится в пределах 60° – 120°.

Литература

1. **Марр Д.** Зрение: Информационный подход к изучению представления и обработки зрительных образов/Пер. с англ. – М.: Радио и связь, 1987. – 400 с.
2. **Путятин Е. П., Аверин С. И.** Обработка изображений в робототехнике. – М.: Машиностроение, 1990. – 320 с.
3. **Хорн Б. К. П.** Зрение роботов/Пер. с англ. – М.: Мир, 1989. – 487 с.
4. **Ерош И. Л., Игнатьев М. Б., Москалев Э. С.** Адаптивные робототехнические системы: Методы анализа и системы обработки изображений. – Л.: Изд-во ЛИАП, 1985. – 144 с.
5. Техническое зрение роботов / В. И. Мошкин, А. А. Петров, В. С. Титов и др.; Под ред. Ю. Г. Якушенкова. – М.: Машиностроение, 1990. – 272 с.

УДК 681.516.7.015.2

ПРОГНОЗИРОВАНИЕ ДВИЖЕНИЯ ОБЪЕКТОВ

Б. А. Коптев,
канд. техн. наук
А. К. Розов,
д-р техн. наук
А. Ф. Романовский,
канд. техн. наук

Фильтрация параметров движущихся объектов может быть осуществлена с использованием аппарата стохастических дифференциальных уравнений. Получаемые в результате решения уравнений оценки параметров позволяют определить прогнозируемое движение объекта. Приводится пример фильтрации и прогноза.

In this article filtering of movement are based on stochastic difference equation. A possibility to applied its using for prognostication movement discuss. The example of applicability is considered.

Имеется большое число работ, описывающих внешнетраекторные измерения объектов, и почти отсутствуют работы, представляющие прогноз их траекторий. В условиях ошибок и устаревания данных, полученных с помощью средств внешнетраекторных измерений, область возможных координат объекта оказывается достаточно обширной.

Неразработанность методов прогноза, возможно, объясняется тем, что пока не найдены методы, позволяющие оптимизировать решение задачи в целом – обеспечить малые ошибки прогноза и малое время на его выполнение. Иначе говоря, пока не удается найти правило δ^*, минимизирующее средние потери

$$R_t(\delta^*) = \inf_{\delta}\left\{cM\nu + M\left[Z_t - \bar{Z}_t^{-2}\right]\right\}, \quad \nu \le t,$$

в которых ν – момент прекращения наблюдений; Z_t и $\bar{Z}_t$ – действительные и прогнозируемые координаты.

В настоящее время возможен другой подход, предполагающий использование имеющихся средств внешнетраекторных измерений и дополнение их процедурами фильтрации и прогноза. Насколько такое разделение снижает возможности прогноза по сравнению с оптимизацией комплекса в целом – должны показать дальнейшие исследования.

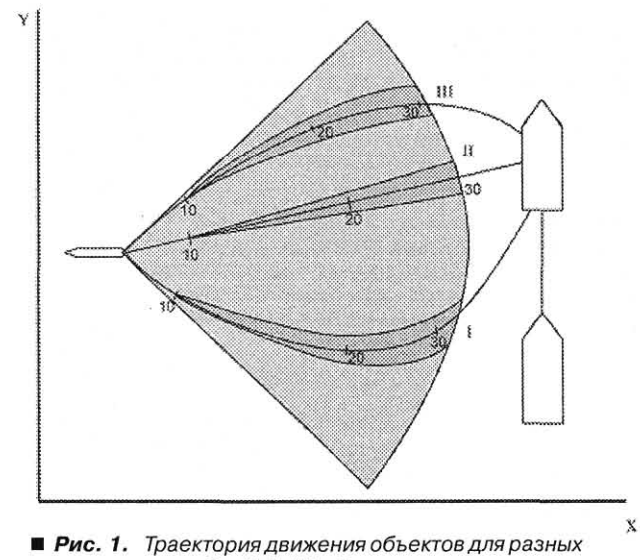
Однако и сейчас подход с частичной оптимизацией представляет значительный интерес. В теоретическом плане – это применение современных методов линейной нестационарной фильтрации. В практическом отношении – это получение вполне приемлемой точности прогноза, позволяющей сузить область ожидаемых координат местоположения объекта.

Будем придерживаться следующей последовательности изложения. Сначала сформулируем математическую модель, затем приведем алгоритмы фильтрации и прогноза траекторий и, наконец, количественно проиллюстрируем возможности фильтрации и прогноза.

Математическая модель обстановки

Системы внешнетраекторных измерений фиксируют последовательно во времени местоположение цели с ошибками. Процедура фильтрации может использоваться для уменьшения числа ошибок, а получаемые с ее помощью данные – для прогноза координат траектории на какое-то время вперед.

Сказанное иллюстрирует рис. 1, на котором приведены последовательно во времени (моменты 1, 2, ...) наступающие значения координат объекта: I – траектория в режиме погони, II – траектория при параллельном сближении, III – траектория в случае маневра уклонения, и соответствующие им прогнозируемые сектора (штриховка) вероятного местоположения объектов поражения. Широкий сектор (точки) – местоположения объекта без прогноза.



■ ***Рис. 1.*** *Траектория движения объектов для разных способов (I, II, III) сближения*

Фильтрация должна осуществляться двумерным фильтром, но, когда составляющие X_t и Y_t независимы, возможно упрощение – замена двумерного фильтра двумя одномерными на каждую из составляющих. Такого варианта будем придерживаться при последующем изложении.

Предполагается, что траектория движения цели может быть представлена многочленом X_t n-й степени

$$X_t = \sum_{k=0}^{n} \frac{\alpha k t^k}{k!}$$

с неизвестными коэффициентами $k = 0, ..., n$. В частном случае, которого мы будем придерживаться, действительно имеющую место траекторию будем представлять многочленом второй степени

$$X_t = \alpha_0 + \alpha_1 t + \frac{1}{2}\alpha t^2.$$

Коэффициенты α_0, α_1 и α_2 зависят от способа наведения (метод погони, параллельное сближение и т. д.). Так, коэффициент α_0, от которого зависит начало координат отсчета, определяется возможным сектором углов, с которых может появиться цель. Коэффициент α_1 определяет линейный снос траектории цели при параллельном ее сближении; коэффициент α_2 – квадратичный снос при наведении по режиму погони. Эти коэффициенты нам неизвестны – случайны, поскольку неизвестен режим движения цели.

Будем считать, что α_0, α_1 и α_2 независимы и нормально распределены. Допущение о независимости следует из множественности режимов поведения, для которых они различны и не связаны между собой. Нормальность распределения – обычное допущение для условий, когда известен диапазон возможных значений случайных величин и их средние значения.

Наблюдаемые (измеренные) траектории могут быть представлены дифференциалом

$$d\eta_t = X_t dt + \sqrt{C_2} dw_t^{(n)}, \qquad (1)$$

где $w_t^{(n)}$ – винеровский процесс.

Значения C_2 при моделировании назначаются тонкими, чтобы ошибки измерений и их дисперсия

$$Д_x(t) = \frac{1}{N}\sum_{j=1}^{N}\left(\eta_t^{(j)} - X_t^{(j)}\right)^2$$

соответствовали возможностям систем внешнетраекторных измерений.

Фильтрация

Для того чтобы оценить α_0, α_1 и α_2, необходимые для прогноза (непосредственно их оценить нельзя), надо рассматривать X_t как многомерный процесс, представляемый тремя производными – $\Theta_0(t)$, $\Theta_1(t)$ и $\Theta_2(t)$. Тогда оценка коэффициентов α_0, α_1 и α_2 будет эквивалентна оцениванию X_t и производных $X_t^{(1)}$ и $X_t^{(2)}$.

Введем обозначения $\Theta_t^{(k)} = X_t^{(k)}$. Тогда (1) эквивалентно системе уравнений

$$d\Theta_0(t) = \Theta_1(t)dt;$$
$$d\Theta_1(t) = \Theta_2(t)dt;$$
$$d\Theta_2(t) = 0;$$
$$d\eta_t = \Theta_0(t)dt + \sqrt{C_2}dw_t^{(n)}.$$

Как уже отмечалось, вектор $(\alpha_0, ..., \alpha_n) = (X_0, X_0^{(1)}, X_0^{(2)})$ предполагается гауссовским со средним m_0 и ковариационной матрицей Γ_t.

Процедуру фильтрации определяет линейный нестационарный фильтр Калмана–Бьюси. Его структура следует из теоремы 10.3 [3]. Согласно этой теореме, многоканальный фильтр определяется уравнениями

$$dm_t = (a_0 + a_1 m_t)dt + \left[(b\circ B) + \Gamma_t A_1^*\right](B\circ B)^{-1}\times$$
$$\times(d\eta_t - A_0 dt - A_1 m_t dt);$$
$$\Gamma_t = a_1\Gamma_t + \Gamma_t a_1^* - \left[(b\circ B) + \Gamma_t A_1^*\right](B\circ B)^{-1}\left[(b\circ B) + \Gamma_t A_1^*\right] + b\circ B;$$
$$m_0 = M(\Theta_0 \mid \eta_0);$$
$$\Gamma_0 = \|\gamma_{ij}(0)\|;$$
$$\Gamma_{ij}(0) = M\left[(\Theta_i(0) - m_i(0))(\Theta_j(0) - m_j(0))^*\right].$$

Применительно к рассматриваемому случаю, когда

$$a_0 = [0\ 0\ 0],\ a_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix},\ A_0 = [0\ 0\ 0],\ A_1 = [1\ 0\ 0],$$

$$[b\circ b] = 0, [B\circ B] = 0, [B\circ B] = C_2, \Gamma_t A_1^*(B\circ B)^{-1} = \frac{1}{C_2}\begin{bmatrix}\gamma_{00}(t)\\ \gamma_{11}(t)\\ \gamma_{22}(t)\end{bmatrix},$$

уравнения фильтрации имеют вид

$$dm_0(t) = m_1(t)dt + \frac{1}{C_2}\gamma_{00}(t)\left[d\eta_t - m_0(t)dt\right];$$
$$dm_1(t) = m_2(t)dt + \frac{1}{C_2}\gamma_{10}(t)\left[d\eta_t - m_0(t)dt\right];$$
$$dm_2(t) = \frac{1}{C_2}\gamma_{20}(t)\left[d\eta_t - m_0(t)dt\right]$$

и

$$\dot\gamma_{00}(t) = 2\gamma_{10}(t) - \frac{1}{C_2}\gamma_{00}^2(t);$$
$$\dot\gamma_{01}(t) = \gamma_{11}(t) + \gamma_{02}(t) - \frac{1}{C_2}\gamma_{22}(t)\gamma_{01}(t);$$
$$\dot\gamma_{02}(t) = \gamma_{12}(t) - \frac{1}{C_2}\gamma_{00}(t)\gamma_{02}(t);$$
$$\dot\gamma_{11}(t) = 2\gamma_{12}(t) - \frac{1}{C_2}\gamma_{01}^2(t);$$
$$\dot\gamma_{12}(t) = \gamma_{22}(t) - \frac{1}{C_2}\gamma_{01}(t)\gamma_{02}(t);$$
$$\dot\gamma_{22}(t) = -\frac{1}{C_2}\gamma_{02}^2(t)$$

с начальными условиями

$$m_0(0)=M\alpha_0,\quad m_1(0)=M\alpha_1,\quad m_2(0)=M\alpha_2;$$
$$\gamma_{01}(0)=Д\alpha_0,\quad \gamma_{01}(0)=0,\quad \gamma_{02}(0)=0;$$
$$\gamma_{11}(0)=Д\alpha_1,\quad \gamma_{12}(0)=0,\quad \gamma_{22}(0)=Д\alpha_2.$$

Ошибки фильтрации уменьшаются по мере увеличения времени наблюдения. Происходит это потому, что подлежащие фильтрации производные $\Theta_0(t)$, $\Theta_1(t)$ и $\Theta_2(t)$ являются детерминированными во времени функциями. Это доказывает и стремление к нулю значений коэффициентов $\gamma_{ij}(t), i=0,1,2; j=0,1,2$. Такая особенность фильтра важна сама по себе. И не только. По мере увеличения времени предварительной фильтрации на интервале [0, t^*] увеличивается точность прогноза координат траектории в моменты $t \geq t_k$.

Прогноз

При отсутствии помех коэффициенты α_0, α_1 и α_2 отвечали бы соотношениям

$$\alpha_0(t)=\Theta_0(t)-\Theta_1(t)t+\frac{1}{2}\Theta_2(t)t^2;$$
$$\alpha_1(t)=\Theta_1(t)-\Theta_2(t)t;$$
$$\alpha_2(t)=\Theta_2(t).$$

В результате фильтрации будут вычисляться оценки производных

$$m_0(t)=M\left[\Theta_0(t)|\,\eta_0^t\right], m_1(t)=M\left[\Theta_1(t)|\,\eta_0^t\right],$$
$$m_2(t)=M\left[\Theta_2(t)|\,\eta_0^t\right]$$

и соответствующие оценки коэффициентов $\bar{\alpha}_0(t)$, $\bar{\alpha}_1(t)$ и $\bar{\alpha}_2(t)$:

$$\bar{\alpha}_0(t)=m_0(t)-m_1(t)t+\frac{1}{2}m_2(t)t^2;$$
$$\bar{\alpha}_1(t)=m_1(t)-m_2(t)t;$$
$$\bar{\alpha}_2(t)=m_2(t).$$

По мере увеличения времени наблюдения оценки $\bar{\alpha}_0(t)$, $\bar{\alpha}_1(t)$ и $\bar{\alpha}_2(t)$ будут стремиться к α_0, α_1 и α_2, следовательно, спустя определенное время t^*, может быть рассчитан прогноз, т. е. значения координат по формуле

$$\bar{X}_t=\bar{\alpha}_0(t^*)+\bar{\alpha}_1(t^*)t+\frac{1}{2}\bar{\alpha}_2(t^*)t^2.$$

Ошибки прогноза характеризуются их дисперсией

$$Д_П(t)=\frac{1}{N}\sum_{j=1}^{N}\left(\bar{X}_t^{(j)}-X_t^{(j)}\right)^2,$$

где j – номер реализации в статистическом эксперименте.

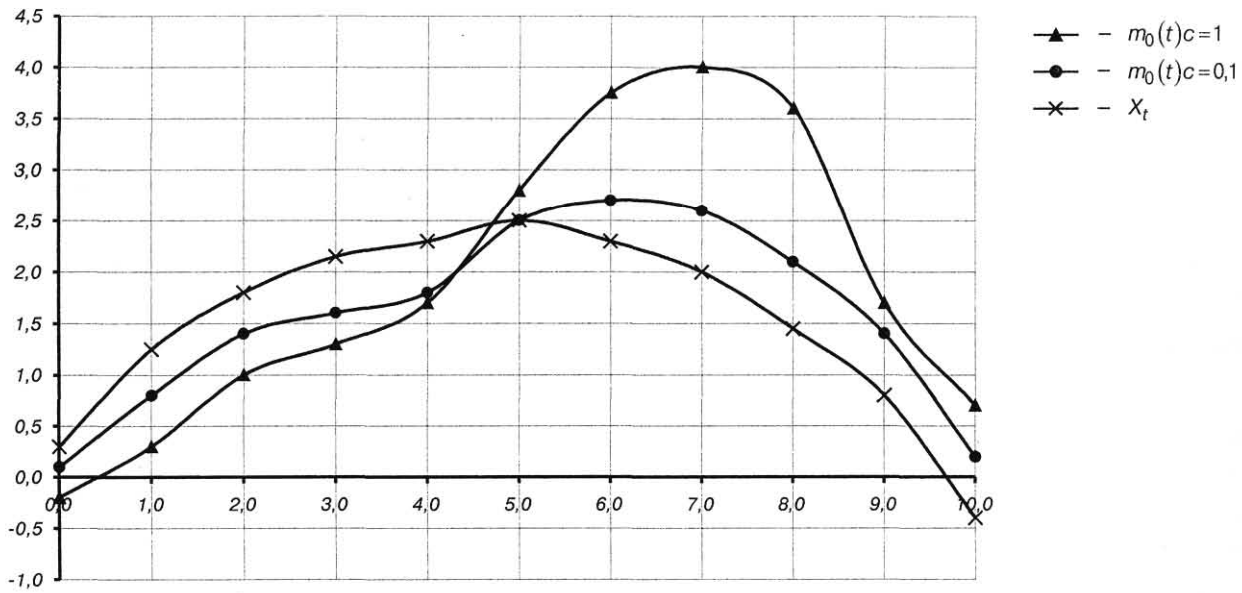
Аналогичным образом следует поступать и с наблюдаемой (измеренной) составляющей Y_t.

Будучи пересчитанными в координаты X, Y, значения $\bar{X}_t$ и $\bar{Y}_t$ определяют средние значения прогноза для моментов времени $t \geq t^*$, а их дисперсия – ширину области ожидаемого местоположения движущегося объекта.

Отметим основные отличия, являющиеся одновременно преимуществами рассмотренных процедур фильтрации и прогноза траекторий.

Процедура фильтрации учитывает распределение коэффициентов многочлена, представляющего траекторию. Благодаря этому оказывается возможным согласование процедуры фильтрации со складывающейся обстановкой.

Данная процедура нестационарна: фильтр все время находится в переходном режиме. Фильтрация оказывается похожей на оценивание параметров сигна-



■ ***Рис. 2.*** *Траектории, получаемые в результате фильтрации*

ла, где ошибки оценивания также уменьшаются по мере увеличения времени наблюдения.

Процедура фильтрации выступает как предварительная операция, подготавливающая данные для прогноза координат траектории после окончания фильтрации. При этом ошибки прогноза оказываются тем меньше, чем больше времени затрачено на предварительную фильтрацию. В зависимости от резерва времени становится возможным рассчитать точность прогнозируемых координат траектории.

Моделирование

Для определения эффективности алгоритмов необходимо статистическое моделирование на ЭВМ. Пока это единственный способ получения количественных характеристик, а также обоснования требований к техническим параметрам цифровых устройств.

Задание производных $\Theta_0(t)$, $\Theta_1(t)$ и $\Theta_2(t)$, а также вычисление оценок осуществлялось в результате решения их рекуррентных соотношений. Приращения наблюдаемых воздействий определялись формулой

$$\eta_{k+1} - \eta_{t_k} = \Theta_0(t_k)\Delta + \sqrt{C_2\Delta}\,X_{t_{k+1}}^{(n)}.$$

Значения коэффициентов α_o, α_1 и α_2 разыгрывались по нормальному закону.

Моделирование проводилось при $\alpha_0 \approx N$ (0,100 м), $\alpha_1 \approx N[0, (5\ \text{м/с})^2]$ и $\alpha_2 \approx N[0, (0{,}2\ \text{м/с}^2)^2]$. Помеха – белый шум с $C_2 = 0{,}1\ \text{с}^{-1}$ и $C_2 = 1{,}0\ \text{с}^{-1}$.

Результаты моделирования иллюстрируются рис. 2, на котором для частного случая, когда $\alpha_0 = -2\ \text{м/с}^2$, приведены X_t и отфильтрованные ее значения при $C_2 = 0{,}1$ и $C_2 = 1{,}0$.

Зависимость дисперсии ошибок фильтрации от времени наблюдения, т. е.

$$Д_{\text{оси}}^{(\text{ф})}(t_k) = \frac{1}{N}\sum_{j=1}^{N}\left[m_0^{(j)}(t_k) - \Theta_0^{(j)}(t_k)\right]^2;$$

$$\Theta_0^{(j)}(t_k) = X_{t_k}^{(j)},$$

где j – номер реализации, имела вид

t, с	2	4	6	8	10
$C_2 = 0{,}1\ \text{с}^{-1}$	0,180	0,120	0,123	0,109	0,094
$C_2 = 1\ \text{с}^{-1}$	1,77	0,605	0,892	0,908	0,871

Зависимость дисперсии ошибок прогноза для моментов времени $t \geq t_0$

$$Д_{\text{оси}}^{(\text{п})}(t_k) = \frac{1}{N}\sum_{j=1}^{N}\left(\bar{X}_{t_k}^{(j)} - X_{t_k}^{(j)}\right)^2$$

при $t^* = 10$ с имела вид

t, с	12	14	16	18	20
$C_2 = 0{,}1\ \text{с}^{-1}$	0,382	1,08	2,48	4,92	8,79
$C_2 = 1{,}0\ \text{с}^{-1}$	3,42	9,62	21,9	43,1	76,9

С увеличением времени фильтрации до $t^* = 20$ с дисперсии ошибок прогноза уменьшаются до следующих значений:

t, с	22	24	26	28	30
$C_2 = 0{,}1\ \text{с}^{-1}$	0,098	0,189	0,332	0,544	0,845

Таким образом, моделирование количественно иллюстрирует уменьшение ошибок фильтрации по мере увеличения времени наблюдения. Ошибки могут уменьшаться до сколь угодно малой величины. Свидетельство тому – стремление к нулю значений дисперсионных коэффициентов $\gamma_{ij}(t)$, $i = 0, \ldots, n$; $j = 0, 1, \ldots, n$. По мере увеличения времени предварительной фильтрации на интервале $[0, t]$ увеличивается точность прогноза координат траектории в моменты $t \geq t^*$.

Важно иметь в виду и то обстоятельство, что в отличие от режима фильтрации, где ошибка фильтрации уменьшалась по мере увеличения времени наблюдения, в режиме прогноза ошибки прогноза, напротив, растут по мере удаления от момента t^*. Это объясняется решающей ролью временного фактора: остающееся различие между α_0, α_1 и α_2 и $\bar{\alpha}_0$, $\bar{\alpha}_1$, $\bar{\alpha}_2$ приводит по мере увеличения времени к накоплению разности в координатах.

Таким образом, теоретически обоснована и подтверждена методом моделирования перспективность использования нестационарной линейной фильтрации траектории движения объектов для составления алгоритмов, обеспечивающих минимизацию ошибок прогноза в решении задач противоракетной и противоторпедной обороны.

Литература

1. **Калман Р. Е., Бьюси Р. С.** Новые результаты в линейной фильтрации и теория предсказания/Пер. с англ. – Техническая механика – 1961. – № 83. – Сер. Д. 1.
2. **Кузьмин С.З.** Основы проектирования систем цифровой обработки радиолокационной информации. – М.: Радио и связь, 1986.
3. **Липцер Р. Ш., Ширяев А.Н.** Статистика случайных процессов. – М.: Наука, 1974.

УДК 621(075.8)

ОЦЕНКА ПРОФЕССИОНАЛЬНОЙ ПРИГОДНОСТИ ОПЕРАТОРОВ ЧЕЛОВЕКО–МАШИННЫХ СИСТЕМ ПО РЕЗУЛЬТАТАМ РЕШЕНИЯ ТЕСТОВЫХ ЗАДАЧ

Dr. Al-Kasasbeh Riad Taha,
al-balqa applied university
А. Е. Городецкий,
д-р техн. наук., профессор
И. Л. Тарасова,
канд. техн. наук
Институт проблем машиноведения РАН, Санкт-Петербург

Анализируются методы отбора претендентов на работу в качестве операторов человеко-машинных систем. Предлагаются векторные оценки интеллекта претендентов, содержащие статические и динамические компоненты, характеризующие способности претендентов к решению задач в условиях неопределенности и к самообучению. Исследуются свойства предлагаемых оценок на примере тестирования пяти претендентов.

Methods of selection of applicants for work are analyzed as operators of human-machine systems. Vector estimations of intelligence of the applicants, the containing static and dynamic components, describing abilities of applicants to the decision of problems in conditions of uncertainty and to self-training are offered. Properties of offered estimations are investigated by the example of testing five applicants.

Введение

В процессе работы оператора в человеко-машинной системе (ЧМС) последнему нередко приходится принимать решения в условиях не полной определенности. При этом в зависимости от интеллектуальных особенностей оператора в процессе его умозаключения, т. е. анализа ситуации выбора, вероятность правильности выбора или уверенность в правильности выбора решения может меняться у оператора с течением времени по тому или иному закону. Это, как показано в работе [1], может приводить к сбоям в работе ЧМС, а иногда и к катастрофическим последствиям.

Для устранения подобных сбоев в ЧМС или хотя бы их максимального уменьшения используются, как правило, некоторые процедуры отбора наиболее подходящих (лучших) для данной ЧМС операторов из имеющихся претендентов. Обычно подобные процедуры основаны на использовании тестирования претендентов и оценки их интеллектуальных способностей к выполнению необходимой в ЧМС работы по тем или иным шкалам.

Известен [2] подход к оценке интеллекта, основанный на вероятностных оценках степени знания и понимания. Также известен [3] подход, в котором мера интеллекта связывается со степенью понимания и включает аспекты осознания и самоосознания. Причем обычно, в том числе и в работах [2, 3], интеллект характеризуется только способностью к обучению, хотя и признается существование разных видов интеллекта [4].

В работах [5–7] предлагается системы искусственного интеллекта рассматривать как целеустремленные системы, описываемые в работе [2]. Однако для оценки интеллектуальности таких систем предлагается [5–7] использовать вектор, компоненты которого характеризуют не только способность к самообучению, но и их приспособленность к получению ценных (правильных) результатов (решений) путем применения эффективных способов действий в нечеткой среде.

Предложенная в [5] векторная оценка интеллектуальности системы искусственного интеллекта имеет статические вероятностные компоненты, оценивающие способность к решению каких-либо нечетких прикладных задач, и динамические вероятностные компоненты, оценивающие способность системы к самообучению. Очевидно, что такой подход можно использовать и для объективной оценки профессиональной пригодности оператора к работе в той или иной человеко-машинной системе. При этом сравнение и выбор лучших из группы тестируемых претендентов может опираться на численные оценки результатов динамического тестирования, когда результаты тестирования фиксируются через различные промежутки времени, на которые разбивается весь сеанс тестирования.

Однако в силу специфических отличий человека от систем искусственного интеллекта, рекомендуемые формулы для вычисления компонент предлагаемой векторной оценки интеллектуальности [5–7] не могут непосредственно использоваться для оценки профессиональной пригодности операторов по результатам динамического тестирования.

Эффективность, достоверность и правдоподобность

Введенные в работах [5–7] векторные оценки интеллектуальности опираются на понятия эффективности, достоверности и правдоподобности, заимствованные из работы [2]. Однако предлагаемые в [5–7] формулы для их вычисления не подходят для оценки человека.

При тестировании операторов им обычно предлагается к решению ряд задач типа:

Если x_1 и x_2 и ... и x_n, то y_1 или y_2 или ... или y_m.

Оператор должен выбрать один или несколько ответов y_i, которые, по его мнению, являются правильными. Результаты тестирования при этом оцениваются по балльной системе. При динамическом тестировании результаты тестирования фиксируются периодически, через некоторые промежутки времени $K(t - t_0)$, t_0 – время начала тестирования, t – установленное контрольное время. В табл. 1 – 5 приведены результаты тестирования пяти претендентов. Каждому претенденту задавалось двадцать (n = = 20) задач и четыре (m = 4) варианта ответов. Фикса-

■ ***Таблица 1.*** *Результаты тестирования оператора № 1*

K	m/n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1																				
	2																				
	3																				
	4																				
2	1			1																	
	2	1	1			1	1	1			1	1	1	1			1	1		1	1
	3		1	1	1	1			1	1	1				1	1	1	1	1		
	4	1			1		1	1	1	1		1	1	1	1	1			1	1	1
3	1			1						1											
	2	1			1		1	1	1	1	1	1	1	1	1	1	1		1		
	3		1	1	1	1			1							1		1	1	1	
	4	1	1			1	1	1			1	1	1	1	1		1	1			1
4	1			1																	
	2				1		1	1	1	1	1		1	1					1		
	3	1	1		1	1					1	1	1		1		1		1		
	4	1	1	1		1	1	1	1	1		1		1	1	1		1		1	1
5	1																				
	2				1																
	3					1	1	1	1		1		1	1			1		1		
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1																				
	2																				
	3				1	1															
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1																				
	2																				
	3					1		1				1									
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1																				
	2																				
	3					1		1			1										
	4	1	1	1	1		1	1	1	1		1	1	1	1	1	1	1	1	1	1
9	1																				
	2																				
	3										1						1			1	
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1
10	1																				
	2																				
	3									1											
	4	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1

■ ***Таблица 2.*** *Результаты тестирования оператора № 2*

K	m/n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1					1	1								1						1
	2	1	1	1	1		1	1	1	1	1	1	1	1			1	1	1	1	1
	3	1	1	1			1	1	1	1	1		1	1	1	1	1	1	1	1	
	4				1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1
3	1		1	1																1	1
	2	1	1	1	1	1	1	1	1	1		1	1		1	1	1				
	3	1	1	1		1		1	1	1		1	1	1	1	1	1		1	1	
	4	1			1	1	1		1	1	1	1	1	1	1	1	1	1		1	1
4	1																				
	2		1	1		1		1	1		1	1	1	1	1					1	
	3	1	1	1	1	1	1			1			1	1	1	1		1			1
	4	1			1		1	1	1	1	1	1					1		1	1	1
5	1																				
	2																				
	3		1	1		1		1			1	1	1	1	1	1		1		1	
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1																				
	2																				
	3																	1		1	
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1																				
	2					1	1	1		1					1						
	3	1	1	1	1				1	1	1		1	1	1	1	1	1	1		
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1																				
	2	1	1		1		1														
	3	1	1	1	1	1	1	1			1	1	1	1	1						1
	4	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1																				
	2		1	1	1								1	1							
	3	1	1	1	1	1	1	1	1					1							
	4	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	1			1																	
	2							1													
	3	1	1	1				1		1			1	1							1
	4	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1

■ ***Таблица 3.*** *Результаты тестирования оператора № 3*

K	m/n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1			1	1		1		1		1					1	1	1			
	2		1	1		1		1	1			1		1	1	1			1	1	
	3	1	1		1			1		1			1				1		1		1
	4	1				1	1			1	1	1	1	1	1			1		1	1
2	1								1												
	2	1	1		1	1	1	1			1	1	1	1	1		1			1	
	3			1		1	1	1	1	1				1		1	1	1	1		1
	4	1	1	1	1					1	1	1	1		1	1		1	1	1	1
3	1	1		1																	
	2		1	1	1			1	1	1	1		1								
	3	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1		1	1	
	4	1	1		1	1	1		1		1	1	1	1	1	1	1	1	1	1	1
4	1																				
	2				1						1		1				1			1	
	3	1		1	1		1			1	1	1	1		1			1			
	4	1	1	1		1	1	1	1			1		1	1	1	1	1	1		1
5	1																				
	2																				
	3				1		1			1	1	1	1				1			1	1
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Окончание таблицы 3

K	m/n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
6	1																				
	2																				
	3						1										1	1		1	1
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1																				
	2																				
	3							1	1					1			1				
	4	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1
8	1																				
	2																				
	3					1		1					1								
	4	1	1	1	1		1	1	1	1	1	1		1	1	1	1	1	1	1	1
9	1																				
	2																				
	3							1					1								
	4	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1
10	1																				
	2																				
	3						1														
	4	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1

■ ***Таблица 4.*** *Результаты тестирования оператора № 4*

K	m/n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1							1	1				1			1			1		
	2			1	1	1	1				1	1		1	1	1	1	1		1	1
	3		1	1		1	1	1			1	1	1	1	1	1	1	1	1	1	
	4	1	1	1		1	1	1	1		1		1	1	1	1	1	1	1	1	1
2	1																				
	2				1			1		1			1							1	
	3		1	1			1	1	1	1	1	1		1	1	1	1	1	1	1	1
	4	1	1	1	1	1	1		1		1		1	1	1	1	1	1	1	1	1
3	1																				
	2				1			1		1			1							1	
	3		1	1			1	1	1	1	1	1		1	1	1	1	1	1	1	1
	4	1	1	1	1	1	1		1		1		1	1	1	1	1	1	1		1
4	1																				
	2							1	1	1								1			
	3		1		1	1			1			1	1				1	1	1	1	1
	4	1	1	1	1	1	1	1		1	1	1	1	1	1	1			1	1	1
5	1																				
	2																				
	3							1	1								1	1			
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1																				
	2																				
	3							1	1								1	1			
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1																				
	2																				
	3									1								1			
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1																				
	2																				
	3							1						1		1					
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1
9	1																				
	2																				
	3							1												1	
	4	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1
10	1																				
	2																				
	3					1						1									
	4	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1

■ ***Таблица 5.*** *Результаты тестирования оператора № 5*

K	m/n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1																				
	2	1			1					1	1			1				1		1	1
	3		1	1				1	1				1			1			1		
	4					1	1					1			1		1				
2	1																				
	2							1	1					1	1			1			
	3	1		1	1						1		1						1	1	
	4		1			1	1			1		1					1	1			1
3	1																				
	2	1								1				1	1						
	3		1		1	1		1			1		1				1	1		1	1
	4			1			1		1			1							1	1	
4	1																			1	
	2																				
	3	1	1		1			1		1	1		1		1		1				
	4			1		1	1		1			1		1		1		1	1	1	1
5	1																				
	2																				
	3	1		1		1		1					1		1		1				
	4	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1																				
	2																				
	3	1				1						1			1						
	4		1	1		1	1	1	1	1	1		1	1	1	1	1	1	1	1	1
7	1																				
	2																				
	3	1		1							1		1								
	4	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1																				
	2																				
	3				1											1					
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1																				
	2																				
	3																				
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	1													.							
	2																				
	3																				
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

ция результатов проводилась через равные промежутки времени десять ($K = 10$) раз. Для удобства анализа таблиц правильные ответы располагались в таблицах четвертыми.

Эффективность решения предложенных при тестировании задач будем вычислять следующим образом:

$$E_i(K) = n_{ia}(K) / (n_{ia}+m), \qquad (1)$$

где $n_{ia}(K)$ – число альтернативных ответов, полученных i-м претендентом к моменту времени $K(t - t_0)$ при решении всех задач.

Достоверность полученных результатов оценим так:

$$V_i(K) = m_{iv}(K) / n_{ia}, \qquad (2)$$

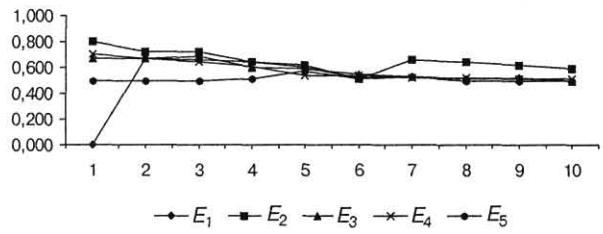
где $m_{iv}(K)$ – число задач, в которых i-м претендентом получен один правильный ответ и не более одного неправильного к моменту времени $K(t - t_0)$.

Правдоподобность полученных результатов оценим так:

$$U_i(K) = m_{iu}(K) / n_{ia}, \qquad (3)$$

■ ***Таблица 6.*** *Эффективность*

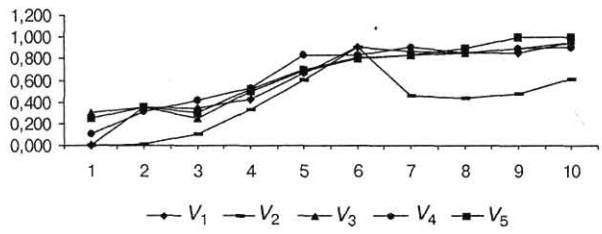
K	1	2	3	4	5	6	7	8	9	10
E_1	0,000	0,667	0,665	0,640	0,600	0,520	0,530	0,510	0,520	0,500
E_2	0,800	0,722	0,718	0,640	0,620	0,510	0,660	0,640	0,620	0,590
E_3	0,667	0,667	0,687	0,600	0,590	0,550	0,530	0,510	0,510	0,500
E_4	0,706	0,667	0,643	0,610	0,540	0,540	0,520	0,520	0,510	0,510
E_5	0,500	0,500	0,500	0,510	0,570	0,510	0,530	0,500	0,500	0,500



■ ***Рис. 1.*** *Зависимость эффективности от интервала времени принятия решения*

■ ***Таблица 7.*** *Достоверность*

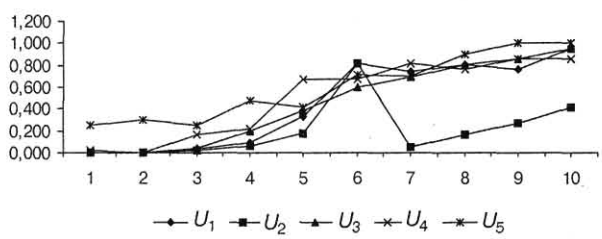
K	1	2	3	4	5	6	7	8	9	10
V_1	0,000	0,350	0,342	0,430	0,670	0,910	0,870	0,860	0,860	0,950
V_2	0,000	0,006	0,100	0,330	0,610	0,910	0,460	0,440	0,480	0,620
V_3	0,300	0,350	0,250	0,500	0,690	0,800	0,830	0,860	0,900	0,950
V_4	0,100	0,310	0,420	0,530	0,830	0,830	0,910	0,860	0,900	0,910
V_5	0,250	0,350	0,300	0,520	0,700	0,810	0,830	0,900	1,000	1,000



■ ***Рис. 2.*** *Зависимость достоверности от интервала времени принятия решений*

■ ***Таблица 8.*** *Правдоподобность*

K	1	2	3	4	5	6	7	8	9	10
U_1	0,000	0,000	0,030	0,090	0,330	0,820	0,740	0,810	0,770	0,950
U_2	0,000	0,000	0,020	0,060	0,180	0,820	0,050	0,170	0,270	0,410
U_3	0,000	0,000	0,040	0,200	0,380	0,600	0,690	0,810	0,860	0,950
U_4	0,020	0,002	0,170	0,220	0,670	0,670	0,820	0,770	0,860	0,860
U_5	0,250	0,300	0,250	0,480	0,410	0,710	0,700	0,900	1,000	1,000



■ ***Рис. 3.*** *Зависимость правдоподобности от интервала времени принятия решения*

где $m_{iu}(K)$ – число задач, по которым i-м претендентом получен только один, причем правильный, ответ к моменту времени $K(t - t_0)$.

Вычисленные значения эффективности, достоверности и правдоподобности ответов пяти тестируемых претендентов приведены в табл. 6–8, а изменения их во времени показаны на рис. 1–3.

Анализ зависимостей, приведенных на рис. 1–3, не позволяет непосредственно сравнить профессиональную пригодность тестируемых операторов. Однако бросается в глаза неудачность ответов оператора № 2 на последней стадии тестирования и очень малая эффективность у оператора № 1 в начале тестирования, что может говорить об их психологических особенностях.

Таким образом конструировать (вычислять) компоненты вектора интеллекта, по которым можно будет сравнивать тестируемых претендентов, предлагается другим способом.

Статические компоненты вектора интеллекта

Статическими компонентами вектора интеллекта могут быть *степень осведомленности* тестируемого о данной сфере деятельности, *степень знания* о методах решения задач в данной предметной области, *степень понимания* выбора правильных решений из множества альтернативных и *степень мотивируемости* при решении поставленных задач.

Вводимые компоненты являются аналогами соответствующих компонент, исполь зуемых в работах [5–7] для оценки систем искусственного интеллекта.

По результатам тестирования (см. табл. 1–5) можно вычислить оценку *неосведомленности* претендента (табл. 9, рис. 4):

$$O_i(K) = (n_{ia}(K) + m(n - 1))/mn \qquad (4)$$

и *степень осведомленности* (табл. 10, рис. 5):

$$I_{io} = 1/K \sum_{j=1}^{K} (1 - (O_i(j) - O_{i\min})/(O_{i\max} - O_{i\min})), \qquad (5)$$

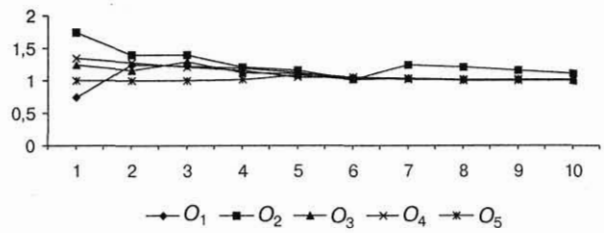
где $O_{i\max}$ и $O_{i\min}$ – максимальные и минимальные значения неосведомленности, приведенные в табл. 9.

Анализ зависимостей, приведенных на рис. 4, показывает, что изменения неосведомленности во времени различны у претендентов, что может характеризовать их обучаемость и поэтому в дальнейшем эта информация будет использована при конструировании динамических компонент вектора интеллекта.

Из диаграммы на рис. 5 видно, что степени осведомленности у претендентов заметно отличаются и поэтому введенная оценка, в принципе, характеризует их интеллектуальность и может быть использована как одна из статических компонент вектора интеллекта.

■ ***Таблица 9.*** *Неосведомленность*

K	1	2	3	4	5	6	7	8	9	10
O_{1k}	0,75	1,25	1,22	1,19	1,12	1,02	1,03	1,01	1,02	1,00
O_{2k}	1,75	1,40	1,39	1,20	1,16	1,02	1,24	1,20	1,16	1,11
O_{3k}	1,25	1,25	1,30	1,12	1,11	1,06	1,03	1,01	1,01	1,00
O_{4k}	1,35	1,27	1,20	1,15	1,05	1,05	1,02	1,02	1,01	1,01
O_{5k}	1,00	1,00	1,00	1,01	1,09	1,01	1,03	1,00	1,00	1,00



■ ***Рис. 4.*** *Зависимость неосведомленности претендента от интервала времени принятия решения*

Аналогично по результатам тестирования (см. табл. 1–5) можно вычислить оценку *знания* претендента (табл.11, рис.6):

$$G_i(K) = (n_{iv}(K) + m(n-1))/mn. \qquad (6)$$

Анализ зависимостей, приведенных на рис. 6, показывает, что знания изменяются в процессе раздумий несколько различно у претендентов, что может характеризовать их обучаемость и поэтому в дальнейшем эта информация будет использована при конструировании динамических компонент вектора интеллекта.

Степень знания (табл. 12, рис. 7) определим следующим образом:

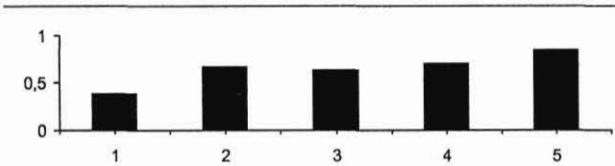
$$I_{Gi} = 1/K \sum_{j=1}^{K} (G_i(j) - G_{i\min})/(G_{i\max} - G_{i\min}), \qquad (7)$$

где $G_{i\max}$ и $G_{i\min}$ – максимальные и минимальные значения знания в табл. 11

Из диаграмм, приведенных на рис. 7, видно, что степени знаний у претендентов заметно отличаются и поэтому введенная оценка, в принципе, характеризует их интеллектуальность и может быть использована как одна из статических компонент вектора интеллекта.

■ ***Таблица 10.*** *Степень осведомленности*

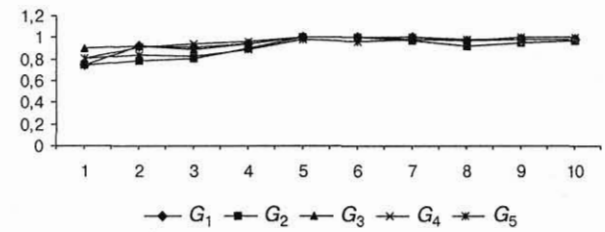
i	1	2	3	4	5
I_{oi}	0,376	0,671	0,633	0,697	0,844



■ ***Рис. 5.*** *Диаграммы степеней осведомленности I_{oi} тестируемых претендентов*

■ ***Таблица 11.*** *Знания*

K	1	2	3	4	5	6	7	8	9	10
G_{1k}	0,75	0,92	0,91	0,94	1,00	1,00	1,00	0,97	0,99	0,99
G_{2k}	0,75	0,79	0,81	0,90	1,00	1,00	0,97	0,95	0,95	0,97
G_{3k}	0,90	0,92	0,89	0,94	1,00	1,00	0,99	0,97	0,99	0,99
G_{4k}	0,81	0,91	0,94	0,96	1,00	1,00	1,00	0,99	0,99	0,99
G_{5k}	0,81	0,84	0,82	0,89	0,99	0,96	0,99	0,97	1,00	1,00



■ ***Рис. 6.*** *Зависимость знания от интервала времени принятия решения*

Аналогично можно вычислить оценку *понимания* (табл. 13, рис. 8):

$$H_i(K) = (n_{ui}(K) + m(n-1))/mn. \qquad (8)$$

Анализ зависимостей, приведенных на рис. 8, показывает, что понимания при решении задач у претендентов изменяются в процессе раздумий несколько различным образом, что может характеризовать их обучаемость, и поэтому в дальнейшем эта информация будет использована при конструировании динамических компонент вектора интеллекта.

Степень понимания (табл. 14, рис. 9) определим так

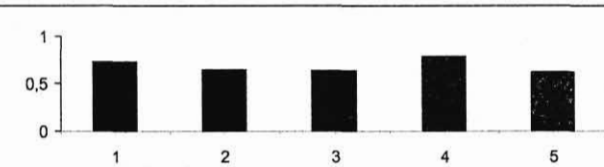
$$I_{Hi} = 1/K \sum_{j=1}^{K} (H_i(j) - H_{i\min})/(H_{i\max} - H_{i\min}), \qquad (9)$$

где $H_{i\max}$ и $H_{i\min}$ – максимальные и минимальные значения понимания (см. табл. 13).

Из диаграмм, приведенных на рис. 9, видно, что степени понимания у претендентов заметно отличаются и поэтому введенная оценка, в принципе, характеризует их интеллектуальность и может быть использована как одна из статических компонент вектора интеллекта.

■ ***Таблица 12.*** *Степень знаний*

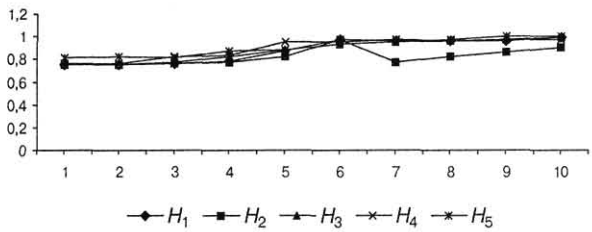
i	1	2	3	4	5
I_{Gi}	0,723	0,64	0,633	0,78	0,617



■ ***Рис. 7.*** *Диаграмма степеней знания I_{Gi} тестируемых претендентов*

■ ***Таблица 13.*** *Понимания*

K	1	2	3	4	5	6	7	8	9	10
H_{1k}	0,75	0,75	0,76	0,79	0,87	0,97	0,96	0,96	0,96	0,99
H_{2K}	0,75	0,75	0,76	0,77	0,82	0,97	0,77	0,82	0,96	0,90
H_{3k}	0,75	0,75	0,77	0,82	0,89	0,94	0,95	0,96	0,97	0,99
H_{4k}	0,76	0,76	0,82	0,84	0,95	0,95	0,97	0,96	0,97	0,97
H_{5k}	0,81	0,82	0,81	0,87	0,89	0,94	0,95	0,97	1,00	1,00



■ ***Рис. 8.*** *Зависимость понимания от интервала времени принятия решения*

Используя вычисленные ранее значения достоверности ответов претендентов (табл. 7, рис. 2), можно вычислить их *инструктивность* (табл. 15, рис. 10):

$$L_{vi}(K) = |V_i(K) - 1/n|. \qquad (10)$$

Анализ зависимостей, приведенных на рис. 10, показывает, что инструктивность при решении задач у претендентов существенно изменяется в процессе раздумий, что может характеризовать их обучаемость, и поэтому в дальнейшем эта информация будет использована при конструировании динамических компонент вектора интеллекта.

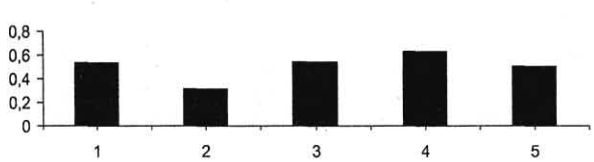
Используя вычисленные ранее значения правдоподобности ответов претендентов (см. табл. 8, рис. 3), можно вычислить их *информативность* (табл. 16, рис.11):

$$L_{ui}(K) = |U_i(K) - 1/m|. \qquad (11)$$

Анализ зависимостей, приведенных на рис. 11, показывает, что информативность при решении задач у претендентов существенно изменяется в процессе раздумий, что может характеризовать их обучаемость, и поэтому в дальнейшем эта информация будет использована при конструировании динамических компонент вектора интеллекта.

■ ***Таблица 14.*** *Степень понимания*

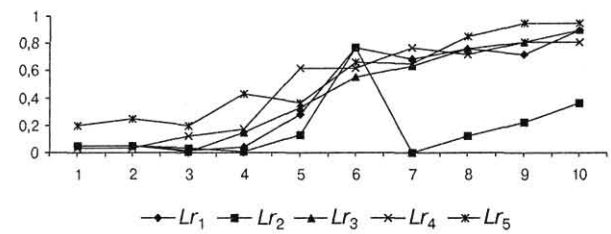
i	1	2	3	4	5
I_{Hi}	0,536	0,311	0,547	0,633	0,507



■ ***Рис. 9.*** *Диаграмма степеней понимания I_{Hi} тестируемых претендентов*

■ ***Таблица 15.*** *Инструктивность*

K	1	2	3	4	5	6	7	8	9	10
L_{v1}	0,05	0,05	0,02	0,04	0,28	0,77	0,69	0,76	0,72	0,9
L_{v2}	0,05	0,05	0,03	0,01	0,13	0,77	0	0,12	0,22	0,36
L_{v3}	0,05	0,05	0,01	0,15	0,33	0,55	0,64	0,76	0,81	0,9
L_{v4}	0,03	0,03	0,12	0,17	0,62	0,62	0,77	0,72	0,81	0,81
L_{v5}	0,2	0,25	0,2	0,43	0,36	0,66	0,65	0,85	0,95	0,95



■ ***Рис. 10.*** *Зависимость инструктивности от интервала времени принятия решения*

На основании инструктивности и информативности можно определить *мотивируемость* (табл. 17, рис. 12) претендентов при решении предлагаемых тестовых задач, вычисляемую в виде суммы:

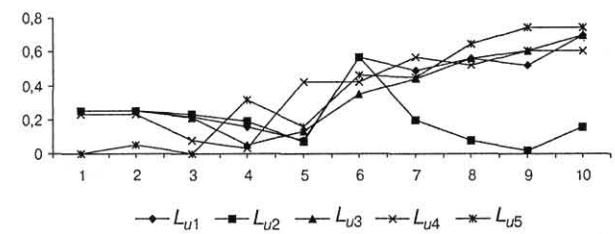
$$L_i(K) = k_v L_{vi}(K) + k_u L_{ui}(K), \qquad (12)$$

где k_v и k_u – весовые коэффициенты (в рассматриваемом примере $k_v = k_u = 1$).

Анализ зависимостей, приведенных на рис. 12, показывает, что мотивируемость при решении задач у претендентов существенно изменяется после примерно половины отведенного времени на раздумья, что может характеризовать их обучаемость, и поэтому в дальнейшем эта информация будет использована при конструировании динамических компонент вектора интеллекта. Кроме того, бросается в глаза резкий спад мотивируемости у оператора № 2 после примерно половины отведенного времени на раздумья, что может говорить о психологических особенностях этого оператора.

■ ***Таблица 16.*** *Информативность*

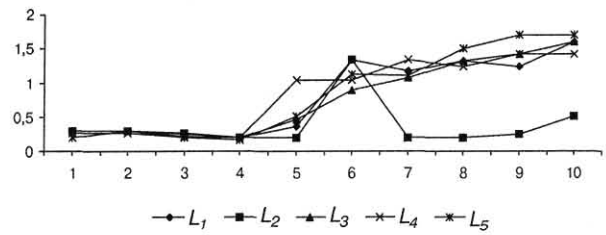
K	1	2	3	4	5	6	7	8	9	10
L_{u1}	0,25	0,25	0,22	0,16	0,08	0,57	0,49	0,56	0,52	0,7
L_{u2}	0,25	0,25	0,23	0,19	0,07	0,57	0,2	0,08	0,02	0,16
L_{u3}	0,25	0,25	0,21	0,05	0,13	0,35	0,44	0,56	0,61	0,7
L_{u4}	0,23	0,23	0,08	0,03	0,42	0,42	0,57	0,52	0,61	0,61
L_{u5}	0	0,05	0	0,32	0,16	0,46	0,45	0,65	0,75	0,75



■ ***Рис. 11.*** *Зависимость информативности от интервала времени принятия решения*

■ ***Таблица 17.*** *Мотивируемость*

K	1	2	3	4	5	6	7	8	9	10
L_1	0,3	0,3	0,24	0,2	0,36	1,34	1,18	1,32	1,24	1,6
L_2	0,3	0,3	0,26	0,2	0,2	1,34	0,2	0,2	0,24	0,52
L_3	0,3	0,3	0,22	0,2	0,46	0,9	1,08	1,32	1,42	1,6
L_4	0,26	0,26	0,2	0,2	1,04	1,04	1,34	1,24	1,42	1,42
L_5	0,2	0,3	0,2	0,16	0,52	1,12	1,1	1,5	1,7	1,7



■ ***Рис. 12.*** *Зависимость мотивируемости от интервала времени принятия решения*

Степень мотивируемости (табл.18 и рис.13) будем определять как:

$$I_L = 1/K \sum_{j=1}^{K} (L_i(j) - L_{i\min})/(L_{i\max} - L_{i\min}), \qquad (13)$$

где $L_{i\max}$ и $L_{i\min}$ – максимальные и минимальные значения мотивируемости (см. табл.17).

Из рис.13 видно, что степени мотивируемости у претендентов заметно отличаются и соответствуют результатам тестирования. Поэтому введенная оценка, в принципе, характеризует их интеллектуальность и может быть использована как одна из статических компонент вектора интеллекта.

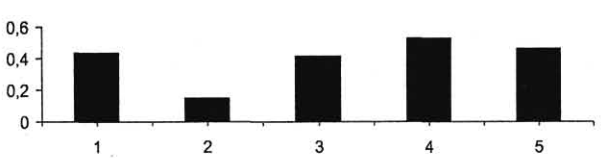
Таким образом, способность претендентов к решению поставленных тестовых задач и, соответственно, к принятию правильных решений в условиях неопределенности можно оценить по предлагаемым статическим компонентам вектора интеллекта (табл. 19, рис. 14).

Иногда можно использовать приближенную, *среднюю оценку способности претендентов*, к принятию решения в условиях неопределенности (табл. 20, рис. 15), вычисляемую следующим образом:

$$I_i^{ct} = 1/4\,(k_O I_{Oi} + k_G I_{Gi} + k_H I_{Hi} + k_L I_{Li}), \qquad (14)$$

■ ***Таблица 18.*** *Степень мотивируемости*

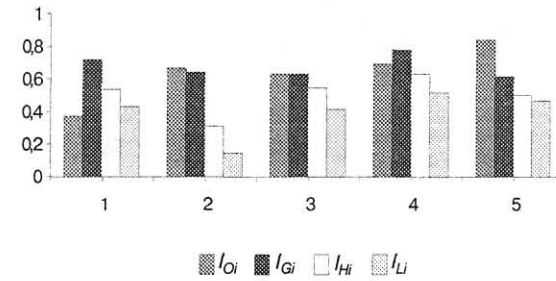
i	1	2	3	4	5
I_{Li}	0,434	0,151	0,414	0,526	0,467



■ ***Рис. 13.*** *Диаграмма степеней мотивируемости I_{Li} тестируемых претендентов*

■ ***Таблица 19.*** *Статические компоненты вектора интеллекта*

I	1	2	3	4	5
I_{Oi}	0,376	0,671	0,633	0,697	0,844
I_{Gi}	0,723	0,64	0,633	0,78	0,617
I_{Hi}	0,536	0,311	0,547	0,633	0,507
I_{Li}	0,434	0,151	0,414	0,526	0,467



■ ***Рис. 14.*** *Статические компоненты вектора интеллекта тестируемых претендентов*

где k_O, k_G, k_H и k_L – весовые коэффициенты, которые в рассматриваемом примере равны единице.

Из диаграмм, приведенных на рис. 15, видно, что претендентов можно легко расставить согласно средней оценке по порядку от лучшего к худшему: (4, 5, 3, 1, 2). Следовательно, при условиях рассматриваемого примера, *средняя статическая оценка интеллекта* вполне может быть использована для оценки способности претендентов к принятию решений в условиях неопределенности.

Динамические компоненты вектора интеллекта

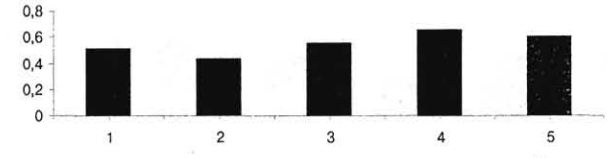
Динамические компоненты так же, как и в работах [5–7], характеризуют способность претендентов к обучению. *Обучаемость* претендентов может оцениваться *вектором обучаемости* (см. далее. табл. 25, рис. 20), компоненты которого вычисляются по результатам тестирования претендентов и характеризуются средними скоростями изменения осведомленности J_{Oi}, знания J_{Gi} и понимания J_{Hi}, вычисляемыми по формулам:

$$J_{Oi} = (\sum_K (\Delta O_i(K)) / \sum_K ((\Delta L_i(K)), \qquad (15)$$

где $\Delta L_i(K) = (L_{iK+1} - L_{iK})/(L_i^{\max} - L_i^{\min})$ – изменение мотивируемости (табл. 21, рис. 16);

■ ***Таблица 20.*** *Средняя статическая оценка интеллекта*

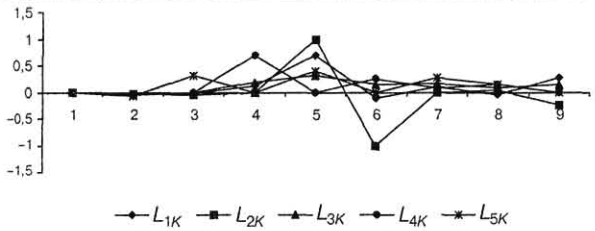
I	1	2	3	4	5
I_i^{ct}	0,517	0,443	0,557	0,659	0,609



■ ***Рис. 15.*** *Средняя статическая оценка способностей I тестируемых претендентов*

■ ***Таблица 21.*** *Изменение мотивируемости*

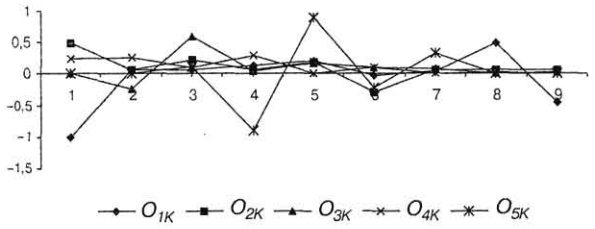
K	1	2	3	4	5	6	7	8	9
L_{1K}	0	–0,042	–0,029	0,114	0,7	–0,114	0,1	–0,057	0,257
L_{2K}	0	–0,035	–0,053	0	1	–1	0	0,035	–0,245
L_{3K}	0	–0,057	–0,014	0,186	0,314	0,13	0,17	0,071	0,129
L_{4K}	0	–0,049	0	0,688	0	0,246	0,082	0,148	0
L_{5K}	0	–0,07	0,31	–0,017	0,4	–0,013	0,267	0,133	0



■ ***Рис. 16.*** *Изменения мотивируемости от интервала времени принятия решения*

■ ***Таблица 22.*** *Изменение осведомленности*

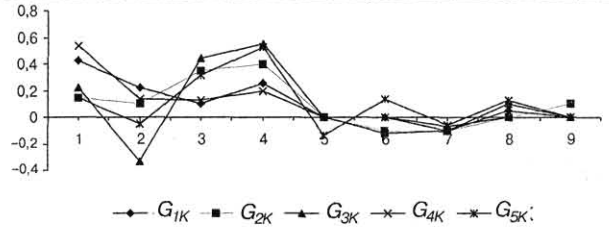
K	1	2	3	4	5	6	7	8	9
O_{1K}	–1	0,06	0,06	0,14	0,2	–0,02	0,04	0,48	–0,46
O_{2K}	0,48	0,055	0,219	0,054	0,192	–0,301	0,054	0,055	0,069
O_{3K}	0	–0,233	0,6	0,034	0,166	0,1	0,07	0	0,03
O_{4K}	0,235	0,256	0,097	0,294	0	0,088	0	0,03	0
O_{5K}	0	0	0,111	–0,889	0,889	–0,222	0,333	0	0



■ ***Рис. 17.*** *Изменения осведомленности от интервала времени принятия решения*

■ ***Таблица 23.*** *Изменение знания*

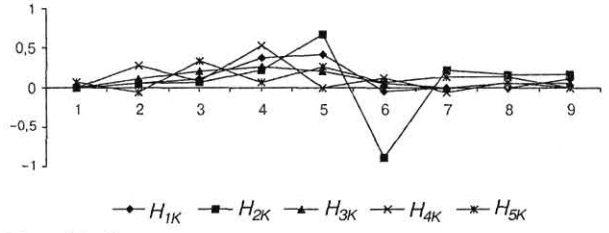
K	1	2	3	4	5	6	7	8	9
G_{1K}	0,425	0,223	0,1	0,252	0	0	–0,1	0,048	0
G_{2K}	0,148	0,1	0,352	0,4	0	–0,1	–0,1	0	0,1
G_{3K}	0,221	–0,336	0,443	0,557	0	–0,115	–0,106	0,106	0
G_{4K}	0,537	0,132	0,131	0,2	0	0	–0,068	0	0
G_{5K}	0,142	–0,053	0,317	0,526	–0,132	0,132	–0,063	0,131	0



■ ***Рис. 18.*** *Изменения знания от интервала времени принятия решения*

■ ***Таблица 24.*** *Изменение понимания*

K	1	2	3	4	5	6	7	8	9
H_{1K}	0	0,051	0,105	0,371	0,422	–0,055	0	0	0,106
H_{2K}	0	0,053	0,058	0,222	0,667	–0,889	0,222	0,165	0,169
H_{3K}	0	0,105	0,211	0,262	0,211	0,055	0	0,055	0,051
H_{4K}	0	0,276	0,076	0,53	0	0,118	–0,061	0,061	0
H_{5K}	0,069	–0,069	0,335	0,064	0,266	0,069	0,133	0,133	0



■ ***Рис. 19.*** *Изменения понимания от интервала времени принятия решения*

$\Delta O_i(K) = (O_{iK+1} - O_{iK})/(O_i^{max} - O_i^{min})$ – изменение осведомленности (табл. 22, рис. 17).

$$J_{Gi} = (\sum_K (\Delta G_i(K)) / \sum_K (\Delta L_i(K)), \qquad (16)$$

где $\Delta G_i(K) = (G_{iK+1} - G_{iK})/(G_i^{max} - G_i^{min})$ – изменение знания (табл. 23, рис. 18).

$$J_{Hi} = (\sum_K (\Delta H_i(K)) / \sum_K (\Delta L_i(K)), \qquad (17)$$

где $\Delta H_i(K) = (H_{iK+1} - H_{iK})/(H_i^{max} - H_i^{min})$ – изменение понимания (табл. 24, рис. 19).

Из зависимостей, приведенных на рис. 16–20, видно, что в процессе тестирования и раздумий у претендентов значительно изменяются осведомленность, знания и понимания, а рассчитанные по предложенным формулам компоненты *вектора обучаемости* существенно различны у тестируемых претендентов. Поэтому предложенная *векторная оценка обучаемости* может использоваться при отборе претендентов на роль оператора ЧМС. Кроме того, иногда можно использовать приближенную, *среднюю оценку обучаемости* претендентов при принятии решений в условиях неопределенности (табл. 26, рис. 21), вычисляемую следующим образом:

$$J_i = 1/3\,(k_O J_{Oi} + k_G J_{Gi} + k_H J_{Hi}), \qquad (18)$$

где k_O, k_G, k_H – весовые коэффициенты, которые в рассматриваемом примере равны единице.

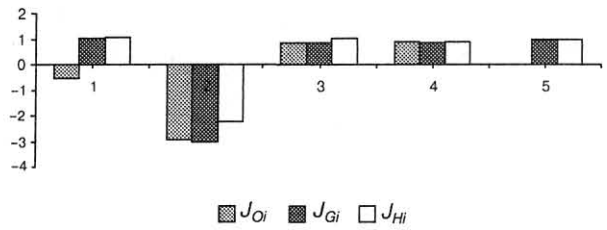
Из диаграммы, приведенной на рис. 21, видно, что средняя оценка обучаемости позволяет легко расставить претендентов по порядку от лучшего к худшему: 3, 4, 5, 1, 2. Следовательно, при условиях рассматриваемого примера средняя оценка обучаемости вполне может быть использована при отборе претендентов к работе в ЧМС по результатам тестирования.

Таким образом, при учете обучаемости вектор интеллекта будет иметь семь компонент (табл. 27, рис. 22), достаточно полно характеризующих способности претендентов для работы в ЧМС, соответствующей тестовым заданиям.

В ряде случаев можно использовать приближенную, *среднюю оценку интеллекта* претендентов при принятии решений в условиях неопределенности (табл. 28, рис. 23), вычисляемую так:

■ ***Таблица 25.*** *Компоненты вектора обучаемости*

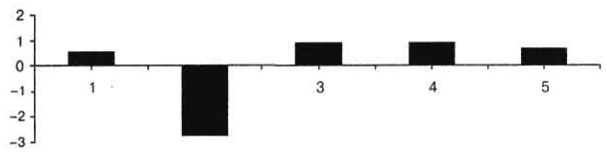
I	1	2	3	4	5
J_{oi}	–0,535	–2,94	0,826	0,897	0
J_{gi}	1,02	–3,02	0,83	0,836	1
J_{hi}	1,07	–2,24	1,02	0,897	1



■ ***Рис. 20.*** *Компоненты вектора обучаемости*

■ ***Таблица 26.*** *Средняя оценка обучаемости*

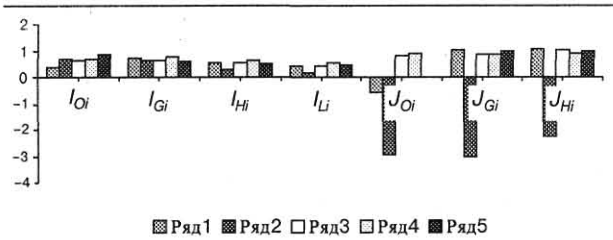
I	1	2	3	4	5
Ji	0,518	–2,73	0,892	0,877	0,667



■ ***Рис. 21.*** *Диаграмма средних оценок обучаемости I_i*

■ ***Таблица 27.*** *Компоненты вектора интеллекта*

I	1	2	3	4	5
I_{Oi}	0,376	0,671	0,634	0,697	0,844
I_{Gi}	0,723	0,64	0,633	0,78	0,617
I_{Hi}	0,536	0,311	0,547	0,633	0,507
I_{Li}	0,434	0,154	0,414	0,536	0,467
J_{Oi}	–0,535	–2,94	0,826	0,897	0
J_{Gi}	1,02	–3,02	0,83	0,836	1
J_{Hi}	1,07	–2,24	1,02	0,897	1



■ ***Рис. 22.*** *Компоненты вектора интеллекта*

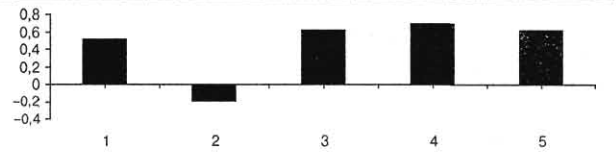
$$I_i = 1/7\,(k_O J_{Oi} + k_G J_{Gi} + k_H J_{Hi} + k_L I_{Li} + C_O J_{Oi} + C_G J_{Gi} + C_H J_{Hi}), \quad (19)$$

где k_O, k_G, k_H, k_L, C_O, C_G, C_H – весовые коэффициенты, которые в рассматриваемом примере равны единице.

Из диаграммы, приведенной на рис. 23, видно, что средняя оценка интеллекта позволяет легко расставить претендентов по порядку от лучшего к худшему: 4, 5, 3, 1, 2. Следовательно, при условиях рассматриваемого примера средняя оценка интеллекта вполне может быть использована при отборе претендентов к работе в ЧМС по результатам тестирования.

■ ***Таблица 28.*** *Средняя оценка интеллекта*

I	1	2	3	4	5
I_i	0,517	–0,19	0,624	0,703	0,62



■ ***Рис. 23.*** *Диаграмма средних оценок интеллекта I_i*

Заключение

Введенные оценки результатов тестирования претендентов на работу в качестве оператора ЧМС позволяют более полно учитывать особенности их мышления, так как они имеют статические и динамические компоненты, учитывающие как способности к принятию решений в условиях неопределенности, так и способности к обучению в процессе работы. В ряде случаев векторные оценки могут быть заменены средними оценками интеллекта, позволяющими легко ранжировать претендентов от лучших к худшему. Однако при этом часть информации теряется. В частности, в рассматриваемом тестовом примере статические и динамические средние оценки дают разное распределение претендентов по степени их пригодности к работе в данной ЧМС. Последнее может означать, что при длительной работе в качестве оператора данной ЧМС оператор № 3 окажется более перспективным, чем, скажем, операторы № 5 или № 4, имеющие на момент окончания тестирования лучшие средние оценки интеллекта, так как у оператора № 3 наивысшая средняя оценка обучаемости.

Представляется, что более тонкое исследование характера изменения во времени предложенных оценок претендентов по результатам тестирования поможет выявить ряд важных для работы в ЧМС психологических особенностей операторов, например таких, как решительность, настойчивость, уверенность и др. [8]. При этом не менее важной проблемой является формирование наиболее информативных тестовых задач по конкретной ЧМС.

Литература

1. **Управление в** условиях неопределенности./ Под ред. А. Е. Городецкого. – СПб.: СПбГТУ, 2002. – 398 с.
2. **Акофф Р., Эмери Ф.** О целеустремленных системах. – М.: Сов. Радио, 1974. – 269 с.
3. **Chein, Isadore.** On the Nature of Intelligence//The Journal of General Phychology. – 32 (1945). – P. 11–126.
4. **Getzel J. W. and Jackson P. W.** Creativity and Intelligence. – New York: John Wiley and Sons, 1962. – 290 p.
5. **Городецкий А. Е.** Оценка качества экспертных систем, как систем целеустремленных действий//Проблемы физической метрологии. – СПб.: Изд-во KN, 1996. – С. 118–126.
6. **Городецкий А. Е.** Об использовании ситуации привычности для ускорения принятия решения в интеллектуальных информационно-измерительных системах // Физическая метрология: теоретические и прикладные аспекты. – СПб.: Изд-во KN, 1996. – С. 141–151.
7. **Gorodetsky A. E.** Fuzzy Decision Making in Design on the Basis of the Hubituality // Fuzzy System Design, L. Reznik, V. Dimitrov, J. Kasprzyk, Physica Verlag. – 1998.
8. **Сидоренко Е. В.** Методы математической обработки в психологии. – СПб.: ООО «Речь», 2002. – 350 с.

УДК 519.711

КОНСТРУИРОВАНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ РЕГУЛЯТОРОВ

М. В. Бураков,
канд. техн. наук, доцент
А.С. Коновалов,
д-р техн. наук, профессор
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

В статье рассматриваются современные технологии интеллектуального управления динамическими объектами. Описывается методика синтеза регуляторов с базами знаний, опирающаяся на применение имитационного моделирования и генетического алгоритма. Приводится алгоритм нечеткого моделирования нелинейных динамических объектов. Обосновывается методика конструирования интеллектуальных регуляторов как сообщества конкурирующих интеллектуальных агентов.

The purpose of this paper is to present a modern technologies of intellectual control of dynamic objects. The technique of synthesis of regulators with knowledge bases basing on application of simulation modeling and genetic algorithm is described. The algorithm of fuzzy modelling of nonlinear dynamic objects is resulted. The technique of designing of intellectual regulators as communities of the competing intellectual agents is proved.

Введение

Родовым признаком интеллектуальных регуляторов (ИР) является присутствие базы знаний (БЗ) о процессе управления объектом. Хотя знания могут иметь разную форму представления, для человека наиболее естественным является использование продукционных правил. Однако экспертного опыта обычно бывает недостаточно для подробного описания всей БЗ. Поэтому ИР конструируются в результате процесса самоорганизации, в ходе которого большую роль играет имитационное моделирование. Рис. 1 иллюстрирует общую схему обучения ИР, где $Y(t)$ и $Y^*(t)$ – реальный и желаемый выход модели; $\varepsilon(t)$ – ошибка управления; $U(t)$ – сигнал управления; $G(t)$ – тестовый сигнал.

Для успешного синтеза ИР нужно обладать адекватной имитационной моделью (ИМ), выбрать структуру регулятора и алгоритм коррекции его параметров.

Если объект хорошо изучен и понятны физические принципы его функционирования, то ИМ может быть построена на основании математического описания. Если же известны только входные и выходные соотношения объекта, то можно попытаться описать его структуру с помощью универсальных аппроксиматоров – нейронной сети (НС) или нечеткой системы.

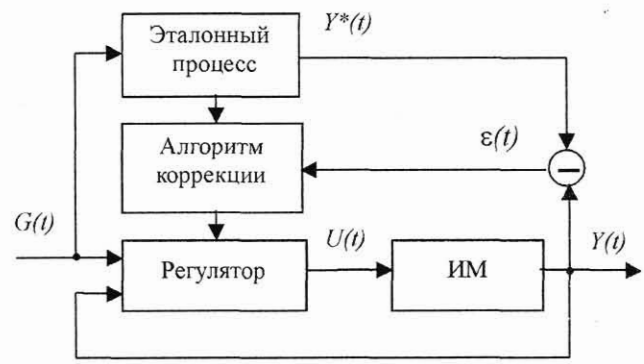
Эволюционная методика синтеза ИР с помощью генетического алгоритма (ГА) предполагает поиск субоптимального решения на базе популяции конкурирующих альтернативных описаний БЗ. Как показано ниже, этот подход естественным образом может быть расширен для решения задачи конструирования ИМ сложного объекта управления, а также при разработке адаптивных систем управления, способных функционировать в условиях неполной информации о параметрах объекта и (или) внешней среды.

Интеллектуальное управление

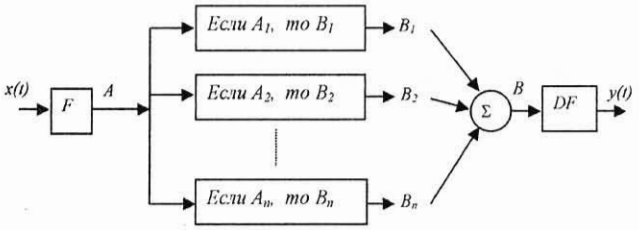
Рассмотрим наиболее популярные варианты представления знаний в интеллектуальных системах.

Ядром нечеткого логического регулятора (НЛР) является БЗ, состоящая из продукционных правил. Существуют два основных подхода к описанию нечетких правил: 1) модель *Mamdani* [1]; 2) *TSK*-модель (*Takagi, Sugeno, Kang*) [2]. Соответственно можно говорить о НЛР двух типов – *M*-типа и *TSK*-типа.

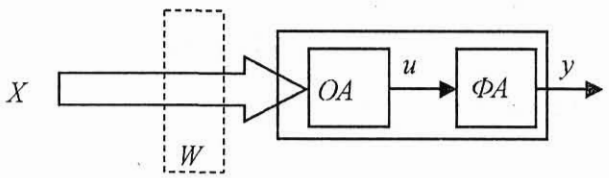
В НЛР *M*-типа консеквенты нечетких правил описываются как термы лингвистической переменной (ЛП) «Сигнал управления», а в НЛР *TSK*-типа консек-



■ ***Рис. 1.*** *Обучение интеллектуального регулятора*



■ ***Рис. 2.*** *Система нечетких правил*



■ ***Рис. 3.*** *Обобщенное описание нейрона*

вент представляет собой полиномиальную функцию входов.

Рассмотрим нечеткий регулятор *M*-типа. В простейшем варианте здесь каждое *i*-е правило имеет одну посылку A_i и одно заключение B_i (рис. 2, где *F* и *DF* – операции фаззификации и дефаззификации). Правила параллельно обрабатывают входную информацию *x(t)*, так что каждое правило порождает свой собственный элементарный выходной сигнал B_i. Общий выходной сигнал *B* является объединением (суммой) этих элементарных сигналов.

«Хорошая» система нечетких правил должна удовлетворять требованиям полноты и непротиворечивости. Полнота предполагает отсутствие во входном пространстве [области определения *x(t)*] «белых пятен».

Непротиворечивость означает, что в базе не должно быть правил, которые имели бы при сходных посылках существенно различные заключения.

Главное достоинство системы нечетких правил заключается в плавности перехода от ситуации к ситуации. Кроме того, отдельное правило контролирует только некоторую область фазового пространства, так что закон управления в целом может быть нелинейным, и существует возможность независимого добавления или изменения отдельных правил (при соблюдении требований полноты и непротиворечивости).

Базовым элементом нейронного регулятора является искусственный нейрон (который в силу своей простоты имеет лишь грубое сходство с биологическим нейроном). Обобщенная модель нейрона показана на рис. 3 (где *OA* – операция агрегирования, которая сравнивает входной вектор и вектор весов, возвращая количественную оценку этого сходства *u*; *ФА* – функция активации).

Нейрон реализует нелинейное отображение

$$X \in R_n \to y \in R_1.$$

Векторы *X* и *W* обычно нормализуются. Для измерения сходства могут быть использованы различные подходы:

1) вычисление скалярного произведения векторов *X* и *W*

$$u = XW = \sum_{i=1}^{n} x_i w_i;$$

2) вычисление эвклидова расстояния между векторами *X* и *W*

$$u = \sqrt{(x_1 - w_1)^2 + (x_2 - w_2)^2 + \ldots + (x_n - w_n)^2};$$

3) вычисление максимального отклонения

$$u = \max(|x_1 - w_2|, |x_2 - w_2|, \ldots |x_n - w_n|) .$$

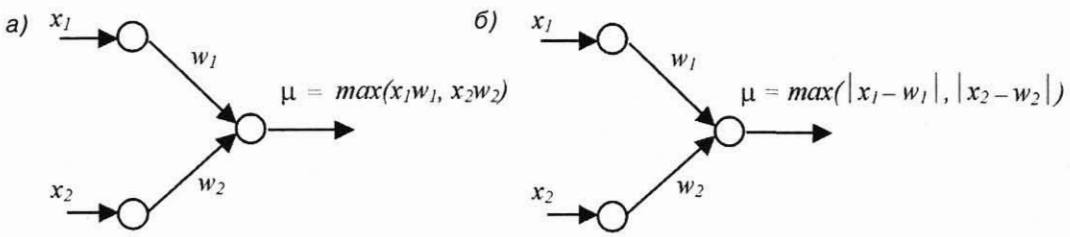
Активационная функция *ФА* выбирается, как правило, нелинейной (сигмоидная функция, линейная с насыщением и т. д.). Заметим, что нейрон с тремя входами, скалярным произведением в качестве измерения сходства и линейной активационной функцией соответствует обычному ПИД-регулятору.

Механизм «симбиоза» нечетких правил и НС достаточно прост в силу внутреннего сходства этих двух разных систем описания знаний [3].

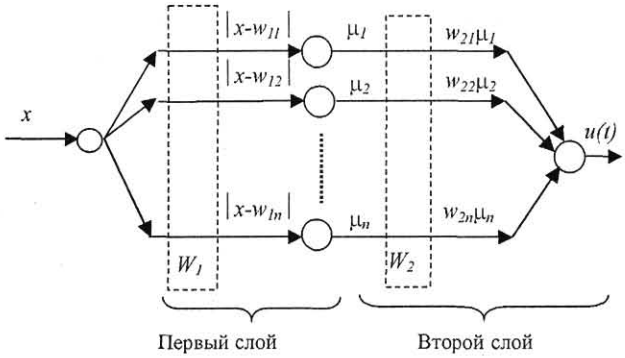
Описание искусственного нейрона может быть выполнено с помощью понятий *T*-нормы и *S*-нормы (обобщенные операции *AND* и *OR*):

$$u = \overset{n}{\underset{i=1}{S}}[w_i T x_i] .$$

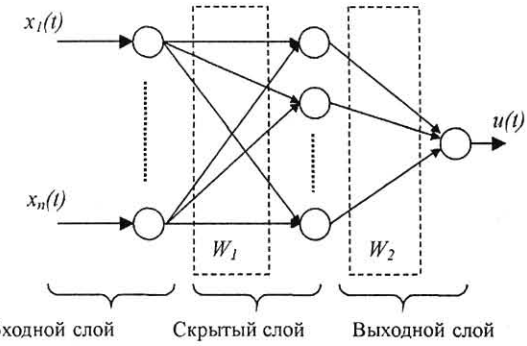
При использовании в качестве *T*-нормы произведения, а в качестве *S*-нормы функции max нейрону с двумя входами соответствует рис. 4, *а*.



■ ***Рис. 4.*** *«Нечеткие» нейроны OR и AND*



■ ***Рис. 5.*** *Реализация системы правил нейронной сетью*



■ ***Рис. 6.*** *НС для описания правил с n посылками*

Очевидно, нейрону на рис. 4, *а* можно сопоставить логическую формулу:

$$(x_1 = w_1)\ OR\ (x_2 = w_2).$$

Нейрон *OR* требует сходства хотя бы одного из входных сигналов с соответствующим весовым вектором для того, чтобы выходное значение *u* было близко к единице.

Однако посылки нечетких правил чаще связаны операцией *AND*. Нечеткий нейрон *AND*, наоборот, должен требовать, чтобы все входные сигналы (посылки) были близки соответствующим весовым векторам, поэтому здесь выгодно использовать критерий максимального отклонения (рис. 4, *б*).

При реализации нейрона *AND* можно использовать измерение декартова расстояния, но этот вариант не вполне отвечает представлениям о работе нечеткого правила, в котором слабое соответствие хотя бы одной посылки должно отражаться на общем соответствии посылок.

Таким образом, структура рис. 2 получает наиболее естественное воплощение при использовании НС, показанной на рис. 5.

НС на рис. 5 описывает систему правил с одной посылкой, здесь веса первого слоя w_{1i} отвечают за описание посылок, а веса второго слоя w_{2i} отвечают за описание заключений продукционных правил.

Рассмотрим стандартный механизм работы НЛР *M*-типа для объекта с *n* входов и одним выходом, реализующего отображение $X \to R$, где $X \in R^n$. Пусть входы обозначаются $x_1, x_2, x_3, \ldots, x_n$.

Для каждого x_i определено *m* функций принадлежности μ_{A_i}, каждая из которых реализует отображение входного сигнала в интервал [0, 1]. Они описывают A_i – термы ЛП, являющейся *i*-й посылкой нечеткого правила.

Так как всего *n* входов и каждый *i*-й вход имеет m_i функций принадлежности, то максимальное количество правил:

$$K = m_1 \times m_2 \times m_3 \times \ldots \times m_n. \qquad (1)$$

Декартово произведение $A_1, A_2, \ldots, A_n$ является нечетким множеством в пространстве *X* с функцией принадлежности, определяемой обычно в виде:

$$\mu_{A_1 \times A_2 \times \ldots A_n}(x_1, x_2, \ldots, x_n) = \\ = \min[\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_n}(x_n)]. \qquad (2)$$

Здесь каждая посылка представлена нечетким множеством A_i, а μ_{A_i} описывает степень соответствия входа x_i терму A_i.

Нечеткое правило описывается следующим образом:

правило *i*: если (x_1 есть A_1^i) и... и (x_n есть A_n^i),
то *u* есть u_i,

где A_j^i – *j*-я нечеткая посылка правила; u_i – одноточечное нечеткое множество.

Степень запуска *i*-го правила μ_i получается по формуле (2).

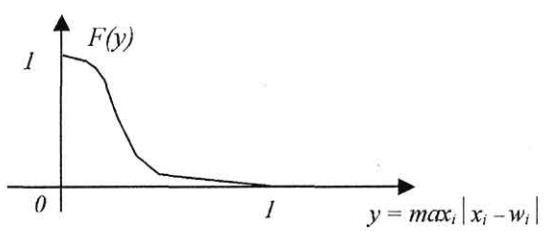
Затем выходные сигналы всех правил комбинируются с помощью стратегии дефаззификации. В настоящее время предложены десятки методов дефаззификации [4], но обычно используется следующая формула (где *N* – количество правил):

$$u = \frac{\sum_{i=1}^{N} \mu_i \cdot a_i}{\sum_{i=1}^{N} \mu_i}. \qquad (3)$$

В случае, когда правила продукционной системы имеют *n* посылок, эквивалентная НС преобразуется к виду, показанному на рис. 6.

Активационные функции нейронов первого слоя должны выдавать сигнал, величина которого обратно пропорциональна максимальному отклонению посылок правила, поэтому в качестве активационной функции здесь можно использовать вариант, представленный на рис. 7.

Нейрон второго слоя должен решать задачу дефаззификации, поэтому его активационная функция должна реализовывать формулу (3).



■ ***Рис. 7.*** *Активационная функция нейрона первого слоя*

Таким образом, двухслойная НС может использоваться для описания системы продукционных правил – наиболее простого и удобного способа представления знаний. Однако эффективное использование НС как основы ИР требует решения ряда проблем, в числе которых:

– определение необходимого количества нечетких правил (т. е. структуры НС); некоторые подходы к решению этой проблемы изложены в [5] и требуют анализа результатов имитационного моделирования;

– выбор параметров НС; эта задача решается в рамках структуры, показанной на рис. 1;

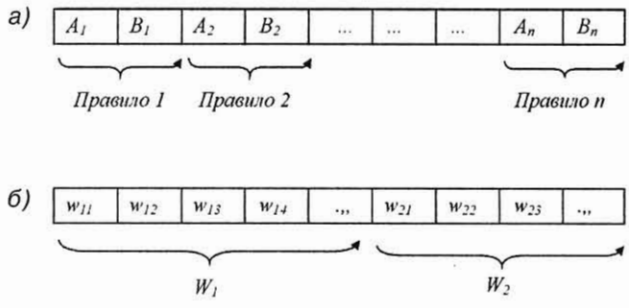
– проблема обеспечения адаптивности ИР; обученная НС трудно поддается оперативной коррекции – процесс обучения должен быть выполнен заново при изменении задачи управления или параметров объекта; иначе говоря, требуется решение известной «проблемы стабильности-пластичности» [6] НС.

Эволюционный синтез регуляторов

БЗ определяет свойства интеллектуального регулятора. Формирование «хорошей» базы правил НЛР, равно как и настройка НС, являются сложными задачами, для решения которых требуется сочетать имитационное компьютерное моделирование и поисковые процедуры в пространстве параметров с целью минимизации заданного критерия качества. Градиентные методы поиска эффективны для унимодальных целевых функций и при малом пространстве поиска. Если же функция мультимодальная и (или) разрывная, то применяют методы случайного поиска, развитием которых является генетический алгоритм (ГА) [7, 8]. Его главное отличие от методов случайного поиска заключается в том, что он накапливает информацию о прошлых полученных решениях.

Генетический алгоритм моделирует естественный отбор – процесс выживания и воспроизведения организмов, наиболее приспособленных к условиям среды, и гибели в ходе эволюции неприспособленных. Резкие, внезапные мутации считаются решающим фактором эволюции, сразу ведущим к возникновению нового вида. Возникают мутации под воздействием на организм внешней среды.

Основой эволюционных процессов в органическом мире служит популяция – большая совокупность особей, каждая из которых эволюционирует самостоятельно, подчиняясь генетическим факторам.



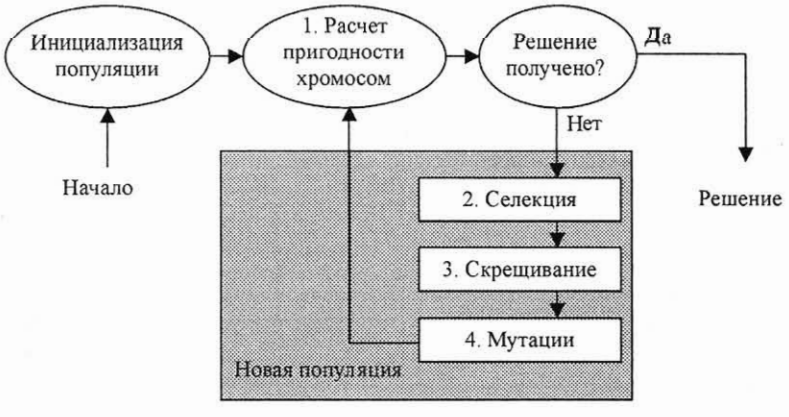
■ ***Рис. 8.*** *Способы описания хромосом*

Каждая особь популяции является обладателем уникального набора признаков, носителем которого являются хромосомы, состоящие из генов. Гены – это материальные структуры, ответственные за формирование признаков организма. Генотип (геном) – совокупность всех генов организма. Фенотип – совокупность всех признаков организма.

При конструировании ИР признаками являются параметры регулятора. Коды этих параметров образуют хромосомы, которые в процессе итераций тестируются и видоизменяются.

Для кодирования параметров могут использоваться вещественные числа или двоичный алфавит: {0,1}. Цепочка битов длиной m может рассматриваться как хромосома, в которой отдельные позиции (биты) выступают в качестве генов. Хромосома для структур, показанных на рис. 2 и 6, будут иметь соответственно следующий вид, показанный на рис. 8, *а* и *б*.

В общем случае, если длина цепочки равна m битов, то максимальный размер популяции оказывается 2^m. При использовании популяции такого размера достаточно протестировать все решения и выбрать лучшее из них. Однако проблема заключается в том, что значение 2^m в реальных задачах очень велико, а процедура тестирования хромосомы занимает вполне определенное время, поэтому реальный допустимый размер популяции $N<<2^m$.



■ ***Рис. 9.*** *Общая структура генетического алгоритма*

Работа ГА управляется тремя генетическими операторами: селекция, скрещивание, мутация (рис. 9).

Первоначальная генерация популяции осуществляется случайным образом в исследуемой области решений. После этого начинается циклическая работа ГА. На первом шаге происходит тестирование всех по очереди хромосом популяции, после чего каждая хромосома получает фиксированный на данном шаге признак – относительную пригодность (ОП) (*fitness function*). В рамках структуры рис. 1 ОП описывает расхождение между желаемым (эталонным) процессом на выходе системы и реальным выходом при подаче на вход тестирующего сигнала конечной длительности.

Если рассмотреть n моментов времени, то получается n обучающих пар: y и z, где y – реальный выход системы, z – желаемый выход. Тогда ошибка может быть определена в виде:

$$\varepsilon = \sum_{i=1}^{n} (y_i - z_i)^2 \text{ или } \varepsilon = \sum_{i=1}^{n} |y_i - z_i|, \qquad (4)$$

и ОП рассчитывается по формуле

$$ОП = \frac{1}{1+\varepsilon}.$$

Таким образом, для получения ОП хромосомы требуется выполнить моделирование всего переходного процесса в системе. Операция оценивания пригодности является основополагающей для правильной селекции, а значит и для эффективной работы ГА. Однако простые формулы (4) не всегда пригодны для расчета ОП. В общем случае задача присвоения ОП фактически является задачей распознавания образов [9].

Селекция (отбор) является основной генетической операцией. Здесь возможны два разных подхода: либо при селекции отбираются отдельные признаки хромосом [10, 11], либо в результате селекции отбираются целиком хромосомы, пригодность которых превышает некоторый порог. На основании селекции происходит генерация новой популяции (популяции потомков).

Скрещивание выполняется с целью комбинирования и смешения признаков в популяции потомков. Мутация небольшого количества генов призвана сообщить потомкам новые признаки, которые могли отсутствовать в родительской популяции [12 и др.].

Схема ГА, показанная на рис. 8, предполагает, что количество потомков равно количеству родителей, и после выполнения мутации новая популяция полностью сформирована, так что может начинаться новый цикл развития. Однако при этом существует опасность потери лучших хромосом, поскольку все потомки могут оказаться хуже своих родителей. Для избежания этой опасности часто используются следующие простые приемы:

– выделяется элита популяции (около 10 %), которая попадает в новую популяцию без изменения;

– выделяются наихудшие хромосомы (также около 10 %), которые замещаются хромосомами, сгенерированными случайным образом.

Остальные хромосомы подвергаются генетическим операциям.

Для анализа текущего состояния популяции и более «справедливого» назначения ОП хромосом могут быть введены следующие критерии:

– селективное давление – отношение ОП наилучшей хромосомы к средней ОП всей популяции;

– селективная распространенность – количество возможных потомков одной хромосомы.

Существуют два подхода к присвоению ОП хромосом: пропорциональное и ранжированное присвоение.

Если численная оценка, которая связывается с каждой хромосомой, используется без изменения в операции селекции, то говорят о пропорциональном присвоении ОП.

Пропорциональное присвоение ОП может вызывать такие негативные явления, как стагнация при низком селективном давлении или преждевременное сужение зоны поиска (попадание в локальный минимум).

Ранжированное присвоение ОП предполагает дополнительную сортировку хромосом популяции и пересчет ОП, полученного на основании целевой функции, так что новое значение ОП зависит только от положения (ранга) хромосомы. Это позволяет контролировать селективное давление и ограничить количество потомков одной хромосомы.

Пусть хромосомы популяции отсортированы так, что лучшая хромосома имеет номер N, а худшая – номер 1 (где N – размер популяции). Тогда линейное ранжирование всех хромосом можно выполнить с помощью формулы

$$ОП(i) = 2 - СД + 2(СД - 1)\frac{(i-1)}{(N-1)},$$

где $СД$ – селективное давление; $ОП$ – относительная пригодность; i – номер хромосомы.

Могут быть также предложены формулы для нелинейного ранжирования, с помощью которых можно повысить селективное давление.

Рекомендации по использованию ГА для синтеза нейронных и нечетких регуляторов изложены в работе [9]. На практике выгодно использовать двух-трехслойные НС, поскольку иначе усложняется процесс обучения и тестирования. Однако сложность структуры регулятора зависит, естественно, от сложности управляемого объекта, и простая НС не способна реализовать сложную управляющую функцию. В работах [13, 14] обосновывается подход, связанный с генетическим синтезом множества относительно простых регуляторов для разных областей фазового пространства сложного объекта управления.

Нечеткое моделирование

Задача кластеризации фазового пространства сложного объекта может выполняться для двух целей:

1) при моделировании объекта – с целью сопоставления каждому кластеру некоторой простой модели для описания выходного сигнала;

2) при конструировании регулятора – с целью поиска сигнала управления, который должен соответствовать каждому кластеру.

Рассмотрим первую задачу. Для ее решения может быть использован регулятор *TSK*-типа [15, 16].

Общий подход к моделированию сложных нелинейных систем заключается в линеаризации системы в заданной рабочей точке. Это может быть полезно для локального анализа, но не позволяет выполнять глобальный анализ поведения системы. *TSK*-модель позволяет агрегировать множество линеаризованных моделей для удобной аппроксимации сложной нелинейной системы.

Построение *TSK*-модели требует решения двух задач – структурной и параметрической идентификаций.

Структурная идентификация предполагает следующие шаги:

1) определение необходимого количества правил «*Если – То*»;

2) разбиение входного пространства с помощью множества функций принадлежности.

Параметрическая идентификация заключается в уточнении параметров функций принадлежности посылок и заключений нечетких правил.

Рассмотрим нелинейную систему с n входами и одним выходом:

$$y = f(x_1, x_2, \ldots, x_n).$$

Пусть известны m рабочих точек $x_1^j, x_2^j, \ldots, x_n^j$, $j = \overline{1,m}$, с помощью которых описывается m линеаризованных подсистем:

$$y^j = f^j(x_1, x_2, \ldots, x_n).$$

TSK-модель представляется в виде набора правил:

$$R^j: \text{if } (x_1 \text{ есть } A_1^j) \text{ и } (x_2 \text{ есть } A_2^j)$$
$$\text{и} \ldots \text{то } y^j = f^j(x_1, x_2, \ldots, x_n)$$

Выход *TSK*-модели описывается в виде

$$f_{fuzzy} = y = \sum_{j=1}^{m} b^j y^j,$$

где $b^j = \dfrac{\tau^j}{\sum_{j=1}^{m} \tau^j}$; $\tau^j = \prod_{i=1}^{n} A_i^j(x_i)$, τ^j – степень запуска j-го правила; y^j – линейная функция в j-й рабочей точке.

Функции принадлежности i-й входной переменной вокруг j-й рабочей точки описываются формулой

$$A_i^j(x_i) = \exp\left(-\frac{(x_i - \alpha_i^j)^2}{(\beta_i^j)^2}\right),$$

где α и β – заданные параметры.

Разбиение входного пространства на области, относящиеся к рабочим точкам, и связывание этих областей в рамках *TSK*-модели позволяет получить удобную в использовании модель нелинейной системы. Мощность такого подхода зависит от того, позволяет ли сложная система выполнить декомпозицию на совокупность более простых подсистем, каждая из которых может быть описана линейной моделью.

При обучении должны быть заданы пары входных и выходных данных (x^k, y^k), $x^k \in X \subset R^n$, $y^k \in Y \subset R$, которые сгенерированы неизвестной системой $g(x)$.

$$x = (x_1, x_2, \ldots, x_n)^T \in X \subset R^n.$$

Задача обучения заключается в поиске нечеткой *TSK*-модели $f_{fuzzy}(x^k)$, такой, что для заданного порога ошибки ε выполняется:

$$\left|g(x^k) - f_{fuzzy}(x^k)\right| = \left|g(x^k) - \sum_{j=1}^{m} \beta^j(x^k) f^j(x^k)\right| \le \varepsilon. \quad (5)$$

Необходимо использовать как можно больше данных, чтобы получить хорошую *TSK*-модель.

Обозначим заданное множество входных и выходных пар (x^k, y^k) как G^0.

Работа блока разбиения включает несколько шагов.

1. Определение линейной аппроксимирующей функции f_0. Здесь используется метод наименьших квадратов:

$$f_0(x) = B^T x;$$
$$B = (X^T X)^{-1} X^T Y,$$

где $X[n \times m]$, $Y[m \times 1]$, $B[n \times 1]$.

Вектор $B[n \times 1]$ представляет собой параметрические оценки. Для того чтобы система была невырожденной, должно выполняться условие $m \ge n$.

2. Поиск точки максимума ошибки x^* для группы входных данных G^0, где максимизируется значение

$$\left|f_0(x^k) - y^k\right|, \quad k = \overline{1,m}.$$

Поскольку построение f_0 базируется на усреднении данных, можно сделать вывод, что в точке x^* существует нелинейность.

3. Разделение групп данных на подгруппы.

После нахождения точки максимума ошибки

$$x^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$$

данные разделяются на две группы по всем входным переменным:

$$G_{i_1}^1: (x^{k_{i1}}, y^{k_{i1}}), \quad k_{i1} = 1, \ldots, c_{i1}, \quad \text{где } x_i > x_i^*;$$
$$G_{i_2}^1: (x^{k_{i2}}, y^{k_{i2}}), \quad k_{i2} = 1, \ldots, c_{i2}, \quad \text{где } x_i > x_i^*.$$

Здесь $i = \overline{1,n}$ и $n + 1 \le c_{i1}, c_{i2} \le m$ – размерности групп.

Число пар в каждой группе должно быть больше числа входных переменных.

После n разбиений мы будем иметь $2n$ подгрупп данных:

$$G_{ij}^1: (x^{k_{ij}}, y^{k_{ij}}), \quad i = \overline{1,n}, \quad j = 1,2, \quad k_{ij} = 1, \ldots, c_{ij}. \quad (6)$$

4. Определение значимости входных переменных линейной аппроксимирующей функции f_{ij} (где $i = 1, \ldots, n$, $j = 1,2$). Для этого выполняется расчет значения ρ_i:

$$\rho_i = \sqrt{\left[\frac{\sum_{k=1}^{m}\left(\sum_{j=1}^{2} f_{ij}(x^k)q_{ij}(x^k) - y^k\right)^2}{m}\right]},$$

где переключательная функция описывается формулой

$$q_{ij}(x^k) = \begin{cases} 1, & x^k = x^{kj} \\ 0, & x^k \neq x^{kj} \end{cases}.$$

Входная переменная x_i является наиболее значимым входом, если значение ρ_i для нее меньше, чем для всех остальных.

Разделение входного пространства на два подпространства относительно наиболее значимой входной переменной позволяет построить начальную *TSK*-модель, которая содержит два правила вида:

$$R^1: \text{ if } x_i \text{ есть } A_i^1, \text{ то } y^1 = a_0^1 + a_1^1 x_1 + \ldots + a_n^1 x_n;$$

$$R^2: \text{ if } x_i \text{ есть } A_i^2, \text{ то } y^2 = a_0^2 + a_1^2 x_1 + \ldots + a_n^2 x_n.$$

Очевидно, если эта модель достаточна для аппроксимации входной и выходной зависимости, то можно далее использовать ГА для оптимизации параметров этой модели. Дальнейшее разделение входного пространства не нужно, если выполняется условие (5), иначе необходимо дополнительное разделение входного пространства на два подпространства в соответствии с (6).

Если нет необходимости в разделении обоих групп входных данных, значит, нелинейность проявляется только в одной из них. Тогда дальнейшее разделение необходимо только для одной из групп данных.

В этом случае с каждой из групп данных G_{ij} ассоциируется величина

$$\rho_{ij} = \sqrt{\left(\frac{\sum_{k_{ij}=1}^{c_{ij}}\left(f_{ij}(x_{k_{ij}}) - y^{k_{ij}}\right)^2}{c_{ij}}\right)},$$

где $k_{ij} = 1,\ldots,c_{ij}$, (c_{ij} – размерность G_{ij}).

Таким образом, если с G_{ij} ассоциирована линейная функция f_{ij}, имеющая большое значение r_{ij}, то необходимо дальнейшее разделение пространства G_{ij}.

Оптимизация начальной *TSK*-модели может выполняться с помощью ГА. В ходе оптимизации корректируются параметры функций принадлежности или параметры линейных аппроксимирующих функций.

Заметим, что *TSK*-модель позволяет также аппроксимировать сложную управляющую функцию как набор локальных регуляторов, контролирующих разные области фазового пространства объекта. На границах областей действия регуляторов согласовываются. Приведенный алгоритм может использоваться для выбора структуры такого регулятора.

Мультиагентная система управления

Традиционный подход к конструированию адаптивных систем заключается во введении в систему управления еще одного контура – контура адаптации, который обеспечивает коррекцию закона управления при неудовлетворительном качестве функционирования. Подобная идея может быть использована и по отношению к ИР [17 и др.]. Однако на этом пути встречаются значительные трудности:

1) ГА связан с большими вычислительными затратами и обычно не может использоваться для адаптации при управлении в реальном времени;

2) в НЛР оперативная коррекция одного правила управления затруднена из-за того, что в процессе дефаззификации в каждый момент времени могут принимать участие множество правил;

3) в процессе адаптации трудно контролировать полноту и непротиворечивость управляющей БЗ, построенной на правилах;

4) сложная управляющая БЗ на основе НС плохо поддается коррекции в реальном времени, поскольку в ней трудно вычленить элементы, подлежащие изменению.

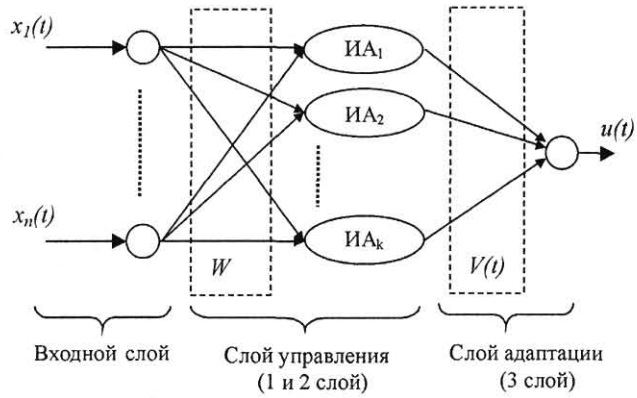
В связи с этим в последнее время большой интерес вызывает подход к созданию адаптивных ИР, связанный с использованием идеологии мультиагентных систем [18, 19 и др.].

Как было показано выше, двухслойная НС соответствует набору продукционных правил, описывающих БЗ нечеткого регулятора. Если же рассмотреть трехслойную НС, то ее можно уподобить (в рассматриваемом контексте) уже целому сообществу интеллектуальных агентов (ИА), решающих общую задачу (рис. 10).

Основная идея этого подхода заключается в том, что ИА, представляющие собой альтернативные описания БЗ, непрерывно получают информацию о поведении управляемого объекта. Каждый из ИА предлагает свой вариант управления в текущий момент времени (таким образом, ИА являются регуляторами-конкурентами). Все варианты оцениваются, так что на объект поступает наиболее предпочтительный вариант. С этой целью вектор весовых коэффициентов V должен изменяться во времени.

В зависимости от решаемой задачи в структуру, показанную на рис. 10, может вкладываться разное содержание. ИА могут описывать варианты стратегий управления в разных областях фазового пространства (ФП) системы, или описывать законы управления в зависимости от текущих параметров системы или внешней среды. В первом случае мультиагентная система (МАС) является обобщением *TSK*-регулятора, а во втором случае возникает адаптивная МАС (АМАС).

Рассмотрим задачи, которые могут быть решены в рамках МАС. Здесь происходит декомпозиция общей задачи управления на ряд подзадач, определенных



■ ***Рис. 10.*** *Мультиагентная управляющая система*

для подобластей ФП, в котором происходит движение управляемого объекта. Управление рассматривается как последовательность перемещений из одной области ФП в другую. Каждой области соответствует локальный регулятор. На границах областей действия регуляторов согласовываются [14]. Такой подход позволяет контролировать сложность и трудоемкость процесса обучения ИА. Если для какой-то области ФП не удается добиться нужного качества управления, то эта область должна быть разбита на подобласти.

В рамках АМАС необходимо использовать набор ИА, БЗ которых синтезированы при существенно различных параметрах модели [20]. Адаптация заключается в активизации ИА, наиболее адекватного текущей ситуации на объекте, и одновременном подавлении неадекватных в текущий момент времени ИА. Таким образом, закон изменения $V(t)$ является центральной проблемой при конструировании адаптивной МАС, для решения которой можно предложить два подхода – идентификационный и безидентификационный.

В первом случае структура МАС должна быть дополнена контуром идентификации, который на основании анализа входных и выходных сигналов пытается определить текущие параметры объекта и выбрать ИА, наиболее им соответствующий.

Во втором случае нужно анализировать только текущее качество управления и проводить поиск адекватного ИА. Для этих целей в работе [20] предлагается вычислять в реальном времени две функции – соответствия и полезности.

Функция соответствия должна описывать близость сигнала управления каждого ИА к сигналу управления, подаваемому на объект. Функция полезности оценивает текущее качество управления. При плохом качестве управления нужно подавлять активные на предыдущем этапе регуляторы и активизировать ранее «заторможенные» регуляторы, предлагающие альтернативные варианты управления.

Предложенная структура может быть, в частности, использована в задаче разработки системы торможения колесами при посадке самолета [21]. Здесь большое значение имеет состояние взлетно-посадочной полосы, которое заранее неизвестно. Полоса может быть мокрой, сухой, полусухой и т. д. – всего может быть выделен ограниченный набор состояний. Каждому состоянию полосы соответствует свой локальный регулятор. Стремительность процесса посадки затрудняет оперативную коррекцию правил или функций принадлежности. Использование АМАС позволяет оперировать набором конкурирующих ИА, так что в итоге включается в работу только один из них, адекватный текущей ситуации.

Заключение

Любая интеллектуальная система может рассматриваться как сообщество агентов, коллективное взаимодействие которых порождает общую реакцию системы. В НЛР в качестве простых агентов выступают отдельные правила управления, в нейронном регуляторе это отдельные нейроны. В мультиагентном интеллектуальном регуляторе в качестве агентов выступают целые БЗ, описывающие альтернативные стратегии управления.

Соответственно при конструировании нечеткого (или нейронечеткого) регулятора анализ экспериментальных данных имеет целью определение необходимого количества правил управления [5].

Если же конструируется интеллектуальная мультиагентная система, то сначала происходит анализ данных моделирования для определения областей фазового пространства, в которых могут функционировать локальные регуляторы (интеллектуальные агенты). На втором этапе может быть выполнен собственно синтез регуляторов. Такие регуляторы должны быть достаточно просты, чтобы не возникало проблем с их обучением и интерпретацией их поведения. Это позволяет контролировать сложность задачи синтеза системы управления в целом, оставляя возможность для контроля получаемых решений.

Описанный подход к синтезу интеллектуальных систем управления может быть перспективен при построении систем управления широким спектром динамических объектов и систем, в том числе – в авиационной технике [14, 21].

Литература

1. **Mamdani E. H.** Application of fuzzy algorithms for control of simple dinamic plant. – IEEE Proc., – 1974. – v.121. – № 12. – P. 1585–1588.
2. **Lee C. C.** Fuzzy logic in control systems: fuzzy logic controller – part 1 and 2. IEEE Transaction on System. – Man and Cybernetics. – 1990. – 20(2). – P. 404–435.
3. **Gupta M. M., Rao D. H.** On the principles of fuzzy neural networks // Fuzzy Sets and Systems. – Vol. 61. – 1984. – P. 1–18.
4. **Van Leekwijck W., Kerre E.E.** Defazzification: criteria and classification // Fuzzy sets and systems 108 (1999). – P. 159–178.
5. **Бураков М. В., Коновалов А. С.** Нейронечеткие системы управления // Информационно-управляющие системы. – №1. – 2002 г. – С. 2–7.
6. **Уосермен Ф.** Нейрокомпьютерная техника: Теория и практика: Пер. а англ. – М.: Мир, 1992. – 240 с.
7. **Holland J.** Adaptation in Natural and Artificial Systems: An Introductory Analysis with application to Biology, Control and Artificial Intelligence. – University of Michigan Press, 1975.
8. **Goldberg D. E.** Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA, 1989.
9. **Бураков М. В., Коновалов А.С.** Проектирование нейронных и нечетких регуляторов с помощью генетического алгоритма //Управление в условиях неопределенности. / Под ред. А. Е. Городецкого / СПб.: СПбГТУ, 2002. – 399 с.
10. **Захаров В. Н., Ульянов С. В.** Нечеткие модели интеллектуальных промышленных регуляторов и систем управления. Имитационное моделирование // Изв. РАН. Техническая кибернетика. – № 5. – 1994. – С. 168–204.
11. **Бураков М. В.** Синтез нейронного регулятора // Изв. АН/ Теория и системы управления. – 1999. – № 3. – С.140–145.
12. **Скурихин А. Н.** Генетические алгоритмы // Новости искусственного интеллекта. – № 4. – 1995. – С. 6–46.
13. **Бураков М. В., Коновалов А. С., Гиль А. В.** Алгоритмы синтеза интеллектуальных регуляторов // Тр. 2-й междунар. научно-практич. конф. «Информационные технологии в моделировании и управлении», 20–22 июня 2000 г. – СПб.: СПбГТУ. – С. 81–83.
14. **Бураков М. В.** Структура нейронечеткого регулятора / Изв. АН Теория и системы управления. – 2001. – № 6. – С. 160–165.
15. **T. Takagi, M. Sugeno.** Fuzzy identification of systems and its application to modeling and control // IEEE Trans. Systems Man Cybernet. – 15(1) 1985. P. 116–132.
16. **Baolin Wu, Xinghao Yu.** Fuzzy modelling and identification with genetic algorithm based learning // Fuzzy sets and systems 113 (2000). – P. 351–365.
17. **T. J. Procyk and E. H. Mamdani.** A linguistic self-organizing process controller, Automatica 15 (1979). – P. 15–30.
18. **Arvin Agah, Kazuo Tanie.** Fuzzy logic controller design utilizing multiple contending software agents // Fuzzy sets and Systems. – 106 (1999). – P. 121–130.
19. **Городецкий В. И.** Многоагентные системы: Современное состояние исследований и перспективы применения // Новости искусственного интеллекта. – 1996. – №1. – С. 44–59.
20. **Бураков М. В.** Механизм адаптации нечеткого регулятора // Изв. АН. Теория и системы управления. – 1998. – № 1. – С. 84–87.
21. **Бураков М. В., Коновалов А. С., Шумилов П. Е.** Нейронный регулятор в системе авиационной антиюзовой автоматики // V Всероссийская научно-технич. конф. «Нейроинформатика-2003». Москва, 29–31 января 2003 г.

УДК 303.732

ЛИНГВО–КОМБИНАТОРНОЕ МОДЕЛИРОВАНИЕ ПЛОХО ФОРМАЛИЗОВАННЫХ СИСТЕМ

М. Б. Игнатьев,
д-р техн. наук, профессор
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Рассматривается лингво-комбинаторное моделирование плохо формализованных систем, для которых существует лишь описание на естественном языке. Моделирование базируется на использовании ключевых слов, основных понятий, сложившихся в предметной области. Модель состоит из трех групп переменных – характеристик основных понятий, изменения этих характеристик и структурированной неопределенности в эквивалентных уравнениях, которая может быть использована для адаптации и управления. В качестве примеров рассматриваются модели города, организма и атмосферы.

This paper discusses utilization of a combinatorial simulation approach for a complex system modeling. When dealing with complex systems one has to consider that conditions and environment are not fully determined. In the course of this paper it is discussed how a poorly formalized system can be efficiently represented and modeled by combinatorial simulation. Practical applications are demonstrated on the examples of atoms, human organism, city and weather.

Построение лингво-комбинаторных моделей

Математические модели имеются лишь для небольшого числа реальных систем. Системы описываются, прежде всего, с помощью естественного языка. В настоящей работе предлагается способ перехода от описания на естественном языке к математическим уравнениям. Например, пусть имеется фраза

$$WORD_1 + WORD_2 + WORD_3. \qquad (1)$$

В этой фразе мы обозначаем слова и только подразумеваем их смысл. В сложившейся структуре естественного языка смысл не обозначается. Ввести понятие смысла можно в следующей форме

$$WORD_1 SENSE_1 + WORD_2 SENSE_2 + \\ + WORD_3 SENSE_3 = 0. \qquad (2)$$

Будем обозначать слова как A_i (от англ. Appearance), а смыслы – как E_i (от англ. Essence). Тогда уравнение (2) может быть представлено как

$$A_1E_1 + A_2E_2 + A_3E_3 = 0. \qquad (3)$$

Уравнения (2) и (3) являются моделями фразы (1). Если мы имеем математическое уравнение $F(x_1, x_2, x_3) = 0$, то можем получить форму (3) посредством дифференцирования этого уравнения. Тогда A_i будут частными производными, а E_i – производными по времени от переменных.

Эта модель является алгебраическим кольцом и мы можем разрешить уравнение (3) относительно A_i либо относительно E_i путем введения третьей группы переменных – произвольных коэффициентов U_s [1, 2, 3]:

$$\begin{aligned} A_1 &= U_1E_2 + U_2E_3; \\ A_2 &= -U_1E_1 + U_3E_3; \\ A_3 &= -U_2E_1 - U_3E_2 \end{aligned} \qquad (4)$$

или

$$\begin{aligned} E_1 &= U_1A_2 + U_2A_3; \\ E_2 &= -U_1A_1 + U_3A_3; \\ E_3 &= -U_2A_1 - U_3A_2, \end{aligned} \qquad (5)$$

где U_1, U_2, U_3 – произвольные коэффициенты, которые можно использовать для решения различных задач на многообразии (3). Например, если хотим достигнуть максимум по переменной x_3, то можем назначить произвольные коэффициенты $U_2 = -bA_1$, $U_3 = -bA_2$ и тогда получим

$$\begin{aligned} \frac{dx_1}{dt} &= U_1A_2 - bA_1A_3; \\ \frac{dx_2}{dt} &= -U_1A_1 - bA_2A_3; \\ \frac{dx_3}{dt} &= b(A_1A_1 + A_2A_2). \end{aligned} \qquad (6)$$

Если $b > 0$, то переменная x_3 устойчиво стремится к максимуму, а для манипуляции траекторией остается коэффициент U_1.

В общем случае, при n переменных и m многообразий, ограничений, число произвольных коэффициентов S будет равно числу сочетаний из n по $m+1$ [1] (таблица):

n\m	1	2	3	4	5	6	7	8
2	1							
3	3	1						
4	6	4	1					
5	10	10	5	1				
6	15	20	15	6	1			
7	21	35	35	21	7	1		
8	28	56	70	56	28	8	1	
9	36	84	126	126	84	36	9	1

$$S = C_n^{m+1},\ n > m. \qquad (7)$$

Число произвольных коэффициентов является мерой неопределенности и адаптивности. Лингво-комбинаторное моделирование заключается в том, что в конкретной предметной области выделяются ключевые слова, которые объединяются во фразы типа (1), на основе которых строятся эквивалентные системы уравнений с произвольными коэффициентами. В частном случае они могут быть дифференциальными уравнениями и при их исследовании может быть использован хорошо разработанный математический аппарат. Лингво-комбинаторное модели- рование включает все комбинации и все варианты решений и является полезным эвристическим приемом при изучении плохо формализованных систем [3–5].

Лингво-комбинаторное моделирование атомов

При построении лингво-комбинаторных моделей атомов будем исходить из ключевых базовых понятий, которые уже сложились в науке.

Рассмотрим в качестве примера атом водорода и в качестве ключевых слов возьмем слова «атом» (*Atom*), «протон» (*Proton*), «электрон» (*Electron*). Тогда фраза (1) будет иметь вид

$$Atom + Proton + Electron. \qquad (8)$$

В эквивалентных уравнениях (3), (4) и (5) A_1 – характеристика атома водорода; E_1 – изменение этой характеристики; A_2 – характеристика протона; E_2 – изменение этой характеристики; A_3 – характеристика электрона; E_3 – изменение этой характеристики. Для моделирования дейтерия используем ключевые слова «атом» (*Atom*), «протон» (*Proton*), «электрон» (*Electron*), «нейтрон» (*Neutron*):

$$Atom + Proton + Electron + Neutron \qquad (9)$$

и получим эквивалентные уравнения

$$\begin{aligned} E_1 &= U_1A_2 + U_2A_3 + U_3A_4; \\ E_2 &= -U_1A_2 + U_4A_3 + U_5A_4; \\ E_3 &= -U_2A_1 - U_4A_2 + U_6A_4; \\ E_4 &= -U_3A_1 - U_5A_2 - U_6A_3, \end{aligned} \qquad (10)$$

где $U_1, U_2, U_3, U_4, U_5, U_6$ – произвольные коэффициенты; A_1 – характеристика атома дейтерия; E_1 – изменение этой характеристики; A_2 – характеристика протона атома дейтерия; E_2 – изменение этой характеристики; A_3 – характеристика электрона атома дейтерия, E_3 – изменение этой характеристики; A_4 – характеристика нейтрона атома дейтерия; E_4 – изменение этой характеристики. В случае атомных реакций возможно превращение дейтерия в водород посредством трансформации уравнений (10) в уравнения (4).

Аналогичным образом возможно построение лингво-комбинаторных моделей всех известных элементов таблицы Менделеева и их изотопов и возможных новых элементов. Это еще один путь для компьютерного моделирования физико-химических реакций. При этом необходимо решать задачу верификации таких моделей применительно к конкретным системам.

Структурная стабильность, совокупность устойчивых связей объекта, обеспечивающих его целостность и тождественность самому себе, т. е. сохранение основных свойств при различных внешних и внутренних воздействиях, обеспечивается адаптационными возможностями атомных и молекулярных систем [6]. В представленных лингво-комбинаторных моделях адаптационные возможности систем определяются числом произвольных коэффициентов в структуре эквивалентных уравнений и наибольшая структурная стабильность достигается в зоне адаптационного максимума, который обнаруживается у различных систем с числом переменных больше шести [1, 2] (см. таблицу). Для удержания систем в зоне адаптационного максимума можно использовать различные методы – рост числа переменных, наложение и снятие ограничений, объединение систем в коллективы. Действительно, если имеем две системы

$$\begin{aligned} S_1 &= C_{n_1}^{m_1+1}; \\ S_2 &= C_{n_2}^{m_2+1}, \end{aligned} \qquad (11)$$

то путем наложения общих ограничений m_{col} получим коллектив

$$S_{col} = C_{n_1+n_2}^{m_1+m_2+m_{col}+1}. \qquad (12)$$

При этом в зависимости от конкретных параметров может быть $S_{col} > S_1 + S_2$, когда объединение в коллектив приводит к росту адаптационных возможностей, или $S_{col} < S_1 + S_2$, когда адаптационные возможности коллектива меньше суммы адаптационных возможностей исходных систем. Лингво-комбинаторное моделирование может явиться полезным инструментом при анализе и синтезе атомно-молекулярных систем.

Моделирование города

Если в качестве ключевых слов взять «население», «пассионарность», «территория», «производство», «экология и безопасность», «финансы», «внешние связи», то в соответствии с изложенной методикой уравнение города будет представлено следующим образом:

$$A_1E_1 + A_2E_2 + \ldots + A_7E_7 = 0, \qquad (13)$$

а эквивалентные уравнения будут иметь вид

$$E_1 = U_1A_2 + U_2A_3 + U_3A_4 + U_4A_5 + U_5A_6 + U_6A_7;$$
$$E_2 = U_1A_1 + U_7A_3 + U_8A_4 + U_9A_5 + U_{10}A_6 + U_{11}A_7;$$
$$E_3 = -U_2A_1 - U_7A_2 + U_{12}A_4 + U_{13}A_5 + U_{14}A_6 + U_{15}A_7;$$
$$E_4 = -U_3A_1 - U_8A_2 - U_{12}A_3 + U_{16}A_5 + U_{17}A_6 + U_{18}A_7;$$
$$E_5 = -U_4A_1 - U_9A_2 - U_{13}A_3 - U_{16}A_4 + U_{19}A_6 + U_{20}A_7;$$
$$E_6 = -U_5A_1 - U_{10}A_2 - U_{14}A_3 - U_{17}A_4 - U_{19}A_5 + U_{21}A_7;$$
$$E_7 = -U_6A_1 - U_{11}A_2 - U_{15}A_3 - U_{18}A_4 - U_{20}A_5 - U_{21}A_6, \quad (14)$$

где A_1 – характеристика населения, которая включает в себя характеристики здоровья, образования, занятости; E_1 – изменение этой характеристики; A_2 – характеристика пассионарности, устремлений групп населения; E_2 – изменение этой характеристики; A_3 – характеристика территории, включая наземные и подземные постройки, этот блок может быть геоинформационной системой; E_3 – изменение этой характеристики; A_4 – характеристика производства, включая оценку различных видов деятельности (научной, производственной, транспортной, торговой и др.); E_4 – изменение этой характеристики; A_5 – характеристика экологии и безопасности; E_5 – изменение этой характеристики; A_6 – характеристика финансов, финансовых потоков и запасов в городе; E_6 – изменение этой характеристики; A_7 – характеристика внешних связей города, включая оценку входящих и выходящих потоков людей, энергии, материалов, информации, финансов; E_7 – изменение этой характеристики; $U_1, U_2, \ldots, U_{21}$ – произвольные коэффициенты, которые могут быть использованы для управления и решения различных задач на многообразии (13). Эта модель используется в системах для поддержки принятия решений городскими властями [4].

Модель ментальных процессов

Обычно ментальные процессы характеризуются ключевыми словами «восприятие», «внимание», «память», «мышление», «язык», «эмоции», «управление движениями». Тогда структура эквивалентных уравнений будет иметь вид (14), где A_1 – характеристика восприятия; E_1 – изменение этой характеристики; A_2 – характеристика внимания; E_2 – изменение этой характеристики; A_3 – характеристика памяти; E_3 – изменение этой характеристики; A_4 – характеристика мышления; E_4 – изменение этой характеристики; A_5 – характеристика языка; E_5 – изменение этой характеристики; A_6 – характеристика эмоций; E_6 – изменение этой характеристики; A_7 – характеристика управления движениями; E_7 – изменение этой характеристики. Уравнения (14) определяют взаимодействие между различными составляющими ментальных процессов в рамках нашей модели. Из этой модели вытекает необходимость в блоке управления для манипуляции произвольными коэффициентами. Этот блок управления можно считать аналогом высшей психической структуры – личности. Ментальные процессы являются частью целостного организма.

Моделирование организма

Организм человека – очень сложная система, которую можно рассматривать на уровне молекул, клеток, органов. Для лечащего врача важно рассмотрение организма прежде всего на уровне органов и при построении лингво-комбинаторной модели мы будем исходить из общепринятого набора органов – «органы движения», «органы пищеварения», «органы дыхания», «мочеполовые органы», «кроветворная и лимфатическая системы», «центральная нервная система», «периферийная нервная система», «железы внутренней секреции», «кожа и сенсорные системы». Уравнение организма будет содержать девять переменных

$$A_1E_1 + A_2E_2 + \ldots + A_9E_9 = 0, \quad (15)$$

а структура эквивалентных уравнений будет иметь следующий вид:

$$E_1 = U_1A_2 + U_2A_3 + U_3A_4 + U_4A_5 + U_5A_6 + U_6A_7 + U_7A_8 + U_8A_9;$$
$$E_2 = -U_1A_2 + U_9A_3 + U_{10}A_4 + U_{11}A_5 + U_{12}A_6 + U_{13}A_7 + U_{14}A_8 + U_{15}A_9;$$
$$E_3 = -U_1A_1 - U_9A_2 + U_{16}A_4 + U_{17}A_5 + U_{18}A_6 + U_{19}A_7 + U_{20}A_8 + U_{21}A_9;$$
$$E_4 = -U_3A_1 - U_{10}A_2 - U_{16}A_3 + U_{22}A_5 + U_{23}A_6 + U_{24}A_7 + U_{25}A_8 + U_{26}A_9;$$
$$E_5 = -U_4A_1 - U_{11}A_2 - U_{17}A_3 - U_{22}A_4 + U_{27}A_6 + U_{28}A_7 + U_{29}A_8 + U_{30}A_9;$$
$$E_6 = -U_5A_1 - U_{12}A_2 - U_{18}A_3 - U_{23}A_4 - U_{27}A_5 + U_{31}A_7 + U_{32}A_8 + U_{33}A_9;$$
$$E_7 = -U_6A_1 - U_{13}A_2 - U_{19}A_3 - U_{24}A_4 - U_{28}A_5 - U_{31}A_6 + U_{34}A_8 + U_{35}A_9;$$
$$E_8 = -U_7A_1 - U_{14}A_2 - U_{20}A_3 - U_{25}A_4 - U_{29}A_5 - U_{32}A_6 - U_{34}A_7 + U_{36}A_9;$$
$$E_9 = -U_8A_1 - U_{15}A_2 - U_{21}A_3 - U_{26}A_4 - U_{30}A_5 - U_{33}A_6 - U_{35}A_7 - U_{36}A_8,$$

где $U_1, U_2, \ldots, U_{36}$ – произвольные коэффициенты, которые могут быть использованы для настройки модели; A_1 – характеристика органов движения; E_1 – изменение этой характеристики и т. д. Эта модель используется в страховой медицине [3].

Моделирование атмосферы

В качестве ключевых слов можно взять метеорологические элементы – «температура», «давление воздуха», «влажность воздуха», «скорость ветра», «направление ветра», «облачность», «осадки», «видимость (прозрачность атмосферы)», «температура почвы», «температура поверхности воды» – 10 переменных. В структуре эквивалентных уравнений этой системы будет содержаться 45 произвольных коэффициентов:

$$E_1 = U_1A_2 + U_2A_3 + U_3A_4 + U_4A_5 + U_5A_6 + U_6A_7 + U_7A_8 + U_8A_9 + U_9A_{10};$$
$$E_2 = -U_1A_1 + U_{10}A_3 + U_{11}A_4 + U_{12}A_5 + U_{13}A_6 + U_{14}A_7 + U_{15}A_8 + U_{16}A_9 + U_{17}A_{10};$$
$$E_3 = -U_2A_1 - U_{10}A_2 + U_{18}A_4 + U_{19}A_5 + U_{20}A_6 + U_{21}A_7 + U_{22}A_8 + U_{23}A_9 + U_{24}A_{10};$$
$$E_4 = -U_3A_1 - U_{11}A_2 - U_{18}A_3 + U_{25}A_5 + U_{26}A_6 + U_{27}A_7 + U_{28}A_8 + U_{29}A_9 + U_{30}A_{10};$$
$$E_5 = -U_4A_1 - U_{12}A_2 - U_{19}A_3 - U_{25}A_4 + U_{31}A_6 + U_{32}A_7 + U_{33}A_8 + U_{34}A_9 + U_{35}A_{10};$$
$$E_6 = -U_5A_1 - U_{13}A_2 - U_{20}A_3 - U_{26}A_4 - U_{31}A_5 + U_{36}A_7 + U_{37}A_8 + U_{38}A_9 + U_{39}A_{10};$$
$$E_7 = -U_6A_1 - U_{14}A_2 - U_{21}A_3 - U_{27}A_4 - U_{32}A_5 - U_{36}A_6 + U_{40}A_8 + U_{41}A_9 + U_{42}A_{10};$$
$$E_8 = -U_7A_1 - U_{15}A_2 - U_{22}A_3 - U_{28}A_4 - U_{33}A_5 - U_{37}A_6 - U_{40}A_7 + U_{43}A_9 + U_{44}A_{10};$$
$$E_9 = -U_8A_1 - U_{16}A_2 - U_{23}A_3 - U_{29}A_4 - U_{34}A_5 - U_{38}A_6 - U_{41}A_7 - U_{43}A_8 + U_{45}A_{10};$$
$$E_{10} = -U_9A_1 - U_{17}A_2 - U_{24}A_3 - U_{30}A_4 - U_{35}A_5 - U_{39}A_6 - U_{42}A_7 - U_{44}A_8 - U_{45}A_9.$$

В этой системе уравнений A_1 – характеристика температуры воздуха; E_1 – изменение этой характеристики; A_2 – характеристика давления; E_2 – изменение этой характеристики и т. д. U_1, U_2, ..., U_{45} – произвольные коэффициенты, наличие которых опре- деляет возможность управления характеристиками. Выявление возможности управления важно для подстройки модели и управления погодой.

Моделирование игр

Лингво-комбинаторный подход можно использовать и при моделировании игр, таких как шахматы и футбол. Рассмотрим простую футбольную ситуацию – два игрока и мяч, что можно описать как

Игрок 1 + *Игрок* 2 + *Мяч*.

Моделью этого выражения будет уравнение (3), где A_1 – характеристика игрока 1; E_1 – изменение этой характеристики; A_2 – характеристика игрока 2; E_2 – изменение этой характеристики; A_3 – характеристика мяча; E_3 – изменение этой характеристики.

Соответствующая эквивалентная система уравнений будет иметь вид (4), где, манипулируя произвольными коэффициентами, можно управлять поведением игроков и мяча. Если ввести новые переменные: A_4 – характеристика расстояния между игроком 1 и мячом; A_5 – характеристика расстояния между игроком 2 и мячом и их изменения, соответственно, то тогда вместо уравнения (4) получим уравнение

$$A_1E_1 + A_2E_2 + A_3E_3 + + A_4E_4 + A_5E_5 = 0.$$

Разрешив его относительно изменений E, получим систему уравнений

$$E_1 = U_1A_2 + U_2A_3 + U_3A_4 + U_4A_5;$$
$$E_2 = -U_1A_1 + U_5A_3 + U_6A_4 + U_7A_5;$$
$$E_3 = -U_2A_1 - U_5A_2 + U_8A_4 + U_9A_5;$$
$$E_4 = -U_3A_1 - U_6A_2 - U_8A_3 + U_{10}A_5;$$
$$E_5 = -U_4A_1 - U_7A_2 - U_9A_3 - U_{10}A_4,$$

где U_1,..., U_{10} – произвольные коэффициенты, манипулируя которыми можно обеспечить сближение игроков с мячом. Аналогичным образом моделируется поведение двух команд по 11 игроков в каждой. Этот подход был использован при моделировании поведения игроков-роботов.

Заключение

Лингво-комбинаторное моделирование – это универсальный метод моделирования плохо формализованных систем в самых различных областях науки, техники, в различных сферах человеческой деятельности. При каждом конкретном применении этого метода необходимо осуществлять верификацию модели, проверять ее на соответствие поведению реального объекта. Наличие произвольных коэффициентов и возможность расширения модели, возможность включения новых переменных, новых ключевых слов позволяют настраивать модель для моделирования сложных реальных объектов.

Литература

1. **Игнатьев М. Б.** Голономные автоматические системы. – М.; Л.: Изд. АН СССР, 1963. – 204 с.
2. **Ignatiev M. B.** Simulation of Adaptational Maximum Phenomenon in Developing Systems. – Proceedings of The SIMTEC'93 – 1993 International Simulation Technology Conference. – San Francisco, USA, 1993. – P. 41–42.
3. **Ignatyev M. B., Makina D. M., Petrischev N. N. and oth.** Global model of organism for decision making support. Proceedings of the High Performance Computing Symposium – HPC 2000, Ed. A. Tentner, 2000 Advanced Simulation Technologies Conference, Washington D.C., USA, 2000 – P. 66–71.
4. **Ignatyev M. B.** Linguo-combinatorial method for complex systems simulation. Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics. – Vol. XI, Computer science II, Orlando, USA, 2002 – P. 224–227.
5. **Ignatyev M. B., Pinigin G. I.** Linguo-combinatorial simulation of universe" XXV General Assembly of International Astronomical Union, Sydney, Australia, 2003 (www.astronomy2003.com)
6. **Бейдер Р.** Атомы в молекулах. – М.: Мир, 2001. – 465 с.

УДК 62-507

ИСКУССТВО ПРОГРАММИРОВАНИЯ ЛИФТА. ОБЪЕКТНО–ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ С ЯВНЫМ ВЫДЕЛЕНИЕМ СОСТОЯНИЙ

Л. А. Наумов,
магистрант
А. А. Шалыто,
д-р техн. наук, профессор
Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Уже около сорока лет одним из примеров, на которых Д. Кнут обучает «Искусству программирования», является задача управления лифтом. На основании невнятного технического задания, используя только словесное описание алгоритмов, он представляет в окончательной форме программу на языке низкого уровня. В предисловии к третьему изданию упомянутой книги приводятся слова Б. Гейтса: «За последние двадцать лет мир изменился». Настоящая работа призвана показать справедливость этого высказывания на примере задачи управления лифтом.

Near forty years, one of the examples, which was used by D. Knuth for teaching «The Art of Computer programming», is the task about lift control. Basing on indistinct description, using only verbal algorithms definitions, he gives source code of the program on the low-level language. In the preface to the third edition of mentioned book there are words of B. Gates: «During last twenty years the world has changed». The purpose of the present article is to show the correctness of last statement on the example of the task about lift control.

Когда б вы знали, из какого сора
Растут стихи, не ведая стыда ...
А. Ахматова

Введение

О книге Дональда Кнута «Искусство программирования» Билл Гейтс сказал: «Если вы считаете себя действительно хорошим программистом..., прочитайте "Искусство программирования"... Если Вы сможете прочесть весь этот труд, то вам определенно следует отправить мне резюме» [1]. Математики говорят, что «в "Корне" есть все!» [2]. Программисты могут то же самое сказать о Кнуте.

Д. Кнут, кроме математических основ программирования, пытается обучать также искусству прикладного программирования. С этой целью он создает виртуальную машину MIX, для программирования которой разработан язык ассемблера. Одним из самых «больших» и подробно рассмотренных примеров, демонстрирующих «искусство программирования по Кнуту», является разработка программного обеспечения для системы управления лифтом (разд. 2.2.5 «Дважды связанные списки» в работе [1]).

Из этого примера следует, что искусством программирования Д. Кнут обладает, так как настоящее искусство не предполагает обоснования процесса создания произведения. Автор на основании технического задания, о качестве которого сказано выше, используя только словесное описание алгоритмов, на многих страницах книги приводит в окончательной форме программу на упомянутом языке низкого уровня, откомментированную не более внятно, чем было сформулировано задание.

В этом случае имеет место ситуация, не удовлетворявшая Э. Дейкстру, который говорил, что «программы часто приводятся в форме готовых изделий, почти без упоминания тех рассуждений, которые проводились в процессе разработки и служили обоснованием для окончательного вида завершенной программы» [3].

Переходя к науке программирования, которая, в отличие от искусства, должна объяснять, как создавалось произведение, отметим, что будем понимать ее существенно шире, чем Д. Грис [4], трактовавший ее только как верификацию программ. В настоящее время наука программирования включает в себя также проектирование [5], реализацию [6], документирование [7] и отладку [8].

Программное обеспечение для системы управления лифтом можно разработать с помощью различных подходов, один из которых связан с использованием конечных автоматов [9–11].

Настоящая работа призвана продемонстрировать преимущества подхода, названного «автоматное программирование с явным выделением состояний» [9, 12–14], на примере задачи управления лифтом. Кроме того, в работе приводится методика преобразования объектной программы, написанной в рамках предлагаемого подхода, в процедурную программу для выполнения ее на микроконтроллере.

Необходимо отметить, что в настоящей работе, для того чтобы не загромождать исходный код не относящимися к делу функциями, инкапсуляция данных не используется. Основная причина отсутствия инкапсуляции связана с тем, что разработанная объектно-ориентированная программа впоследствии преобразуется в процедурную программу для микроконтроллера. При этом, если бы инкапсуляция применялась, то, в результате указанного преобразования, она перестала бы выполнять свою основную задачу – сокрытие данных.

Особенности объектно-ориентированного программирования с явным выделением состояний

Автоматное программирование может использоваться в одном из двух вариантов: как процедурное программирование с явным выделением состояний [12] или как объектно-ориентированное программирование с явным выделением состояний [13].

В настоящей работе, как и в работе [13], используется второй подход, основанный на двух парадигмах: объектной и автоматной. При этом отметим, что в указанной статье автоматы реализуются как методы классов, в то время как в настоящей работе предлагается реализовывать их как классы. Это позволяет в полной мере совместить гибкость объектно-ориентированного программирования с наглядностью и ясностью автоматного подхода.

Проектирование каждого автомата состоит в создании по словесному описанию (декларации о намерениях) схемы связей, описывающей его интерфейс, и графа переходов, определяющего его поведение. По этим двум документам формально и изоморфно может быть построен модуль программы, соответствующий автомату.

Используя объектную парадигму, автоматы предлагается разрабатывать как наследники базового класса Automat. Этот класс реализует типовые функции автоматов (основные и вспомогательные). В наследниках определяются только функции, специфические для конкретных автоматов.

Перечислим основные функции автоматов, реализованные в базовом классе [14]:

- организация выполнения действий в вершинах графа переходов (для автоматов Мура), на его дугах и петлях (для автоматов Мили), а также в вершинах, на дугах и петлях (для автоматов Мура–Мили) [15];
- организация взаимодействия автоматов:
  - вызов автоматов с определенными событиями;
  - реализация вложенных автоматов;
  - обмен номерами состояний.

Отметим, что если взаимодействие по вложенности возможно только «сверху вниз» в иерархии автоматов, то остальные два способа могут осуществляться в обе стороны, как «сверху вниз», так и «снизу вверх».

Из вспомогательных функций автоматов в классе Automat реализована поддержка протоколирования. При этом возможно:

- автоматическое протоколирование:
  - при начале работы автомата в определенном состоянии с определенным событием;
  - при переходах из состояния в состояние;
  - при завершении работы автомата в определенном состоянии;
- добавление описаний входных и выходных воздействий автомата.

В классах-наследниках переопределяется ряд функций базового класса и добавляются входные воздействия (события и переменные), внутренние переменные, выходные воздействия, объекты управления, а также вложенные и вызываемые автоматы.

В настоящей работе, как отмечалось ранее, предлагаемый подход иллюстрируется примером моделирования лифта. При этом с помощью среды Microsoft Visual C++ 6 была разработана программа Lift. Эта программа размещена на сайте http://is.ifmo.ru в разделе «Статьи».

Как отмечалось ранее, программа *Lift* является объектно-ориентированной. Такую программу удобно разрабатывать на персональном компьютере и легко переносить на PC-подобные контроллеры. Однако, кроме таких контроллеров, в системах управления используются микроконтроллеры, для которых отсутствуют компиляторы с объектно-ориентированных языков. Поэтому для микроконтроллеров применяется процедурное программирование.

В настоящей работе предлагается методика преобразования ядра объектно-ориентированной программы с явным выделением состояний на языке C++ в процедурную программу с явным выделением состояний на языке C. В данном случае под ядром программы понимается ее фрагмент, в котором отсутствует интерфейсная часть и не реализованы функции входных и внутренних переменных, а также выходных воздействий. Методика иллюстрируется примером переноса ядра программы *Lift* на микроконтроллер Siemens SAB 80C515. Полученная в результате программа также размещена на сайте http://is.ifmo.ru в разделе «Статьи».

Формулировка задачи о лифте

Попытаемся из потока слов, описаний «сопрограмм», протокола работы и исходного кода программы на языке машины MIX [1] извлечь формулировку задачи.

Дано пятиэтажное здание с одним лифтом. Этаж с номером ноль – подвальный, один – цокольный, два – первый, три – второй, четыре – третий. Будем считать, что этажи пронумерованы числами от нуля до четырех.

На каждом этаже – две кнопки для вызова лифта на движение вверх и вниз. На нулевом этаже кнопка «Вниз» заблокирована, как и кнопка «Вверх» на четвертом этаже.

В кабине лифта имеется панель с пятью кнопками для перемещения на конкретный этаж. В ней также размещены два индикатора движения (вверх и вниз). Будем рассматривать их как единое устройство, формирующее одно из трех значений: *GoingUp* (лифт дви-

■ ***Таблица 1***

Имя	Значение	Комментарий
TInactive	300	Если лифт находится на каком-либо этаже без движения в течение этого времени, то он должен быть автоматически направлен на второй этаж. Для определения действий в этом случае необходимо выполнить четвертый шаг словесного описания работы системы (см. ниже)
TDoorsTimeout	76	Если после открытия дверей прошло TDoorsTimeout десятых долей секунды, то необходимо попытаться их закрыть
TDoors	20	Время открытия и закрытия дверей лифта
TWaitTimeout	40	Время, через которое система пытается закрыть двери лифта. Входящий/выходящий человек может помешать этому. Описываемая задержка требуется для того, чтобы после вхождения последнего человека в кабину двери через некоторое время закрылись
TStarting	15	Время ускорения лифта при начале движения
TUp	51	Время равномерного подъема лифта на один этаж
TUpBraking	14	Время замедления лифта перед остановкой при подъеме
TDown	61	Время равномерного спуска лифта на один этаж
TDownBraking	23	Время замедления лифта перед остановкой при спуске
TAfterRest	20	Время перед стартом лифта после выхода из режима ожидания
THuman	25	Время, требуемое человеку, чтобы войти или выйти из кабины
TWaitLimit	600	Максимальное время, которое человек согласен ждать лифт

■ ***Таблица 2***

Имя	Комментарий
Floor	Номер этажа, на котором находится лифт. Эта переменная получает новое значение при начале движения с этажа
State	Состояние движения лифта (*GoingUp, GoingDown или Neutral*)
Queue[i]	Список людей, ожидающих лифт на *i*-м этаже
Elevator	Список пассажиров, находящихся в кабине лифта
CallCar[i]	Вектор, *i*-я ячейка которого содержит единицу, если от какого-либо из пассажиров кабины поступал вызов для движения на *i*-й этаж, и ноль – в противном случае
CallUp[i]	Вектор, *i*-я ячейка которого содержит единицу, если с *i*-го этажа поступал вызов на движение вверх
CallDown[i]	Вектор, *i*-я ячейка которого содержит единицу, если с *i*-го этажа поступал вызов на движение вниз

жется вверх), *GoingDown* (лифт движется вниз) или *Neutral* (лифт находится в режиме ожидания).

Первоначально лифт находится на втором этаже в режиме ожидания (состояние *Neutral*). Ни одна кнопка вызова не нажата.

При моделировании необходимо ввести масштаб времени. Будем измерять его в десятых долях секунды. Зададим продолжительность характерных для системы действий. В табл. 1 приведены имена временных параметров, их значения и комментарии к ним.

Отметим, что время перемещения лифта из состояния покоя на один этаж вверх (вниз) с остановкой на этом этаже определяется соотношением *TStarting* + *TUp* + *TUpBraking* (*TStarting* + *TDown* + *TDownBraking*). Прохождение этажа «транзитом» происходит быстрее, так как при этом движение лифта не должно замедляться.

Если с этажа, на котором в данный момент находится лифт, поступил вызов или один из пассажиров лифта желает выйти на данном этаже, то производится открытие дверей. Это занимает *TDoors* единиц времени. По истечении *TDoorsTimeout* единиц времени после открытия дверей на этаже необходимо автоматически попытаться их закрыть. Если это не удалось, то далее лифт будет пробовать закрывать двери каждые *TWaitTimeout* единиц времени.

Если лифт выполнил все вызовы и остался стоять на этаже, то, по истечении *TInactive* единиц времени, он должен вернуться на второй этаж, так как считается, что там наиболее вероятно появление новых пассажиров.

Для реализации временных задержек в систему введены три таймера: 1) таймер бездействия C1 (для отсчета *TInactive* единиц времени); 2) таймер C2 (для остальных временных операций, не связанных с работой дверей); 3) таймер C3 (для временных операций, связанных с работой дверей).

Для моделирования описываемой системы используются переменные и структуры данных, перечисленные в табл. 2.

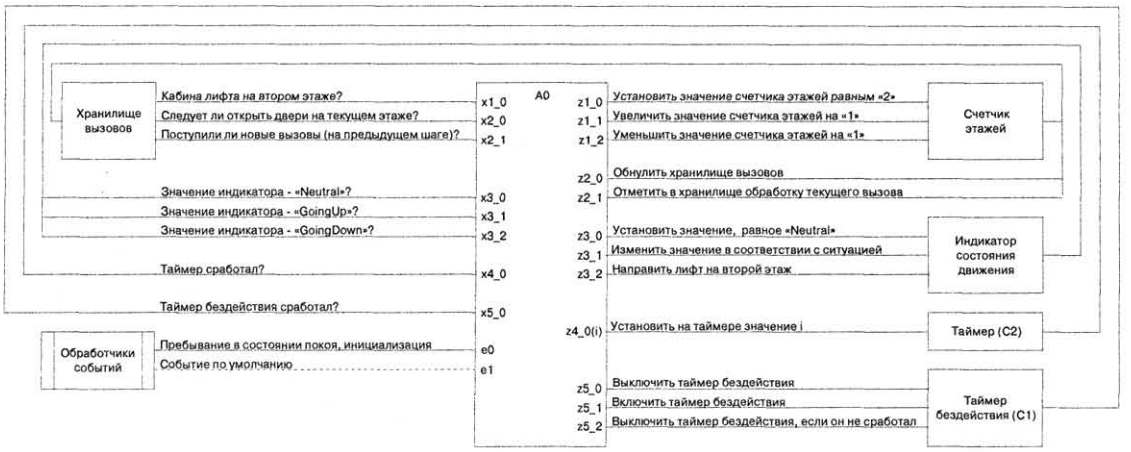
Опишем работу лифта по шагам.

0. Ожидание вызова. *Floor* = 2, *State* = *Neutral*. Если поступает вызов со второго этажа, то перейти к шагу 1. Если вызов – с другого этажа, то перейти к шагу 4.

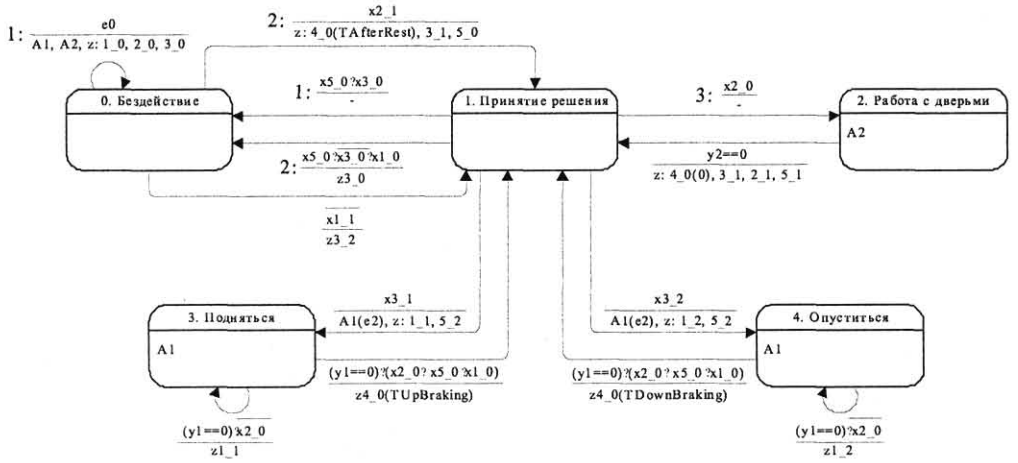
1. Открытие дверей. Сбросить таймер *C1*. Открыть двери (*TDoors* единиц времени) и перейти к шагу 2. Все пассажиры считаются дисциплинированными, и поэтому они не будут входить до полного открытия дверей.

2. Вход и выход пассажиров.

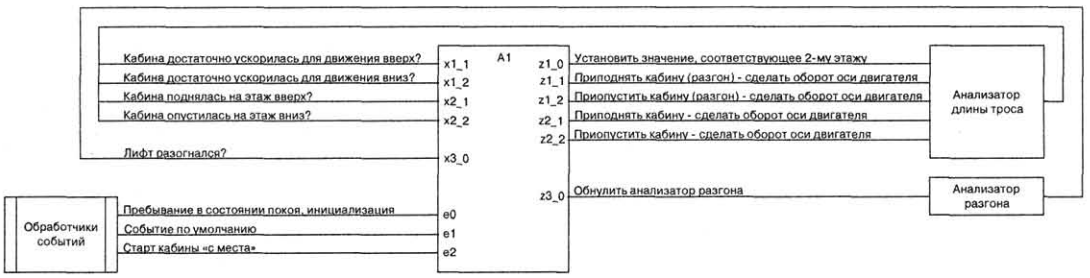
2.1. Если пройдет *TInactive* единиц времени (сработает таймер C1), то перейти к шагу 4.



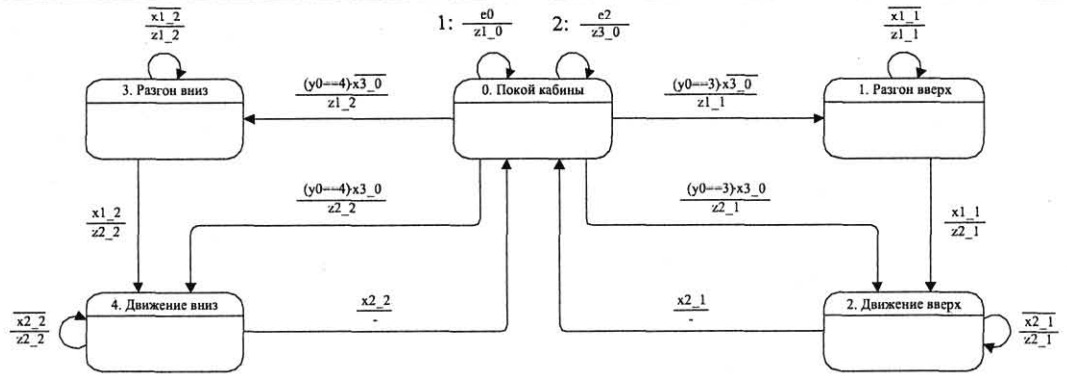
■ ***Рис. 1.*** *Схема связей автомата A0. Принятие решений*



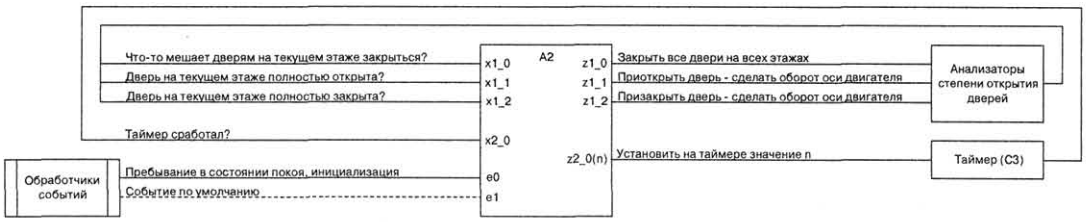
■ ***Рис. 2.*** *Граф переходов автомата A0. Принятие решений*



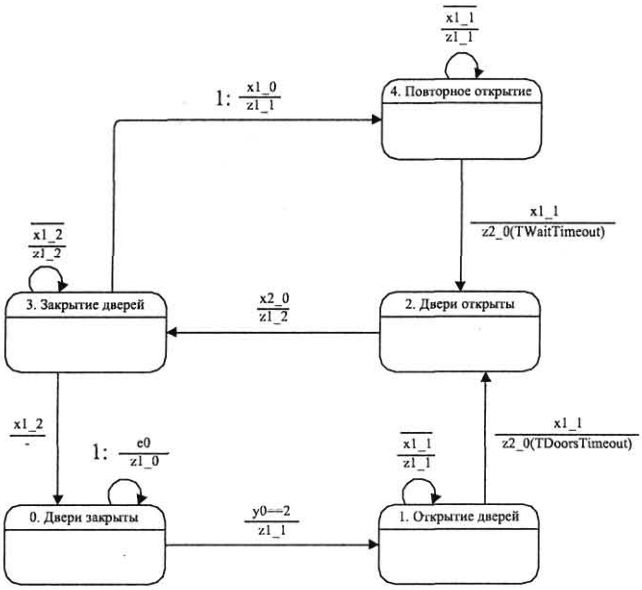
■ ***Рис. 3.*** *Схема связей автомата A1. Управление двигателем, перемещающим лифт между этажами*



■ ***Рис. 4.*** *Граф переходов автомата A1. Управление двигателем, перемещающим лифт между этажами*



■ ***Рис. 5.*** *Схема связей автомата А2. Управление пятью двигателями, открывающими/закрывающими двери на этажах*



■ ***Рис. 6.*** *Граф переходов автомата А2. Управление пятью двигателями, открывающими/закрывающими двери на этажах*

2.2. Сбросить таймер С2. Каждого пассажира кабины (элемент списка *Elevator*), который ехал до данного этажа, выпустить из лифта. Это займет *THuman* единиц времени на одного пассажира.

2.3. Пока на этаже есть люди, ждущие лифт для движения в том же направлении, в котором он двигался (их надо искать в списке *Queue*[*Floor*]), люди впускаются внутрь кабины по одному. Это займет *THuman* единиц времени на каждого пассажира. Как только человек входит в кабину лифта, он сразу же нажимает на кнопку целевого этажа (изменяет значение ячейки вектора *CallCar*[*i*]). Если вызов на этот этаж был произведен раньше, то фиксировать его не требуется.

2.4. Если по таймеру С2 прошло *TDoorsTimeout* единиц времени, то перейти к шагу 3.

3. Закрытие дверей.

3.1. Если пройдет *TInactive* единиц времени (сработает таймер *С1*), то перейти к шагу 4.

3.2. Попытаться закрыть дверь. Если не получилось – повторять попытки каждые *TWaitTimeout* единиц времени.

3.3. Когда двери закроются – выполнить присваивания: *CallUp*[*Floor*] = 0 (если *State* ≠ *GoingDown*), *CallDown*[*Floor*] = 0 (если *State* ≠ *GoingUp*) и *CallCar*[*Floor*] = 0. Присвоения удаляют информацию об обработанных вызовах. Перейти к шагу 4.

4. Принятие решения о дальнейшем движении.

4.1. Если *State* = *GoingUp* (*GoingDown*) и есть еще вызовы на движение вверх (вниз), то значение переменной *State* не изменится. Это условие означает, что существует значение *i*, большее (меньшее) значения переменной *Floor*, для которого значение из ячеек *CallCar*[*i*], *CallUp*[*i*] или *CallDown*[*i*] отлично от нуля. Перейти к подпункту 4.4.

4.2. Если вызовов на движение вверх (вниз) нет, но есть вызовы на движение вниз (вверх), то присвоить переменной *State* значение *GoingDown* (*GoingUp*). Перейти к подпункту 4.4.

4.3. Если вызовов вообще нет и при этом значение *Floor* меньше (больше) двух, то присвоить переменной *State* значение *GoingUp* (*GoingDown*). При *Floor* = 2 присвоить переменной *State* значение *Neutral*. Перейти к подпункту 4.4.

4.4. В результате, если *State* = *GoingUp*, то перейти к шагу 5, если *State* = *GoingDown* – к шагу 6, а если *State* = *Neutral* – к шагу 0.

5. Подняться на этаж.

5.1. Увеличить значение переменной *Floor* на единицу и начать движение. Это займет *TStarting* + *TUp* единиц времени.

5.2. Если есть вызов для движения на этаж *Floor*, то необходимо, чтобы лифт притормозил. Это займет *TUpBraking* единиц времени. Перейти к шагу 1.

5.3. Если нет вызовов для движения на этаж *Floor*, то повторить шаг 5.

6. Спуститься на этаж.

6.1. Уменьшить значение переменной *Floor* на единицу и начать движение. Это займет *TStarting* + *TDown* единиц времени.

6.2. Если есть вызов для движения на этаж *Floor*, то необходимо, чтобы лифт притормозил. Это займет *TDownBraking* единиц времени. Перейти к шагу 1.

6.3. Если нет вызовов для движения на этаж *Floor*, то повторить шаг 6.

Человек в рамках решаемой задачи представляет собой сущность, характеризуемую тремя атрибутами:

1) *CurFloor* – номер этажа, на котором он появляется;

2) *TgtFloor* – номер этажа, на который он хочет попасть (*CurFloor* ≠ *TgtFloor*);

3) *WaitFor* – максимальное время, которое человек согласен ждать лифт. Если по истечении этого времени он не попадет в кабину, то пойдет пешком (покинет этаж *CurFloor*). При этом его вызов остается в силе. Начальное значение атрибута *WaitFor* равно *TWaitLimit*.

Взаимодействие людей с лифтом осуществляется через списки *Queue*[*i*] и *Elevator*, а также вектора *CallUp*[*i*], *CallDown*[*i*] и *CallCar*[*i*].

Проектирование автоматов и классов

На основании словесного описания, приведенного в предыдущем разделе, можно построить схему алгоритма, которая будет весьма громоздкой. По-

этому в дальнейшем будем использовать автоматный подход.

Построим три автомата, каждый из которых определяет поведение соответствующей компоненты системы. Они обеспечивают:

принятие решений (автомат A0);

управление двигателем, перемещающим лифт между этажами (автомат A1);

управление пятью двигателями, открывающими/закрывающими двери на этажах (автомат A2).

Схема связей автомата A0 приведена на рис. 1, его граф переходов – на рис. 2, схема связей автомата A1 на рис. 3, его граф переходов – на рис. 4, и, наконец, схема связей автомата A2 – на рис. 5, его граф переходов – на рис. 6.

Все три автомата являются автоматами Мили, и поэтому каждый из них будет реализован с помощью одного оператора *switch*. Для автомата A0 характерно, что в состояние «Работа с дверьми» вложен автомат A2, а в состояния «Подняться» и «Опуститься» – автомат A1. На переходах из состояния «Принятие решения» в состояния «Подняться» и «Опуститься» осуществляется вызов автомата A1 с событием e2.

Класс A0 реализует автомат принятия решений и содержит пять объектов, управляемых им: 1) хранилище вызовов (реализуемое векторами *CallCar*[*i*], *CallUp*[*i*] и *CallDown*[*i*]); 2) счетчик этажей (*Floor*); 3) индикатор состояния движения (*State*); 4) таймер (C2); 5) таймер бездействия (C1). Кроме того, он содержит объекты классов A1 и A2, реализующих автоматы A1 и A2, соответственно.

Автоматы A1 и A2 не содержат вложенных автоматов и не вызывают другие автоматы с какими-либо событиями.

Класс A1 реализует автомат управления двигателем, перемещающим кабину лифта между этажами, и содержит этот двигатель.

Основной характеристикой положения лифта является длина части троса между кабиной и точкой перегиба троса. Будем измерять эту величину в «расстояниях между этажами». Таким образом, данная характеристика – число с плавающей точкой от нуля до четырех. Скорость перемещения лифта зависит от степени его разгона/торможения, которая определяется анализатором разгона. Анализаторы длины троса и разгона в совокупности моделируют двигатель, перемещающий кабину между этажами.

Класс *A2* реализует автомат управления пятью двигателями, открывающими/закрывающими двери на этажах, а также содержит эти двигатели и таймер (C3).

Основной параметр, на который влияет двигатель, открывающий/закрывающий двери, – степень их открытия (вещественное число от нуля до единицы). Он равен отношению числа сделанных оборотов оси двигателя к общему числу оборотов, необходимых для того, чтобы двери открылись полностью. Анализаторы степени открытия дверей моделируют соответствующие двигатели.

Кроме того, системе управления дверьми необходим таймер для определения времени закрытия дверей.

У каждого из рассмотренных автоматов имеется так называемое «событие по умолчанию», с которым он вызывается при отсутствии прочих событий.

В заключение раздела отметим, что включение объектов управления в классы автоматов не является обязательным.

Описание базового класса `Automat` и вспомогательных макроопределений

Рассмотрим базовый класс `Automat`, реализующий функциональность, описанную во введении:

```
class Automat
{
public:
   int y;              // Переменная, хранящая состояние автомата
   int y_old;          // Переменная, хранящая состояние, в котором автомат начал
                       // последнюю итерацию
   int BaseIndent;     // Минимальный базовый сдвиг для всех записей в протоколе
       bool DoLog;     // Писать ли протокол? Вот в чем вопрос!
   Automat* Nest;      // Указатель на автомат, запустивший данный
   Automat(bool DoLog=false); // Конструктор
   void Main(int e);   // Главная функция автомата

// Основные функции
protected:
   // Шаг автомата. Необходимо переопределять в наследниках
   virtual void Step(int e)=0;

   // Пост-шаг автомата, выполняемый только в случае перехода из состояния в
   // состояние (только для автоматов Мура и Мура-Мили). Необходимо переопределять
   // в наследниках
   virtual void PostStep(int e)=0;

public:
   // Выполняет шаг автомата A с событием e
   void DoNested(int e, Automat* A);

// Вспомогательные функции
public:
   // Записать в протокол строку s с отступом indent. Необходимо
   // переопределять в наследниках
   virtual void WriteLog(CString s, int indent=0)=0;

protected:
   // Записать в протокол информацию о том, что автомат запущен с событием e
   // (indent - величина отступа). Необходимо переопределять в наследниках
   virtual void WriteLogOnStart(int y,int e,int indent=0)=0;

   // Записать в протокол информацию о переходе автомата из состояния y_old в
   // состояние y (indent - величина отступа). Необходимо переопределять в
   // наследниках
   virtual void WriteLogOnStateChanged(int y,int y_old,int indent=0)=0;

   // Записать в протокол информацию о том, что автомат завершил работу в
```

```
   // состоянии y (indent - величина отступа).
Необходимо переопределять в
   // наследниках
   virtual void WriteLogOnFinish(int y,int
indent=0)=0;

// Объекты управления. Добавить в наследниках
protected:

// Вызываемые автоматы. Добавить в наследниках
protected:
// События. Добавить в наследниках
protected:

// Входные переменные. Добавить в наследниках
protected:
// Внутренние переменные. Добавить в наследниках
private:

// Выходные воздействия. Добавить в наследниках
protected:
};
```

В конце рассмотренного кода приведен состав членов класса, которые необходимо добавлять в наследники класса `Automat` для реализации конкретных автоматов.

Кратко опишем функции-члены класса `Automat`. Одной из них является функция Main – главная функция автомата, обеспечивающая выполнение действий в вершинах графа переходов, на его дугах и петлях:

```
void Automat::Main(int e)     // Главная функция
автомата
{
   y_old=y;                   // Запомнить состо-
яние перед началом итерации
   WriteLogOnStart(y,e);      // Протоколировать
факт начала итерации
   Step(e);                   // Выполнить шаг
   WriteLogOnStateChanged(y,y_old);  // Протоко-
лировать факт перехода или
                              // сохранения со-
стояния
   if (y!=y_old) PostStep(e); // Если переход был
- выполнить пост-шаг
   WriteLogOnFinish(y);       // Протоколировать
факт окончания итерации
}
```

Параметром этой функции является идентификатор события, который передается функциям `Step` и `PostStep`. При реализации автоматов Мили (Мура) единственный оператор switch необходимо разместить в переопределенной в наследнике функции `Step` (`PostStep`). Для автоматов Мура–Мили требуется переопределять обе эти функции.

Макроопределения `DECLARE_NO_STEP` и `DECLARE_NO_POSTSTEP` позволяют реализовать функции `Step` и `PostStep` как пустые.

Протоколирование поведения автомата реализуется посредством переопределения функций `WriteLog`, `WriteLogOnStart`, `WriteLogOnStateChanged` и `WriteLogOnFinish`. Последние три функции, как правило, используют первую. Функцию `WriteLog` следует вызывать дополнительно, например, из функций входных и внутренних переменных, а также функций выходных воздействий.

Макроопределения `DECLARE_NO_LOG`, `DECLARE_NO_LOG_ON_START`, `DECLARE_NO_LOG_ON_STATE_CHANGED` и `DECLARE_NO_LOG_ON_FINISH` позволяют реализовать указанные выше функции как пустые.

Все четыре функции протоколирования содержат необязательный параметр `indent` – отступ при размещении информации в протоколе. Сумма значений параметра `indent` и переменной `BaseIndent` определяет количество пробелов, добавляемых перед записью в протокол, что позволяет наглядно отразить в нем вложенность автоматов.

Для включения/выключения режима протоколирования используется булева переменная `DoLog`. Поэтому рекомендуется перечисленные выше функции протоколирования (или только функцию `WriteLog`) начинать со следующей строки:

```
if (!DoLog) return;
```

Для временного отключения протоколирования используются макроопределения `SWITCH_LOG_OFF` и `SWITCH_LOG_ON`. Необходимо, чтобы они применялись только парами, и при этом первое макроопределение должно всегда предшествовать второму. Приведем пример их использования в программе *Lift*:

```
void A0::z5_2()
{
    WriteLog("z5_2: Выключить таймер бездействия,
если он не сработал",3);
    SWITCH_LOG_OFF // Отключить протоколирование
    if (!x5_0()) z5_0();
    SWITCH_LOG_ON  // Включить протоколирование
}
```

В этом фрагменте программы временное отключение протоколирования используется для того, чтобы при вызове функции `z5_2()` в протоколе не отражались вызовы функций `x5_0()` и `z5_0()`.

Конструктор класса `Automat` инициализирует необходимые переменные:

```
Automat::Automat(bool DoLog)
{
    y=0;
    Nest=NULL;
    BaseIndent=0;
    this->DoLog=DoLog;
}
```

Используемая в тексте конструктора переменная `Nest` требуется для работы с вызываемыми автоматами. Вызов автомата выполняется с помощью функции `DoNested`, реализованной следующим образом:

```
void Automat::DoNested(int e, Automat* A)
{
    A->Nest=this;
    A->Main(e);
}
```

Например, вызов автомата `B` с событием `e` из автомата `A` должен быть осуществлен, как показано ниже:

```
DoNested(e, &B);
```

При этом автомат A, при необходимости, сможет определить состояние автомата B, обратившись к переменной B.y, а автомат B – узнать состояние автомата A, получив значение переменной Nest->y.

Разработка автоматов – потомков класса Automat

При разработке классов наследников необходимо:
1) переопределить функции Step и/или PostStep;
2) переопределить функции протоколирования WriteLog, WriteLogOnStart, WriteLogOnStateChanged и WriteLogOnFinish;
3) объявить экземпляры структур данных, соответствующие объектам управления;
4) объявить объекты, реализующие автоматы, которые вызывает данный автомат;
5) объявить целочисленные статические константы, соответствующие обрабатываемым событиям;
6) реализовать функции входных переменных, возвращающие значения для анализа;
7) реализовать функции внутренних переменных. (эти функции могут возвращать значения произвольных типов);
8) реализовать функции выходных воздействий, не возвращающие значений.

При решении задачи о лифте классы, соответствующие автоматам A0, A1 и A2, разработаны по этому плану. Приведем в качестве примера объявление класса A0.

```
////////////////////////////////////////////////
///////////////
// Автомат, реализующий функциональность кабины
лифта

class A0 : public Automat
{
protected:
   virtual void Step(int e); // Шаг автомата
   DECLARE_NO_POSTSTEP;      // Пост-шаг не нужен

public:
   virtual void WriteLog(CString s,int indent=0);
protected:
   virtual void WriteLogOnStart(int y,int e,int
indent=0);
   virtual void WriteLogOnStateChanged(int y,int
y_old,int indent=0);
   virtual void WriteLogOnFinish(int y,int
indent=0);

// Вспомогательные определения
public:
   // Значения индикатора движения
   typedef enum StateValuesType {NEUTRAL, GOINGUP,
GOINGDOWN} StateValues;

   // Тип структуры, реализующей хранилище вызо-
вов
   typedef struct CallsStoreType
   {
      int up[5],down[5],car[5];
      bool newcalls;
   } CallsStore;

// Устройства
public:
   unsigned floor;       // Счётчик этажей
   unsigned timer;       // Таймер
   int inactivitytimer; // Таймер бездействия
   CallsStore calls;     // Хранилище вызовов
   StateValues state;    // Индикатор движения

// Автоматы, к которым будет обращаться данный
public:
   A1 aA1; // Автомат A1
   A2 aA2; // Автомат A2

// События
public:
   const static int e0;       // Пребывание в со-
стоянии покоя, инициализация
   const static int e1;       // Событие по умол-
чанию

// Входные переменные
protected:
   // Переменные, получаемые от счётчика этажей
   int x1_0(); // Кабина лифта находится на вто-
ром (базовом) этаже?

   // Переменные, получаемые от хранилища вызовов
   int x2_0();         // Нужно ли открыть двери
на текущем этаже?
   int x2_1();         // Поступили ли новые вы-
зовы (на предыдущем шаге)?

   // Переменные, получаемые от индикатора движе-
ния
   int x3_0();         // Значение индикатора -
«Neutral»
   int x3_1();         // Значение индикатора -
«GoingUp»
   int x3_2();         // Значение индикатора -
«GoingDown»

   // Переменные, получаемые от таймера
   int x4_0();         // Таймер сработал?

   // Переменные, получаемые от таймера бездей-
ствия
   int x5_0();         // Таймер бездействия сра-
ботал?

// Выходные воздействия
protected:
   // Воздействия на счётчик
   void z1_0();        // Установить значение
счётчика этажей равное двум
   void z1_1();        // Увеличить значение счёт-
чика этажей на один
   void z1_2();        // Уменьшить значение счёт-
чика этажей на один

   // Воздействия на хранилище вызовов
   void z2_0();        // Обнулить хранилище вы-
зовов
   void z2_1();        // Отметить в хранилище об-
работку текущего вызова

   // Воздействия на индикатор движения
   void z3_0();        // Установить значение
«Neutral»
```

```
   void z3_1();        // Изменить значение, в со-
ответствии с текущей ситуацией
   void z3_2();        // Направить лифт на вто-
рой (базовый) этаж

   // Воздействия на таймер
   void z4_0(unsigned n);     // Установить зна-
чение таймера равное n

   // Воздействия на таймер бездействия
   void z5_0();        // Выключить таймер бездей-
ствия
   void z5_1();        // Включить таймер бездей-
ствия
   void z5_2();        // Выключить таймер бездей-
ствия, если он не сработал

// Внутренние переменные
private:
   // Переменные, связанные с индикатором движе-
ния
   StateValues i3_0();        // Вычислить состо-
яние индикатора движения, адекватное
                              // текущей ситуации
};
```

Большую часть исходного кода программы *Lift* занимает реализация интерфейса. Отметим, что вызовы главной функции автомата A0 из интерфейсной части программы происходят только в двух случаях: при повторной инициализации автомата и при выполнении шага. Они инициируются нажатием пользователем на кнопки *Reset*, *Step*, *Pass* или *Auto*. Другие автоматы из интерфейсной части не вызываются. Подробно интерфейс программы описан в следующем разделе.

Интерфейс программы

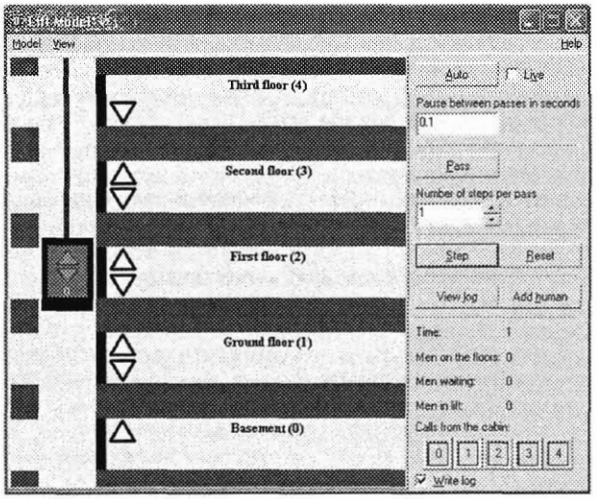
Опишем внешний вид окна программы (рис. 7).

В его левой части находится область визуализации системы, предназначенная для отображения лифта, этажей, кнопок вызова на этажах и людей. Справа расположены средства управления моделью, а сверху – главное меню. Перечислим упомянутые средства управления:

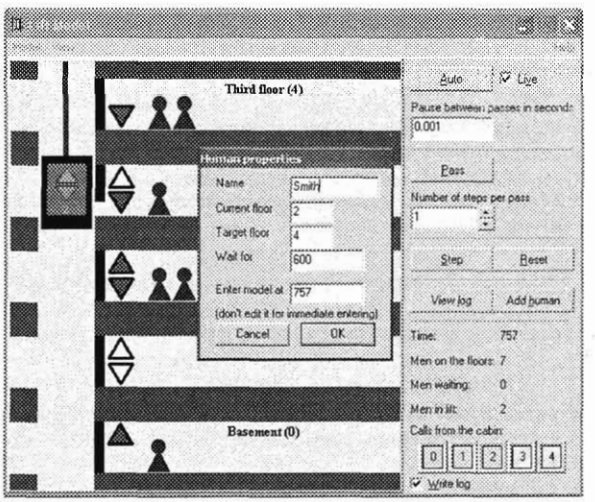
- кнопка *Auto* предназначена для перехода в режим автоматического выполнения итераций и выхода из него. В этом режиме происходит постоянное выполнение пассов (наборов итераций) одного за другим. Паузу между ними можно настроить;
- флажок *Live* предназначен для включения/выключения режима случайного автоматического добавления людей на этажи;
- поле ввода *Pause between passes in seconds* определяет паузу (в секундах) между пассами при работе системы в автоматическом режиме;
- кнопка *Pass* позволяет выполнить набор итераций;
- поле ввода *Number of steps per pass* определяет количество итераций в пассе;
- кнопка *Step* позволяет выполнить итерацию;
- кнопка *Reset* обеспечивает выполнение сброса системы (перевод каждого автомата и их объектов управления в начальные состояния). В результате лифт пуст, находится на втором этаже, все двери закрыты, людей на этажах нет и вызовов нет;
- кнопка *View log* позволяет открыть/закрыть окно отображения протокола;
- кнопка *Add human* позволяет добавить человека на этаж. При этом появляется диалоговое окно для задания его параметров, описанных ниже;
- поле *Time* отображает номер текущей итерации;
- поле *Men on the floors* отображает число людей на этажах;
- поле *Men waiting* отображает число людей в очереди ожидания, размещение которых на этажах было запланировано на будущее;
- поле *Men in lift* отображает число людей в лифте. Это число также отображается на кабине лифта;
- панель *Calls from the cabin* с пятью кнопками представляет собой пульт управления в кабине лифта. При этом нажатые кнопки подсвечиваются;
- флажок *Write log* предназначен для включения/выключения режима протоколирования.

На рис. 8 изображено окно программы после появления людей на этажах. На нем также показано окно настройки свойств добавляемого человека, открывающееся при нажатии кнопки *Add human*, которое позволяет изменять следующие параметры «личности»:

- *Name* – имя человека. Имя вида «*Human#...*» генерируется автоматически, но при необходимости может быть изменено вручную. Этот параметр используется только при протоколировании;
- *Current floor* – этаж, на котором следует разместить человека;



■ ***Рис. 7.*** *Окно программы Lift*



■ ***Рис. 8.*** *Окно программы Lift после добавления людей*

• *Target floor* – этаж, на который должен попасть человек;

• *Wait for* – время ожидания лифта (в итерациях), по истечении которого человек покидает этаж, не дождавшись лифта;

• *Enter model at* – итерация, на которой человека следует разместить на этаже. Значение по умолчанию – текущая итерация. Если указанное значение не редактировать, то человек будет размещен незамедлительно.

Нажатие правой кнопки «мыши» на изображении человека приводит к отображению окна просмотра его параметров.

Когда человек появляется на этаже, он «нажимает» на кнопку вызова для движения вверх или вниз, а при входе в кабину он «нажимает» на кнопку на пульте управления.

Отметим, что пользователь программы может нажимать кнопки вызовов с этажей и кнопки пульта управления с помощью «мыши».

Обратим внимание, что нажатие на кнопку отменить невозможно и вызов рано или поздно будет обработан. Так, если человек появился на этаже, вызвал лифт и ушел, не дождавшись его прибытия, то лифт все равно приедет на этаж и двери откроются.

На кабине лифта, в виде двух стрелок, изображен индикатор направления движения лифта.

В случае, если на этаже пытаются разместиться более десяти человек, то изображаются только первые десять.

Как отмечалось выше, в системе имеются средства для ведения и просмотра протокола работы модели. Окно просмотра изображено на рис. 9.

В протоколе итерации отделены друг от друга специальными скобками. Вложенность автоматов отражается отступами. Информация о входных и внутренних переменных, а также выходных воздействиях добавляется с еще большими отступами.

Строка, начинающаяся с двух звездочек, содержит информацию, являющуюся внешней для системы управления лифтом, и может представлять собой одно из следующих сообщений:

«Пассажир ... с этажа ... решил попасть на этаж ... (готов ждать ... единиц времени)»;

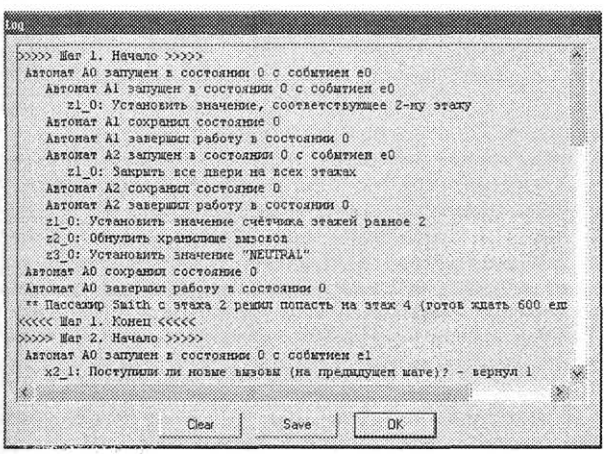
«Пассажир ... вошел в лифт на этаже ...»;

«Пассажир ... приехал на этаж ...»;

«Пассажир ... решил пойти пешком».

Информация о нажатии кнопок вызова пользователем фиксируется в протоколе с помощью сообщений:

«Поступил вызов на движение (вверх/вниз) с ... этажа»;



■ *Рис. 9. Окно просмотра протокола*

«Из кабины лифта поступил вызов на движение на ... этаж».

Опишем назначение кнопок в окне протоколирования. Кнопка *Clear* предназначена для очистки протокола, кнопка *Save* позволяет записать протокол в текстовый файл, а кнопка *OK* – закрыть окно.

Завершим описание интерфейса рассмотрением главного меню. В нем пункт *Reset* подменю *Model* повторяет функциональность кнопки *Reset*, а пункт *Exit* того же подменю служит для выхода из программы.

Подменю *View* содержит три пункта:

1) *Log window* аналогичен кнопке *View log*, описанной выше;

2) *Timeouts tuning window* открывает окно, позволяющее настраивать временные параметры, описанные в табл. 1 (кроме параметра *TWaitLimit*). По умолчанию эти параметры настроены, как в работе [1];

3) *Add human window* аналогичен кнопке *Add human*, описанной выше.

Подменю *Help* содержит единственный пункт *About*, отображающий информацию о программе.

На этом завершается описание созданной модели лифта, функционирующей на персональном компьютере. Ее работоспособность и соответствие техническому заданию (разд. 2) подтверждаются практическим использованием.

Переход от объектного варианта программы к процедурному

При переходе от модели разработанной выше программы к реальному управлению лифтом на базе PC-подобного контроллера достаточно создать наследники классов, реализующих автоматы модели, переопределив лишь виртуальные функции входных и внутренних переменных, а также выходных воздействий. Таким образом, реализацию реальной системы на указанном вычислительном устройстве следует наследовать от модели.

В случае построения системы на микроконтроллерах такой подход неприменим, так как они обычно программируются процедурно. Изложим методику, позволяющую перенести программу, написанную в рамках предлагаемого подхода на языке C++, на язык C, и проиллюстрируем ее примером переноса ядра программы *Lift* на микроконтроллер Siemens SAB 80C515. При этом в качестве среды разработки программного обеспечения для микроконтроллеров используется продукт Keil µVision 2.

Опишем эту методику.

1. Создать каталог, в котором будет размещен переносимый проект (в рассматриваемом примере он назван *Hard*). Скопировать в каталог файлы, реализующие автоматы (A0.h, A0.cpp, A1.h, A1.cpp, A2.h и A2.cpp). Изменить расширение файлов с «cpp» на «c». Создать файл Common.h и файл проекта µVision (Lift.uv2) для требуемого микроконтроллера, добавив в последний все семь упомянутых файлов.

2. Из каждого файла с исходным кодом, имеющим расширение «c», удалить директивы `#include`, кроме тех, которые включают одноименные заголовочные файлы или заголовочные файлы стандартных библиотек. Во все файлы с исходным кодом следует добавить директиву «`#include "Common.h"`», а в файл *Common.h* скопировать необходимые общие определения макросов, структур данных из файла *stdafx.h* и т. п.

3. Преобразовать определения классов в заголовочных файлах. Для этого удаляются ключевые слова `class`, а методы преобразуются в функции с именами вида

«<имя класса>_<имя функции>». Например, методы Step для трех рассматриваемых автоматов будут преобразованы в функции A0_Step, A1_Step и A2_Step. Кроме того, необходимо удалить из определений макросы «DECLARE_NO_...» и удалить или преобразовать функции протоколирования, перенаправив их вывод.

4. События, устройства и другие члены классов преобразуются по аналогии с методами, как указано в п. 3. В результате они превратятся в константы и статические переменные с определенными префиксами в названии. При условии сохранения уникальности имен, префиксы можно опустить. Таким образом, для того, чтобы обратиться к некой сущности, бывшей до преобразования членом класса, следует писать имя класса, за ним – подчеркивание (вместо точки) и имя члена класса. Кроме того, для автомата Ai необходимо добавить в заголовочный файл переменную «static int Ai_y», в которой будет храниться номер состояния автомата. Объявления объектов, реализующих вложенные автоматы, необходимо заменить на объявления их главных функций, как внешних (extern).

5. В файлах с исходным кодом необходимо заменить подстроку «::» на символ «_». В результате функции-члены класса получат требуемые имена. Внутри файла, реализующего автомат Ai, следует добавить префикс «Ai_» для обращений к переменной состояния, событиям, функциям входных и внутренних переменных, а также функциям выходных воздействий. Это почти всегда, можно сделать, выполнив в текстовом редакторе замену подстроки «y» на подстроку «Ai_y» (в режиме «только слово целиком»), а также произведя следующие замены: «e» на «Ai_e», «x» на «Ai_x», «t» на «Ai_t», а «z» на «Ai_z».

6. Обращение к переменной состояния головного автомата, которое в объектном случае осуществляется посредством переменной Nest, следует преобразовать в обращение к внешней переменной, объявленной в заголовочном файле. При вызове вложенного автомата, обращение к функции-члену «DoNested(e, &Ai)» следует заменить на вызов функции «Ai_Step(e)». В общем случае, при необходимости обратиться к устройству, событию, состоянию или главной функции другого автомата, они должны быть объявлены с помощью директивы extern. Поэтому в рассматриваемом примере в файле A0.h должны присутствовать объявления следующих внешних переменных:

```
extern const int A1_e2;
extern int A1_y;
extern int A2_y;
```

7. Необходимо привести все конструкции в соответствие с особенностями микроконтроллера. Например, для рассматриваемого микроконтроллера не предусмотрен тип данных bool. Поэтому в файле Common.h должны появиться следующие строки:

```
typedef int bool;
static const int true=1;
static const int false=0;
```

На этом завершается преобразование ядра объектно-ориентированной программы в процедурную. Одним из важнейших свойств изложенной методики является то, что все указанные выше замены и преобразования могут быть выполнены автоматически.

Для окончательного построения программы необходимо изменить или переписать функции входных и внутренних переменных, а также функции выходных воздействий, в которых осуществляются такие операции, как, например, взаимодействие с реальными устройствами, обращения к портам ввода-вывода и т. п.

Выводы

Отметим, что предложенный подход к построению системы управления лифтом обеспечивает создание хорошо документированного и легко модифицируемого проекта.

Методика преобразования объектно-ориентированной программы в процедурную позволяет, отладив первую из них, достаточно легко перенести ее в управляющее устройство, например, на базе микроконтроллера. На сайте http://is.ifmo.ru в разделе «Статьи» размещена программа *Lift*, а также ядро программного обеспечения системы управления лифтом для микроконтроллера Siemens SAB 80C515, полученное из программы *Lift* с помощью описанной выше методики переноса.

Кроме работы [1], использованной авторами в качестве прототипа, известны работы [16, 17], в которых также рассмотрена разработка программного обеспечения для задачи управления лифтом.

Как и в настоящей статье, в них используются объектно-ориентированное проектирование и программирование. Однако в работе [16] лифт проектируется для трехэтажного здания, а в [17] – для двухэтажного. В настоящей работе, как и в прототипе, моделируется лифт для пятиэтажного здания, что является более сложным.

Кроме того, в [16, 17] на этаже в каждый момент времени может находиться не более одного человека, что значительно упрощает логику поведения системы. В описанном проекте число людей на этажах практически не ограничено.

К недостаткам работы [16] относится также то, что состояния автомата «Управление лифтом» выделяются на основании нескольких не связанных между собой характеристик (наличие пассажиров, степень открытия дверей и направление движения лифта). Это привело к построению весьма сложного автомата, содержащего одиннадцать состояний, взаимодействующего с рядом других автоматов. В настоящей работе выделяются только три автомата (по пять состояний), причем каждый из них отвечает за определенную составляющую системы.

Еще один недостаток работы [16] состоит в том, что при реализации автоматов объектный подход не используется в должной мере. При этом процедурная реализация каждого автомата «обертывается» в метод одного и того же класса.

Недостатком работы [17] является то, что в ней UML-проектирование и программирование на C++ не связаны формально.

Предлагаемый подход лишен указанных «минусов», за счет совместного использования преимуществ объектного и автоматного подходов.

В заключение отметим, что в настоящее время все шире применяется технология объектно-ориентированного проектирования, названная Rational Unified Process [18]. Существуют и другие технологии, например, изложенные в [17, 19], а также в настоящей статье.

Авторы надеются, что, ознакомившись с предлагаемым подходом, читатели согласятся со словами Б. Гейтса «за последние двадцать лет мир изменился», по крайней мере, в области искусства программирования лифта.

Д. Кнут тоже не «стоит на месте». Он модернизировал модель MIX, разработав новую архитектуру MMIX [20]. Однако эти усовершенствования не коснулись области проектирования программ.

Литература

1. **Кнут Д.** Искусство программирования: В 3-х т. – Т. 1: Основные алгоритмы. – М.: Вильямс, 2001. – 712 с.
2. **Корн Г., Корн Т.** Справочник по математике для научных работников и инженеров. – М.: Наука, 1978. – 820 с.
3. **Дейкстра Э.** Дисциплина программирования. – М.: Мир, 1979. – 239 с.
4. **Грис Д.** Наука программирования. – М.: Мир, 1984. – 416 с.
5. **Буч Г.** Объектно-ориентированный анализ и проектирование. – М.: Бином, СПб.: Невский диалект, 1998. – 558 с.
6. **Страуструп Б.** Язык программирования C++. – М.: Бином, СПб.: Невский диалект, 2001. – 1098 с.
7. **Шалыто А. А.** Новая инициатива в программировании. Движение за открытую проектную документацию // Мир ПК. – 2003. – № 9. – С. 52–56. (http://is.ifmo.ru, раздел «Статьи»).
8. **Тэллес М., Хсих Ю.** Наука отладки. М.: Кудиц-образ, 2003. – 556 с.
9. **Шалыто А. А.** Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. – 628 с.
10. **Любченко В. С.** Конечно-автоматная технология программирования // Труды международн. научно-метод. конф. «Телематика'2001». СПб.: Изд-во СПбГИТМО(ТУ), 2001. – С. 127–128.
11. **Сацкий С.** Дизайн шаблона конечного автомата на C++ // RSDN Magazine. 2003. № 1. – С. 20–24.
12. **Шалыто А. А., Туккель Н. И.** Программирование с явным выделением состояний // Мир ПК. 2001. № 8. – С. 116–121; № 9. – С. 132–138. (http://is.ifmo.ru, раздел «Статьи»).
13. **Шалыто А. А., Туккель Н. И.** Танки и автоматы // BYTE/Россия. 2003. – № 2. – С. 69–73. (http://is.ifmo.ru, раздел «Статьи»).
14. **Шалыто А. А.** Технология автоматного программирования // Мир ПК. 2003. – № 10. – С. 74–78. (http://is.ifmo.ru, раздел «Статьи»).
15. **Брауэр В.** Введение в теорию конечных автоматов. – М.: Радио и связь, 1987. – 392 с.
16. **Наумов А. С., Шалыто А. А.** Система управления лифтом. Проектная документация. http://is.ifmo.ru, раздел «Проекты». – 51 с.
17. **Дейтел Х. М., Дейтел П. Дж.** Как программировать на C++/ 3-е изд. – М.: Бином, 2003. – 1152 с.
18. **Крачтен Ф.** Введение в Rational Unified Process. – М.: Вильямс, 2002. – 198 с.
19. **Шопырин Д. Г., Шалыто А. А.** Объектно-ориентированный подход к автоматному программированию. - http://is.ifmo.ru, раздел «Проекты». – 57 с.
20. **Еремин Е. А.** MMIX – учебный RISC-процессор нового тысячелетия от Дональда Кнута // Информатика. – 2002. – № 40. – С. 18–27.

УДК 531.383-1:537.2

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ВЕНТИЛЬНЫМ ИНДУКТОРНО-РЕАКТИВНЫМ ДВИГАТЕЛЕМ

Ю. А. Голландцев,
канд. техн. наук, доцент
ГНЦ РФ ЦНИИ «Электроприбор» (Санкт-Петербург)

Рассматриваются особенности построения и реализации аппаратной и программной части микропроцессорной системы управления вентильным индукторно-реактивным двигателем.

Hardware and soltware design and implementation for the microprocessor control system for a switched reluctance motors are considered.

Введение

Вентильный индукторно-реактивный двигатель (ВИРД), получивший в англоязычной технической литературе наименование Switched Reluctance Motor (*SRM*), привлекает внимание разработчиков систем автоматического управления простым конструктивным исполнением – прямые зубцы на статоре, безобмоточный зубчатый ротор, число зубцов которого не равно числу зубцов статора, катушечные обмотки, обеспечивающие высокую технологичность конструкции двигателя. Конкурентоспособность ВИРД достигается за счет выбора повышенных электромагнитных нагрузок. Индукция в зазоре может составлять величину около 2 Тл, плотность тока в обмотке статора в режиме пуска имеет значения 10–30 А/мм^2. Простые схемы вентильных коммутаторов, принципиально не имеющие режима короткого замыкания источника питания, обеспечивают формирование однополярных дискретных напряжений на обмотке статора. Микропроцессорное формирование токов в фазах, имеющих несинусоидальный характер, позволяет получить желаемые характеристики привода, построенного на основе ВИРД [1, 2, 3].

Структурная схема микропроцессорного управления электромеханической системой приведена на рис. 1. Основу системы составляет четырехфазный вентильный индукторно-реактивный двигатель. Вентильный коммутатор построен по схеме асимметричного моста. Управление транзисторами осуществляется с помощью драйверов HCPL – 316 фирмы «Agilent Technologies». В позиционной следящей системе используется два датчика угла. Основной датчик угла ВТ–5, работающий в режиме фазовращателя, измеряет угол при неограниченном повороте ротора с точностью на уровне угловой минуты. Дополнительный оптический датчик угла работает в ограниченном диапазоне измерения (около 5°), позволяя на порядок повысить точность измерения угла в позиционной следящей системе до нескольких секунд.

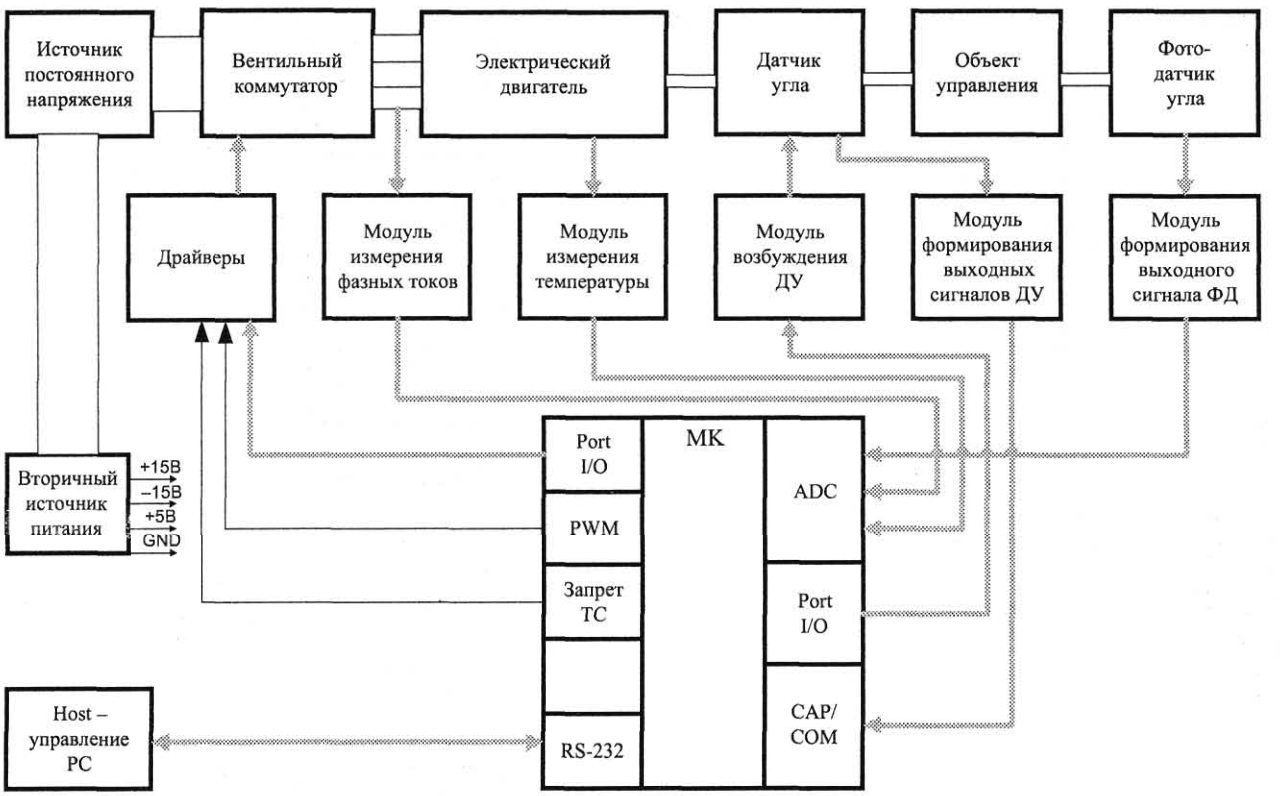
Для формирования динамических характеристик системы используются датчики тока, измеряющие токи в фазах обмотки двигателя. Сумма токов, протекающих в фазах обмотках, является модулем тока статора. Контроль теплового состояния изоляции обмотки двигателя осуществляется датчиком AD22100 фирмы «Analog Devices».

Микропроцессорная система управления ВИРД реализована на микроконтроллере фирмы «SIEMENS» SABC-167, имеющем в своем составе достаточный набор программируемых периферийных модулей, а также средства программирования и отладки.

Программное обеспечение микропроцессорной системы управления ВИРД строится по модульному принципу [4]. За каждым модулем закреплено выполнение определенных управляющих или вычислительных функций. В состав программного обеспечения входят программный пакет Keil C-167, программа LabVIEW, объектная программа работы МПСУ ВИРД.

Программный пакет Keil C-167 содержит компилятор языка «С» для микроконтроллера С-167, текстовый редактор, отладчик, загрузчик. Для настройки и контроля режимов работы микропроцессорной системы управления ВИРД в режиме реального времени используется аппаратно-программный комплекс виртуальных приборов и инструментов LabVIEW фирмы «National Instruments». Программный продукт LabVIEW как средство прикладного программирования по своей логической структуре близок к конструкции языка «С». Однако для создания программ приема, обработки и представления данных используется графическое программирование в виде блок-схем, что соответствует объектно-ориентированным языкам программирования.

Укрупненная структурная схема алгоритма работы МПСУ макета ВИРД приведена на рис. 2. Программа работы МПСУ макета ВИРД занимает в памяти 12 кбайт. Для программирования доступно 32 кбайт, при снятии ограничений на тип адресации – 65 кбайт. Частота синхронизации работы МПСУ макета ВИРД выбрана 2,44 кГц, период синхронизации – 0,41 мс. Время выполнения одного прохода по базовой части программы составляет 0,15 мс без учета затрат на



■ ***Рис. 1.*** *Структурная схема системы управления ВИРД*

выполнение позиционного регулятора, с учетом последнего – 0,32 мс.

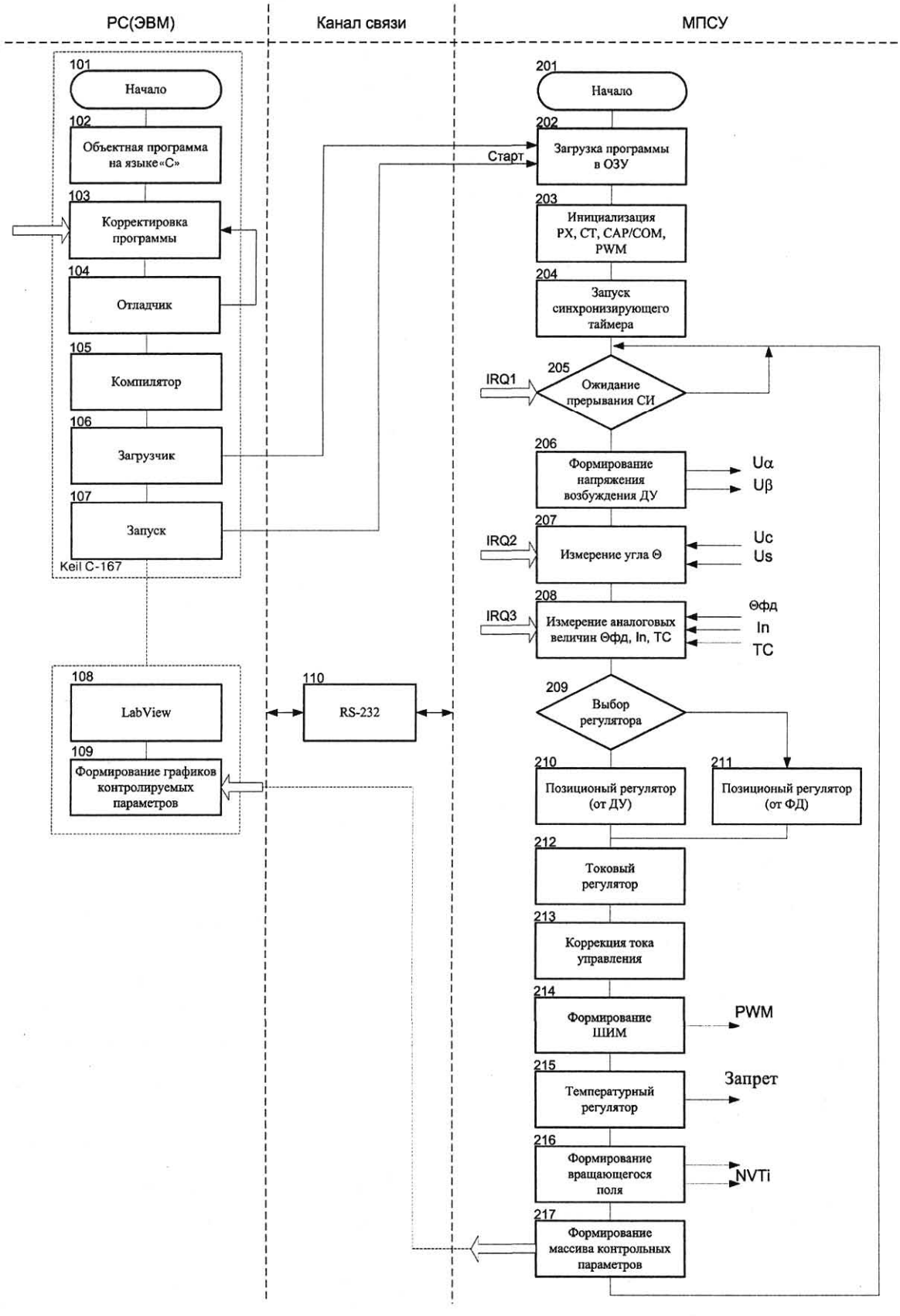
После включения питания РС (ЭВМ) и МПСУ предоставляется возможность для составления объектной программы работы МПСУ ВИРД на языке высокого уровня «С». В рабочем режиме выбирается один из возможных режимов работы МПСУ ВИРД: непосредственное вращение ротора в заданном направлении, гармоническое колебание, позиционирование в фиксированном угловом положении. Позиционирование в точке возможно при использовании либо грубого датчика (ВТ-5), либо точного оптического датчика. Требуемые значения параметров формируются в блоке имитатора сигналов.

Далее последовательно выполняется корректировка, отладка и компиляция программы в машинные коды микроконтроллера С-167. С помощью подпрограммы «ЗАГРУЗЧИК» происходит запись компилированной объектной программы в ОЗУ микроконтроллера. Одновременно можно настроить работу программы LabVIEW, сформировать панель представления результатов, определить вид и масштабы контролируемых данных. Последовательность описанных действий выполняется в блоках 101–110, 201, 202 алгоритма работы МПСУ ВИРД.

После выдачи команды «ЗАПУСК» микропроцессорная программа переходит в исходный адрес программы, по которому начинается инициализация режимов работы программируемых периферийных модулей микроконтроллера. К последним относятся: интерфейс RS-232, CAP/COM (модуль захвата/сравнения) синхронизирующего таймера Т0, CAP/COM таймера Т1, Fast PWM (быстрый ШИМ), АЦП, настройка линий ввода/вывода на требуемый режим работы, описание переменных и присвоение им начальных значений (блок 203).

В блоке 204 осуществляется запуск синхронизирующего таймера Т0, выставляются разрешения на все прерывания системы. Прерывание IRQ1 от Т0 имеет самый высокий приоритет. Далее система переводится в режим ожидания. Ожидание прерывания синхронизирующего таймера реализовано с помощью бесконечного цикла, в теле цикла осуществляется только одна операция: выдача текущих значений по протоколу RS-232. С поступлением импульса синхронизирующего таймера начинает выполняться основной модуль программы, при этом выставляется запрет на прерывание от синхронизирующего таймера, чтобы не допустить преждевременного завершения основного тела программы.

Датчик угла ВТ-5 используется в режиме фазовращателя (резольвера), для реализации которого формируется двухфазное напряжение возбуждения частотой, равной $f_{ДУ} = f_S / 4$. Измерение текущего угла поворота ротора осуществляется в модуле CAP/COM. При поступлении на линию порта P2.14 переднего фронта сигнала с синусной обмотки ДУ блоком CAP/COM вырабатывается событие-прерывание CC14IE. Обработка косинусного сигнала ДУ происходит аналогичным образом. По прерыванию CC14IE запускается обработчик прерывания, в котором происходит определение фазового сдвига синусной и косинусной обмоток ДУ. Далее формируется текущий угол и вычисляется обратная разность по углу. Описанные действия выполняются в блоках 206 и 207.



■ ***Рис. 2.*** *Алгоритм работы МПСУ ВИРД*

В блоке 208 производится последовательное сканирование трех входов мультиплексора внутреннего АЦП, на которые подаются аналоговые сигналы: оптического датчика угла, датчика потребляемого тока и датчика температуры. Цифровые коды измеренных величин используются в соответствующих регуляторах.

В МПСУ ВИРД предусмотрено два режима позиционирования в зависимости от выбранного датчика обратной связи: грубого канала неограниченного угла поворота – от ВТ-5 и точного канала ограниченного угла поворота – от оптического датчика. Структура позиционных регуляторов реализована в виде цифрового рекурсивного фильтра. Регуляторы различаются между собой численными значениями коэффициентов и параметрами ограничений на изменение вычисленных управляющих значений. Модуль управляющего сигнала используется как уставка в токовом регуляторе. Признак знака управляющего сигнала поступает в блок формирования вращающегося поля двигателя и определяет порядок переключения транзисторов вентильного коммутатора. Структура и коэффициенты позиционного регулятора выбираются при начальном запуске программы. Указанные процедуры реализуются в блоках 209, 210 и 211.

В блоке 212 вычисляется управляющее воздействие токового регулятора, выполненного по структуре ПИ-регулятора. Коррекция вычисленного управляющего воздействии осуществляется в соответствии с предварительно подобранным аппроксимирующим выражением в блоке 213. Скорректированное значение записывается в счетчик модуля ШИМ, который работает в режиме перезапуска. Сформированный импульс регулируемой длительности поступает на вход разрешения вентильного коммутатора.

Температурный релейный регулятор вырабатывает сигнал запрета, который поступает на вход разрешения вентильного коммутатора.

В блоке 216 осуществляется формирование вращающего поля. Вычисленные сигналы на открытие транзисторов, объединенные по «И» с сигналами PWM и ЗАПРЕТ, поступают на вход драйверов управления транзисторами вентильного коммутатора.

Формирование массива контролируемых параметров осуществляется в блоке 217. Считывание и передача текущих значений сформированных параметров происходит по последовательному интерфейсу RS-232 с частотой, определяемой PC.

После выполнения основного модуля программы выдается разрешение на прерывание синхронизирующего таймера, и программа переходит в режим ожидания следующего синхроимпульса.

Работоспособность микропроцессорной системы управления и эффективность предложенных вычислительных алгоритмов подтверждают экспериментальные графики работы системы в режимах позиционирования и гармонических колебаний.

Литература

1. **Miller J. E.** Switched reluctance motors and their control. – Oxford: Magna physics publishing and Clarendon press, 1993. – 205 с.
2. **Голландцев Ю. А., Гутнер И. Е.** Вентильный индукторно-реактивный двигатель // Изв. вузов. Приборостроение. – 2002. – Т. 45. – № 8. – с.12–18
3. **Дискретный** электропривод с шаговыми двигателями / Под общ. ред. М. Г. Чиликина. – М.: Энергия, 1971. – 428 с.
4. **Козаченко В. Ф.** Микроконтроллеры: Руководство по применению 16-разрядных микроконтроллеров INTELMCS – 196/206 во встроенных системах управления. – М.: Изд. Эком, 1997. – 350 с.

УДК 681.391.1

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ БУЛЕВЫХ ФУНКЦИЙ ДЛЯ ОРГАНИЗАЦИИ КРИПТОГРАФИЧЕСКИХ ПРЕОБРАЗОВАНИЙ ПОТОКОВОЙ ИНФОРМАЦИИ

А. Б. Бубликов,
ассистент
И. Л. Ерош,
д-р техн. наук, профессор
М. Б. Сергеев,
д-р техн. наук, профессор
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Рассмотрены характеристики булевых преобразований при использовании для защиты потоковой информации. Приведены сравнительные результаты исследования программной и аппаратной реализации, отмечены преимущества и особенности рассматриваемых преобразований.

Characteristics of the boolean transformations for protection stream information are considered. Comparative results of research of program and hardware realization are presented. Advantages and features of considered transformations are emphasized.

Введение

Целью данной работы является исследование возможности использования алгоритмов криптографических преобразований на основе булевых функций в реальных системах передачи потоковой информации. Долгое время стандартом в этой области являлся алгоритм RC4, однако слабая (на сегодняшний день) стойкость этого алгоритма к атакам и достаточно ресурсоемкая аппаратная реализация (алгоритм RC4 изначально разрабатывался с оптимизацией для использования программных реализаций на базе персональных компьютеров) заставляют искать новые алгоритмы для таких применений. Поэтому в качестве объекта исследования было выбрано семейство алгоритмов криптографических преобразований на основе булевых функций.

Основные принципы построения алгоритмов таких криптографических преобразований предложены в работе [1] на примере систем распределенного контроля и управления. Обсуждая вопрос простоты преобразований и возможных применений, автор ориентировался на результаты теоретических расчетов. Однако возникшая практическая задача защиты потоковой видеоинформации [2] с интенсивностью передачи от 2 Мбит/с и выше привела к необходимости оценки реальных параметров программной и, учитывая особенности таких систем, аппаратной реализаций.

Оценка программной реализации

Для оценки эффективности предложенного алгоритма основным критерием было выбрано время преобразования при условии достаточной стойкости полученного шифротекста к прямым атакам, т. е. перебору всех возможных вариантов для получения исходного сообщения. В качестве алгоритма, используемого для сравнения, был выбран RC4 [3] как удовлетворяющий описанным выше требованиям [4].

В исследуемом алгоритме использовались функции, построенные с помощью метода минимизации слабо определенных функций, описанных в работе [1]. Для упрощения процесса оценки функция строилась для аргументов размером 4 бита, доопределенных до размера 5 бит таким образом, чтобы все элементы набора не были связаны между собой сдвигом [1].

■ ***Таблица 1.*** *Исходные данные*

Входной массив информации	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
Выходной массив информации	03	09	0E	0B	0F	04	07	0C	0A	02	06	00	0D	05	08	01

Выходной массив информации задавался случайным образом и приведен в табл. 1.

Построенные методом минимизации [5] функции кодирования $b1_{enc}[i]$ и декодирования $b1_{dec}[i]$ для данных значений имеют вид

```
b1enc[i]=(b0[7]&b0[9]&b0[10]&b0[11])|(!b0[4]&b0[6]&b0[7]
&b0[8]&!b0[9]&!b0[10])|(!b0[6]&b0[7]&b0[9]
&!b0[10]&!b0[11])|(!b0[5]&b0[6]&!b0[7]
&!b0[8]&b0[9])|(!b0[4]&!b0[5]&b0[7]
&b0[8]&!b0[9]&!b0[10])|(b0[4]&b0[5]&!b0[7]
&b0[8])|(b0[6]&b0[7]&!b0[8]&b0[9]&!b0[10])
|(!b0[6]&!b0[9]&b0[10]&!b0[11])|(!b0[7]&b0[9]
&!b0[10]&b0[11])|(!b0[4]&!b0[5]&b0[6]&!b0[7]&b0[8]
&!b0[10])|(!b0[7]&!b0[8]&b0[9]&b0[11])
|(b0[5]&!b0[6]&!b0[7]&b0[8]&!b0[9])|(b0[6]&b0[7]
&!b0[8]&!b0[9]&b0[10])|(!b0[6]&!b0[8]&b0[10]&!b0[11])
|(b0[4]&!b0[6]&!b0[7]&b0[8])|(b0[7]&!b0[9]&!b0[10]&b0[11])
|(!b0[4]&!b0[5]&!b0[6]&!b0[7]&!b0[8]&!b0[9]&!b0[10]);
```

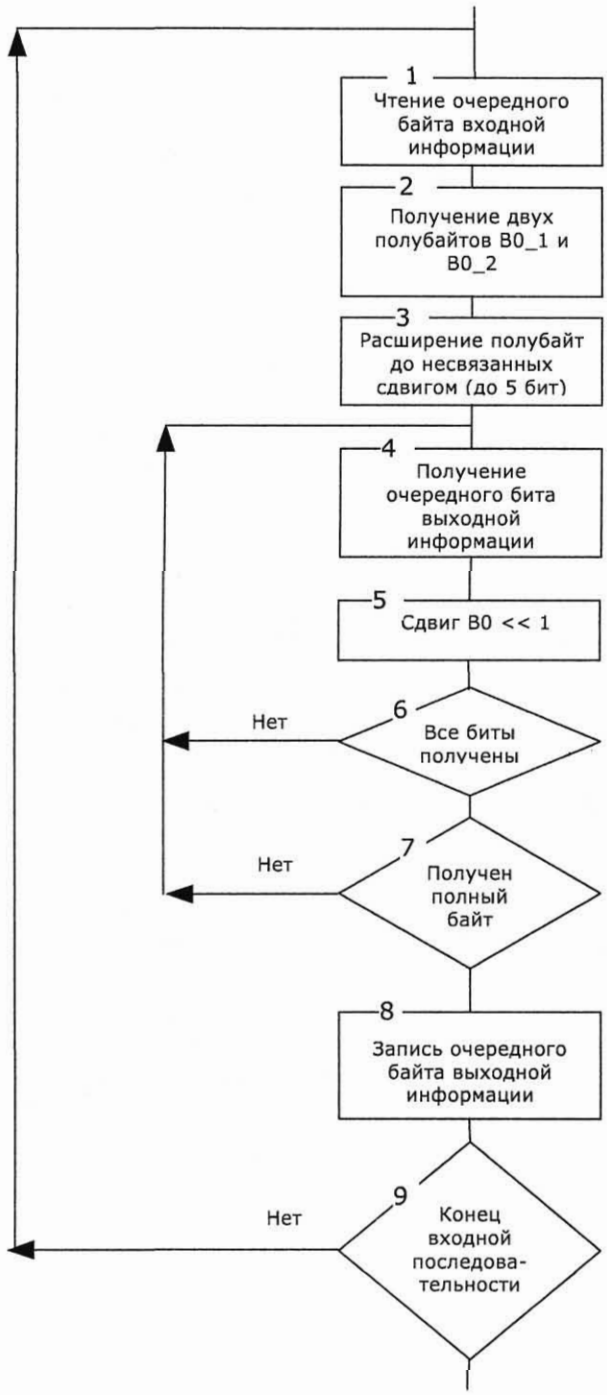
```
b1dec[i]=(b0[5]&!b0[6]&!b0[8]&b0[9])|(b0[6]&!b0[7]&!b0[8]
&!b0[9]&b0[10])|(b0[7]&!b0[8]&!b0[9]&b0[11])|(!b0[4]
&!b0[6]&b0[7]&b0[8]&!b0[11])|(!b0[5]&!b0[6]&b0[8]
&b0[9]&!b0[11])|(b0[4]&!b0[5]&!b0[7]&b0[8])|(b0[5]&b0[7]
&b0[9])|(!b0[6]&b0[7]&!b0[8]&b0[9]&!b0[10])|(b0[4]
&!b0[5]&b0[6]&b0[8])|(!b0[6]&!b0[7]&!b0[8]&b0[10]&!
b0[11])|(!b0[4]&!b0[5]&!b0[6]&!b0[7]&!b0[9]&!
b0[10])|(!b0[4]&b0[5]&!b0[7]&b0[8]&!b0[9])|(b0[7]
&b0[8]&!b0[9]&b0[10])|(!b0[7]&!b0[8]&b0[9]
&b0[11])|(b0[6]&b0[7]&!b0[8]&b0[9]&b0[10]),
```

где b0 – регистр, содержащий расширенный до 12 бит полубайт входной информации; b1 – регистр, содержащий полубайт выходной информации; b0[*n*] – *n*-й бит регистра b0; ! – булева операция «НЕ»; & – булева операция «И»; | – булева операция «ИЛИ».

Рассматриваемый алгоритм является итерационным и поэтому для получения 5 бит выходной информации необходимо преобразовать функцией 5 бит входной информации пять раз, на каждом новом шаге сдвигая ее влево на одну позицию. Как видно из самой функции, очередное значение бита выходного полубайта рассчитывается с использованием 11 бит входного слова.

Укрупненная структурная схема алгоритма преобразования представлена на рис. 1. В данном алгоритме на первом шаге считывается очередной байт информации, который на втором шаге разбивается на два полубайта. Каждый из этих двух полубайтов расширяется до 5 бит с целью получения чисел, не связанных сдвигом, путем установки пятого бита в единицу. В этот момент времени два полубайта хранятся в переменных размером 12 бит, в которых биты с шестого по двенадцатый равны нулю. Каждая из переменных преобразуется в итерационном цикле четыре раза для получения четырех бит выходной информации, которые затем объединяются в байт и записываются в выходную последовательность.

Преобразования по рассматриваемому алгоритму были реализованы на языке C++ с использованием контейнерного класса std::bitset стандартной библиотеки STL. Для компиляции программы были использованы компиляторы Borland C++ (с оптимизацией по скорости и без) и Microsoft C++ .NET с оптимизацией по скорости.



■ ***Рис. 1.*** *Укрупненная схема алгоритма реализации криптопреобразования на основе булевых функций*

Тестирование полученных реализаций проводилось при шифровании файла объемом 451 710 байт без учета дисковых операций и второстепенных вычислений. Учитывалось суммарное время работы основного итерационного цикла алгоритма (см. рис. 1).

■ ***Таблица 2.*** *Сравнение результатов*

Алгоритм	Булевы преобразования			RC4
	Borland C++, без оптимизации по скорости	Borland C++, с оптимизацией по скорости	Microcoft C++ .NET, с оптимизацией по скорости	Borland C++, с оптимизацией по скорости
Время преобразования файла размером 451 710 байт, с	41,93	8,90	0,88	0,03

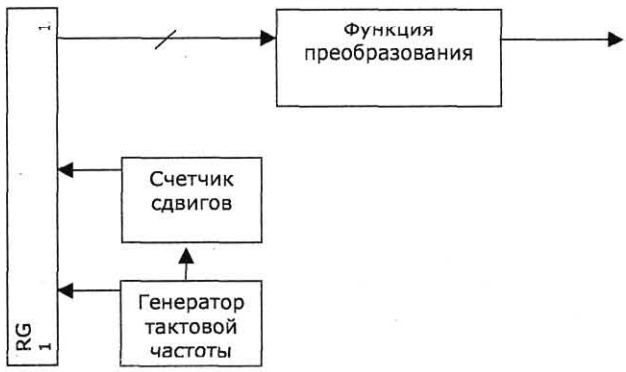
Сравнительные результаты времен преобразования указанного файла предлагаемым алгоритмом и RC4 приведены в табл. 2.

Как видно из результатов тестирования, даже лучшая реализация на основе булевых преобразований уступает по скорости (при прочих равных условиях) реализации алгоритма RC4, скорость которого принята за достаточную для потокового шифрования [3]. Основная причина – сложность программной реализации методов работы с векторами бит и операций над отдельными элементами таких векторов в языках программирования высокого уровня.

Оценка аппаратной реализации

Для защиты потоковой информации с использованием булевых преобразований в информационно-управляющих системах на основе дистанционно-управляемых автономных модулей [3] с интенсивными обменами необходимо использовать простую аппаратную реализацию, встраиваемую в сетевой канал таких модулей. Анализ сложности преобразования и обеспечения непрерывности информационного тракта показывает, что аппаратно возможна реализация на программируемой логике, например фирмы «ALTERA».

Аппаратная реализация алгоритма представлена на рис. 2.



■ ***Рис. 2.*** *Схема аппаратной реализации*

Логика функционирования

Каждый такт в сдвиговом регистре *RG* происходит сдвиг исходной информации вправо на один бит на входах логики преобразования, реализующей функцию преобразования. После установления новых значений по истечении времени срабатывания на выходе логики преобразования появляется очередной бит кодированной информации. Таким образом, при выборе метода кодирования по 4 бита для преобразования байта информации потребуется два полных цикла работы схемы.

Период тактовой частоты выбирается из соображений гарантированного срабатывания схемы программируемой логики, реализующей преобразования. Таким образом, для определения периода тактовой частоты и, как следствие, временного интервала кодирования порции потоковой информации необходимо определить время срабатывания самого длинного пути в логике преобразования. Такой путь представлен на рис. 3.

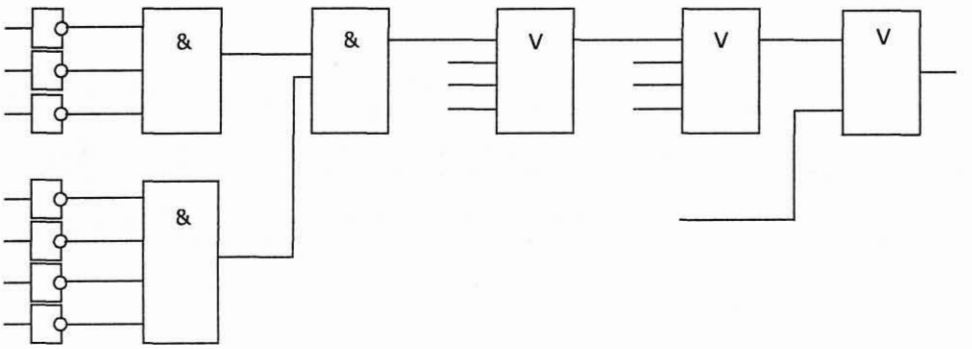
Полученный для наших исходных данных путь полностью умещается в одном логическом блоке ALTERA. Соответственно, период тактовой частоты для данной схемы будет равен времени срабатывания одного логического блока, что соответствует, в зависимости от типа выбранного для реализации блока, задержке от 4,5 до 15 нс. Полученная скорость кодирования составит от 10 до 30 Мбайт/с. Следует помнить, что данный результат получен для функции кодирования полубайта. Данное условие было введено для наглядности анализа и не подходит для реальных условий эксплуатации в связи с небольшой стойкостью к атакам по шифротексту. Даже булева функция кодирования байта гораздо менее тривиальна, ее аппаратная реализация имеет гораздо более длинные пути прохождения сигнала.

Для аппаратной реализации стандартного алгоритма RC4 ($n = 8$) требуется один буфер 256 байт только для чтения, для хранения ключа, один буфер 256 байт с произвольным доступом для хранения рабочего блока и два 8-битных регистра. Для кодирования одного блока информации в 256 байт по алгоритму RC4 требуется 1286 циклов работы [6]. Аппаратная реализация алгоритма RC4 на семействе микросхем ALTERA FLEX8000 способна выдавать 11 000 блоков в секунду, что составит около 3 Мбайт/с. Такая достаточно низкая скорость для потокового алгоритма связана с тем, что RC4 разрабатывался с условием оптимизации скорости программных реализаций алгоритма. Для сравнения компьютер на базе процессора Intel Pentium 100 способен выдавать около 22 000 блоков в секунду [6].

Заключение

Предложенный в работе алгоритм открывает новое направление в защите информации, поскольку использует булевы функции как функции криптографического преобразования данных.

Хотя программные реализации показали несколько худшие по сравнению с выбранным эталонным алгоритмом RC4 результаты, необходимо отметить несколько несомненных достоинств булевых преобра-



■ ***Рис. 3.*** *Самый длинный путь в блоке логики преобразования при аппаратной реализации*

зований, привлекательных при аппаратной реализации:

гибкость по отношению к размеру элементарного блока – метод булевых функций можно использовать с любым произвольным элементарным блоком, вплоть до двух бит;

криптостойкость и скорость преобразования информации булевыми функциями находятся в линейной зависимости от размера элементарного блока и легко поддаются прогнозированию;

отсутствие зависимости от предыдущей информации, позволяющее избежать необходимости в синхронизации при обрыве связи в информационном канале;

отсутствие буферных схем для временного хранения информации;

отсутствие ключа в обычном смысле – в качестве ключа в алгоритме используется булева функция, что несколько увеличивает криптостойкость;

значительно более простая аппаратная реализация по сравнению с известными алгоритмами.

Литература

1. **Ерош И. Л.** Защита информационных потоков в системах распределенного контроля и управления // Информационно-управляющие системы. – 2002. – № 1. – С. 40–46.
2. **Бубликов А. Б.** Организация защищенного канала передачи видеопотока в информационно-управляющих IP-системах // VI научная сессия аспирантов ГУАП: Сб. докл.: В 2-х ч. – Ч. I. Технические науки. – СПб.: Изд-во ГУАП, 2003. – С. 211 – 212.
3. **Шнайер Б.** Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.: Триумф, 2003. – 816 с.
4. **Астапкович А. М., Востриков А. А., Сергеев М. Б., Чудиновский Ю. Г.** Информационно-управляющие системы на основе Интернет // Информационно-управляющие системы, 2002. – № 1. – С. 12–18.
5. **Ерош И. Л.** Дискретная математика. Булева алгебра, комбинационные схемы, преобразования двоичных последовательностей. Учебн. пособ. – СПб.: Изд-во ГУАП, 2001. – 96 с.
6. **Goldberg I., Wagner. D.** Architectural considerations for cryptanalytic hardware – ISAAC Group U. C. Berkeley, (**http://www.cs.berkeley.edu/~iang/isaac/hardware**)

УДК : 681.3.06: 616.089.5

ПЕРСПЕКТИВЫ ПРИМЕНЕНИЯ В АВИАЦИИ ИНТЕГРИРОВАННЫХ НАШЛЕМНЫХ СИСТЕМ НЕЙРОФИЗИОЛОГИЧЕСКОГО КОНТРОЛЯ

А. П. Шепета,
д-р техн. наук, прфессор
И. О. Жаринов,
ассистент
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Настоящая статья направлена на решение актуальной задачи организации и обеспечения безопасности (надежности) полетов пилотируемых летательных аппаратов. В работе рассматривается эффективный нейрофизиологический подход к комплексной оценке физиологического состояния пилота в динамике полета летательного аппарата в реальном масштабе времени. Применяется математический аппарат теории статистических решений, вероятностные методы и модели, основанные на статистической обработке нестационарных и квазистационарных гауссовских случайных процессов, аппроксимирующих реализации электрофизиологических показателей пилотов ЛА. Результаты работы имеют наибольший интерес для учета условий повышенных динамических перегрузок, испытываемых пилотом ЛА, когда организация эффективного управления параметрами полета осложнена предельными импульсными нагрузками. Создание надежной автоматической системы анализа комплексного физиологического состояния пилота ЛА и ее введение в контур системы автопилота позволит значительно повысить безопасность (надежность) полетов.

The present article is directed on the solution of an actual problem of flights organization and safety control (reliability). In activity the effective physiological approach to complex estimation of pilot physiological condition in flight dynamics in real time is esteemed. The mathematical methods of the theory by the statistical solutions, probabilistic methods and models, founded on statistical processing of non-steady and pseudo-steady normal stochastic processes, approximating implementation of electrophysiological parameters of the pilots is applied. The outcomes of activity have the greatest concern for the count of conditions heightened dynamic overloads assayed by the pilot, when the organization of efficient control flight parameters is complicated by limiting impulsive loads. The creation of reliable automatic system of complex analysis physiological condition of the pilot and its introducing in contour of autopilot system will allow considerably to increase safety (reliability) of flights.

Введение

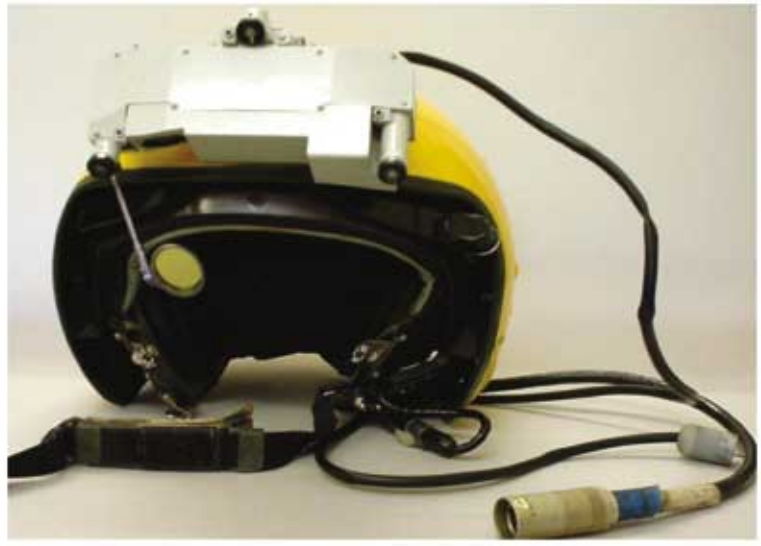
Нашлемная система целеуказания и индикации (НСЦИ) представляет собой одну из наиболее совершенных систем авиационной электроники, предназначенных для управления оружием и режимами стрельбы современных истребителей и вертолетов.

В отличие от систем, использующих для управления оружием коллиматорные авиационные индикаторы (индикаторы на лобовом стекле), она не требует доворота летательного аппарата (ЛА) в направлении цели и обеспечивает прицеливание управляемого оружия в направлениях, не совпадающих со строительной осью ЛА. При прицеливании пилоту необходимо совместить с целью прицельную метку, индицируемую с помощью коллиматорной подсистемы индикации, жестко закрепленной на шлеме (рис. см. на 3-й стороне обложки), и осуществить пуск оружия. Этот способ дает возможность существенно сократить время прицеливания и обеспечить преимущество в воздушном бою с ЛА, не оснащенными НСЦИ, а также быстрое поражение наземных целей и уход от средств противовоздушной обороны противника [3].

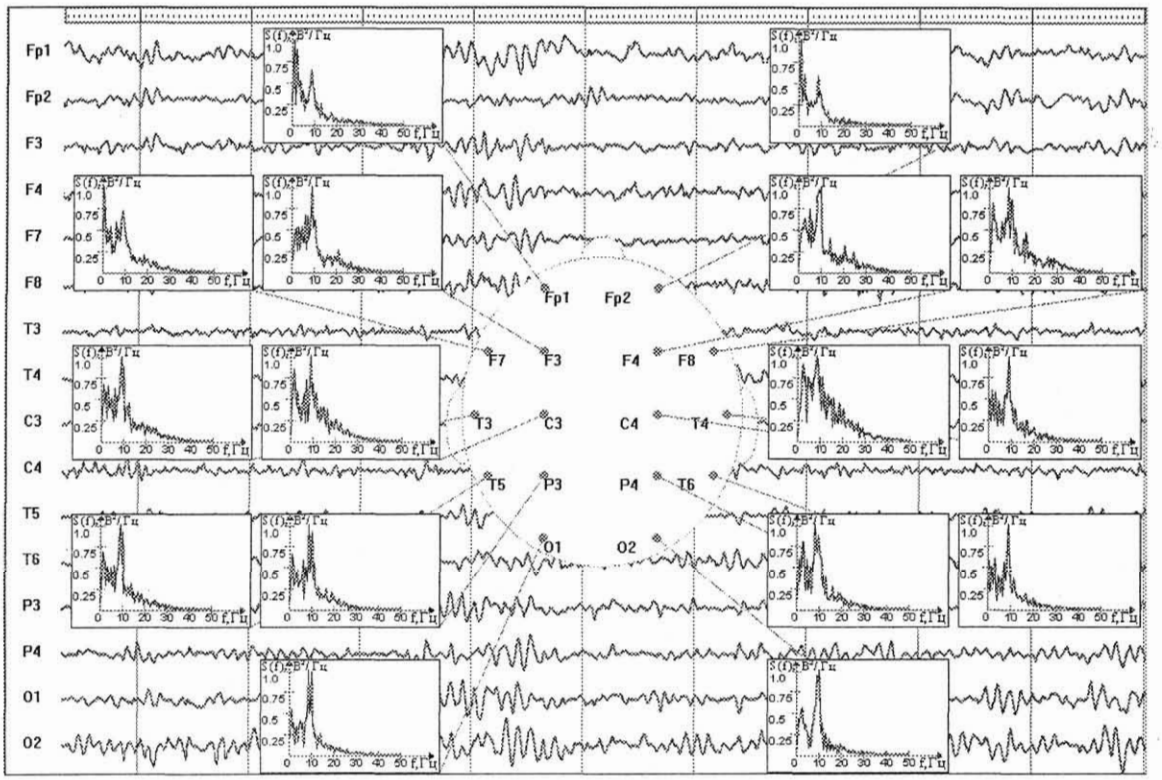
Сегодня НСЦИ широко разрабатываются и производятся многими зарубежными фирмами [3], в частности, «Sextant Avionics» (Франция), «Kayser Electronics» (США), «GEC-Marconi Avionics» (США), «Elbit» (Израиль), АО «Арсенал» (Украина), СПб ОКБ «Электроавтоматика» (Россия).

Мировой опыт применения НСЦИ и эксплуатации авиационной техники и авиатренажеров определил новое перспективное направление функциональной нагрузки комплексных нашлемных систем – создание эффективных средств анализа и статистической обработки реализаций электрических сигналов жизнедеятельности пилота, отражающих его физиологическое состояние в условиях динамики полета современного ЛА.

Широко известны трагические случаи гибели как гражданских, так и военных самолетов из-за отсутствия автоматических средств слежения и контроля состояния организма пилотов, а также резервирующих систем принятия экстренных решений в реальном масштабе времени.



Рисунки к статье
А. П. Шепеты и И. О. Жаринова
“Перспективы применения в авиации
интегрированных нашлемных систем
нейрофизиологического контроля”,
см. с. 58–62



■ ***Рис. 1.*** *Система отведений электроэнцефалографического процесса*

Регистрация нейросигнала пилота ЛА

Одной из важнейших задач человечества в XXI веке является обеспечение безопасности полетов пилотируемых летательных аппаратов за счет снижения удельного числа авиакатастроф, происходящих по причине отсутствия систем жизнеобеспечения и резервирования на борту современных самолетов.

В 1996 г. авторы настоящей статьи совместно с рядом авторитетных научных сотрудников в нашей стране и за рубежом предложили альтернативную теории информации идею определения адекватности поведения человека в условиях наличия внешних факторов на основе комплексного анализа ограниченного числа электрофизиологических показателей жизнедеятельности организма человека. В рамках проведения ряда НИР (1997–2002 гг.) и исследований по грантам (1999–2002 гг.) показано, что разумно выбранные показатели в полной мере могут отражать текущее состояние пилота, а грамотно выбранный математический аппарат позволяет проводить непрерывное сопровождение и оценку его состояния на основе подходов по сегментации (разделению во времени на классы состояний) и многоальтернативной классификации реализаций информационных сигналов, получаемых от датчиков нейрофизиологического контроля (рис. 1).

Электроэнцефалография является одним из основных методов объективного тестирования функций нервной системы человека. В настоящее время не существует универсальных алгоритмов, пригодных для всестороннего анализа сигнала электроэнцефалограммы (ЭЭГ), а тем более решения задач автоматической диагностики. Однако многие прикладные задачи решаются при помощи ЭВМ весьма успешно. Общие закономерности формирования сигнала составляют априорную информацию, которая используется для синтеза и оптимизации алгоритмов статистической обработки. В частности, экспериментально подтверждено, что нестационарный случайный процесс, соответствующий ЭЭГ-сигналу, может быть разделен на квазистационарные участки различной длительности, в пределах которых статистические свойства процесса существенно не изменяются. Эти квазистационарные участки описывают физиологические состояния соответствующих отделов головного мозга и являются важными диагностическими признаками. Определение границ квазистационарных участков составляет задачу сегментации ЭЭГ. При этом важное значение имеют статистические или спектральные характеристики сегментов, поэтому, кроме задачи собственно сегментации [2, 4], необходимо решать задачу классификации сигнала [1, 2].

Авторегрессионный анализ в электроэнцефалографии

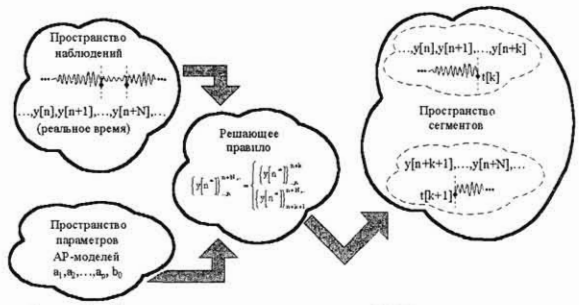
Использование для анализа и математической аппроксимации ЭЭГ-сигналов параметрических моделей авторегрессии – скользящего среднего (АРСС) находится сейчас в стадии интенсивного развития: совершенствуются вычислительные методы, уточняются статистические аспекты, выясняются границы применимости.

АРСС-анализ базируется на предположении, что текущие значения сигнала имеют статистическую связь с его «предысторией». АРСС-модель ЭЭГ представляет значения дискретных отсчетов $y^{(i)}[n]$ i-го квазистационарного участка посредством линейного соотношения вида

$$y^{(i)}[n] + \sum_{k=1}^{p^{(i)}} a_k^{(i)} \times y^{(i)}[n-k] = b_0^{(i)} \times x[n] + \sum_{k=1}^{q^{(i)}} b_k^{(i)} \times x[n-k],$$

$$i = 1, 2, \ldots, M,$$

■ ***Рис. 2.*** *Диаграмма параметрического синтеза моделей ЭЭГ-сигналов*



■ ***Рис. 3.*** *Диаграмма сегментации ЭЭГ-сигнала*

где $\{x[n]\}$ – порождающий процесс (последовательность независимых, нормально распределенных случайных величин с нулевым математическим ожиданием и единичной дисперсией). Коэффициенты $\left\{a_1^{(i)}, a_2^{(i)}, \ldots, a_{p^{(i)}}^{(i)}\right\}$ и $\left\{b_0^{(i)}, b_1^{(i)}, \ldots, b_{q^{(i)}}^{(i)}\right\}$, а также величины $p^{(i)}$ и $q^{(i)}$ являются параметрами модели. Параметры $p^{(i)}$ и $q^{(i)}$ определяют порядок модели авторегрессии и скользящего среднего, соответственно, а величина (p^i+q^i) определяет порядок АРСС-модели. Индекс *(i)* означает описание фрагмента сигнала, соответствующего определенному физиологическому состоянию человека, моделью *i*-го класса из *M* возможных взаимоальтернативных классов.

Адекватную параметрическую модель ЭЭГ удается получить (рис. 2), даже полагая коэффициенты $\left\{b_1^{(i)}, \ldots, b_{q^{(i)}}^{(i)}\right\}$ равными нулю.

При этом уравнение определяет чисто авторегрессионную модель ЭЭГ

$$y^{(i)}[n] + \sum_{k=1}^{p^{(i)}} a_k^{(i)} y^{(i)}[n-k] = b_0^{(i)} x[n], \; i = 1, 2, \ldots, M.$$

В последующем через параметры АРСС-модели могут быть выражены модель сигнала, оценки частотных и корреляционных функций, определены отрезки стационарности сигнала, решены задачи сегментации и классификации.

Сегментация ЭЭГ-сигнала

Анализ экспериментально полученных реализаций составляет существенную часть научного исследования в области электроэнцефалографии. Часто оказывается целесообразным выделить в реализации сигнала некоторые участки, которые следует рассматривать отдельно. Эти участки можно охарактеризовать набором признаков и получить сжатое описание ЭЭГ как описание взаимного расположения выделенных участков (классов) с учетом названий их форм, что дает представление о последовательности сменяющих друг друга физиологических состояний с детекцией их продолжительности во времени.

С содержательной точки зрения ЭЭГ можно разделить на чередующиеся участки относительно «плавного» хода и короткие «переходные» участки, характеризующиеся быстрым изменением формы кривой. Изменчивость и сложность поведения сигнала на переходных участках позволяет их обнаруживать и использовать в качестве границ сегментации.

Для математической формулировки задачи сегментации необходимо описать стационарные участки ЭЭГ, соответствующие выбранному множеству классов *I*, некоторой математической моделью. Класс ЭЭГ описывается как последовательность независимых случайных величин $\{y[n^*]\}_{\ldots,n-N}^{n,\ldots} = \ldots, y[n-N], y[n-N+1], \ldots, y[n], \ldots$ с функцией плотности вероятности $p_N^{(i)}\left\{\{y^{(i)}[n^*]\}_{n-N}^{n}, Q^{(i)}(t)\right\}$, зависящей от вектора параметров $Q^{(i)}(t)$. Для параметров характерны скачкообразные изменения в неизвестный момент времени t^*:

$$Q^{(i,k)}(t) = \begin{cases} Q^{(i)}(t), \; t \le t^*; \\ Q^{(k)}(t), \; t > t^*, \; i \ne k. \end{cases}$$

Момент изменения (МИ) t^* считается неизвестным параметром с множеством возможных значений *T*. По наблюдениям $\{y[n^*]\}_{n-N}^{n}$ реализации сигнала необходимо обнаружить МИ и оценить его значение t^{**}.

В случае описания классов ЭЭГ АРСС-моделями задача сегментации (рис. 3) формулируется следующим образом. Параметры модели *i*-го класса $\left\{a_1^{(i)}, a_2^{(i)}, \ldots, a_{p^{(i)}}^{(i)}, b_0^{(i)}, b_1^{(i)}, \ldots, b_{q^{(i)}}^{(i)}\right\}$ в каждый момент времени $t[n]$ объективно принадлежат одному из *M* классов и остаются постоянными на таком промежутке времени, число отсчетов реализации сигнала в котором больше порядка АРСС-модели $\left(p^{(i)} + q^{(i)}\right)$ этого класса. Тогда, по имеющейся реализации случайной последовательности $\{y[n-N], y[n-N+1], \ldots, y[n]\}$, необходимо определить, к какому из *M* возможных взаимоисключающих классов принадлежит данный участок сигнала. Математически процесс перехода ЭЭГ из одного класса *i* в другой класс *k* описывается выражением

$$Q^{(i,k)}(t) = \begin{cases} \left\{a_1^{(i)}, a_2^{(i)}, \ldots, a_{p^{(i)}}^{(i)}, b_0^{(i)}, b_1^{(i)}, \ldots, b_{q^{(i)}}^{(i)}\right\}, \\ t \le t^*; i = 1,2,\ldots,M; \\ \left\{a_1^{(k)}, a_2^{(k)}, \ldots, a_{p^{(k)}}^{(k)}, b_0^{(k)}, b_1^{(k)}, \ldots, b_{q^{(k)}}^{(k)}\right\}, \\ t > t^*; k = 1,2,\ldots,M, i \ne k. \end{cases}$$

Границей сегмента является МИ t^*, обнаружить который (получить оценку) необходимо с наименьшим запаздыванием.

Свойства сегментов весьма чувствительны к тонким сдвигам функционального состояния, объективно харак-

теризуют структуру ЭЭГ и отражают микросостояния головного мозга. Моменты резких изменений индицируют реакции ЭЭГ на «переключения» различных систем головного мозга из одного микросостояния в другое, имеют важное значение для исследования отделов коры больших полушарий с помощью анализа совпадений во времени МИ в различных отведениях ЭЭГ, позволяют получить информацию о пространственно-временной организации процессов, проходящих в мозге.

Классификация ЭЭГ-сигнала

Классификация ЭЭГ представляет собой процедуру отнесения исследуемого участка (сегмента) реализации сигнала, задаваемого в виде последовательности наблюдаемых значений $\{y[n^*]\}_{n-N}^{n}$, к одному из M взаимоисключающих (взаимоальтернативных) классов $i = 1, 2, \ldots, M$. Это означает, что существует однозначное отображение совокупности наблюдаемых значений, являющихся конечным числовым множеством Y, на множество классов I, количество элементов в котором задано и равно M. Без потери общности, классы $i = 1, 2, \ldots, M$ могут быть заменены их номерами $1, 2, \ldots, M$ (натуральными числами). Тогда классификация представляет собой процедуру отображения наблюдаемых значений $\{y[n^*]\}_{n-N}^{n}$ на конечное множество натуральных чисел $\{y[n^*]\}_{n-N}^{n} \to \{1,2,\ldots,M\}$. Принимая во внимание числовую природу множеств, последнее отображение отождествляется (рис. 4) с некоторой решающей функцией $i = func\left(\{y[n^*]\}_{n-N}^{n}\right)$, принимающей целочисленные значения $i = \{1, 2, \ldots, M\}$.

В практической задаче классификации ЭЭГ множество наблюдений имеет простую структуру и может быть представлено в виде оцененных на основе измеренных значений $(p+q)$-характеристик $\left\{a_1^{(i)}, a_2^{(i)}, \ldots, a_{p^{(i)}}^{(i)}, b_0^{(i)}, b_1^{(i)}, \ldots, b_{q^{(i)}}^{(i)}\right\}$ классов [5]. При этом, если обозначить количество наблюдений некоторым числом υ, то классификация сигналов сводится к $(p+q)\upsilon$ – аргументной решающей функции

$$i = func\left(a_1^{(i)}, a_2^{(i)}, \ldots, a_{p^{(i)}}^{(i)}, b_0^{(i)}, b_1^{(i)}, \ldots, b_{q^{(i)}}^{(i)}, \upsilon^{(i)}\right),$$

принимающей целочисленные значения $i = \{1, 2, \ldots, M\}$.

Важнейшей особенностью реальных значений является то, что наблюдения $\left\{\{y[n^*]\}_{n-N}^{n}\right\}_{pq\upsilon}$ неизбежно подвержены случайным возмущениям, непредсказуемый вероятностный характер которых сказывается на всех стадиях, начиная от процесса получения (регистрации) самих измерений $\{y[n^*]\}_{n-N}^{n}$ и заканчивая вычислением целочисленных значений решающей функции $func\left(a_1^{(i)}, a_2^{(i)}, \ldots, a_{p^{(i)}}^{(i)}, b_0^{(i)}, b_1^{(i)}, \ldots, b_{q^{(i)}}^{(i)}, \upsilon^{(i)}\right)$. Дестабилизирующими факторами при классификации ЭЭГ выступают как погрешности измерений, измерительных приборов и неточности регистрации, так и шумы и помехи физической физиологической природы. Взаимодействуя между собой, указанные возмущения приводят к тому, что наблюдения оказываются реализациями случайных велечин, решающая функция $func\left(a_1^{(i)}, a_2^{(i)}, \ldots, a_{p^{(i)}}^{(i)}, b_0^{(i)}, b_1^{(i)}, \ldots, b_{q^{(i)}}^{(i)}, \upsilon^{(i)}\right)$ становится случайной функцией, и в результате номер i-го класса также оказывается случайной величиной. Следовательно, классификация ЭЭГ связана с исследованием случайных отображений и оказывается возможной только на основе статистических методов обработки.

Источником информации о распознаваемых классах ЭЭГ-сигнала в ряду статистических экспериментов является совокупность результатов независимых наблюдений $\{y[n^*]\}_{n-N}^{n}$, составляющих обучающую N_T и контрольную экзаменационную N_K выборки.

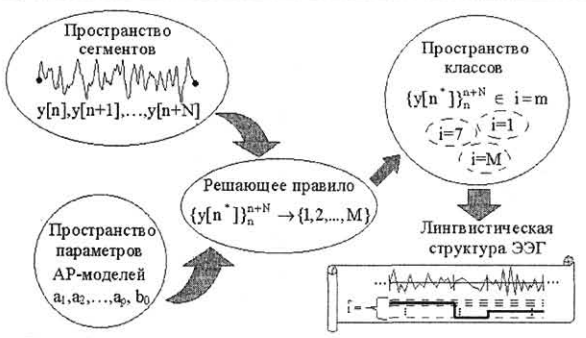
Обучение является неотъемлемой составной частью распознающего процесса и имеет своей конечной целью формирование эталонных описаний, форма которых определяется видом моделей и способом их использования в решающих правилах. Последние заимствуются статистическим распознаванием из хорошо разработанной теории статистических решений, в рамках которой для нормального закона распределения при гипотезе и альтернативе оптимальные решающие правила основаны на формировании статистики отношения правдоподобия $\Lambda^{(i,k)}[n]$ и сравнении его значения, или некоторой монотонной функции от него, с определенным пороговым уровнем C, значение которого определяется выбранным критерием качества:

$$\Lambda^{(i,k)}[n] = \frac{p\left\{\{y[n^*]\}_{n-N}^{n} / Q^{(i)}\right\}}{p\left\{\{y[n^*]\}_{n-N}^{n} / Q^{(k)}\right\}} \geq C,$$

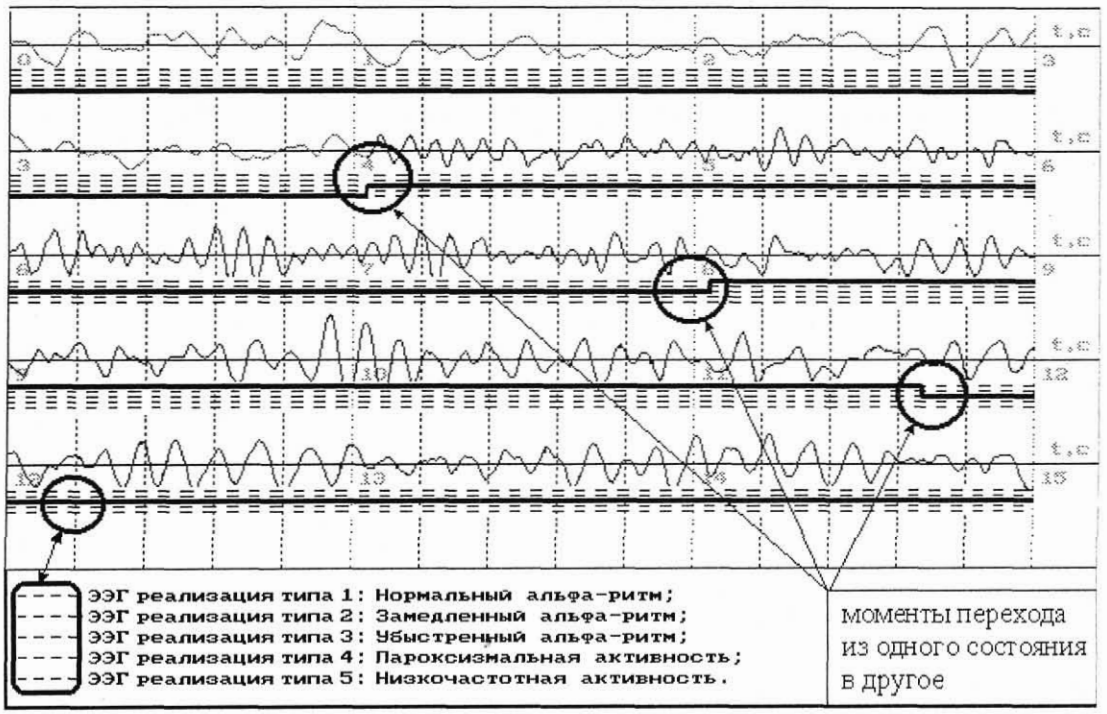
где $p\left\{\{y[n^*]\}_{n-N}^{n} / Q^{(i)}\right\}$ – условная совместная N-мерная функция плотности вероятности выборочных значений $\{y[n^*]\}_{n-N}^{n}$ при условии их принадлежности к i-му классу. В случае описания классов авторегрессионными моделями функция плотности вероятности имеет вид [4]

$$p\left\{\{y[n^*]\}_{n-N}^{n} / Q^{(i)}\right\} =$$

$$= \frac{1}{b_0^{(i)}\sqrt{2\pi}} \exp\left(-\frac{1}{2\left(b_0^{(i)}\right)^2}\left(y[n] + \sum_{k=1}^{p^{(i)}} a_k^{(i)} y[n-k]\right)^2\right).$$



■ ***Рис. 4.*** *Диаграмма классификации ЭЭГ-сигнала*



■ ***Рис. 5.*** *Пример автоматической сегментации и классификации ЭЭГ-сигнала*

Выбор порогового уровня *C* определяется в соответствии с критериями теории статистических решений: байесовским, Неймана–Пирсона, минимаксного, Вальда, максимума апостериорной вероятности, максимального правдоподобия и т. д.

Результаты

На рис. 5 в наглядной графической форме представлен результат практической обработки авторским программным обеспечением 15-секундной реализации реального ЭЭГ-сигнала, содержащего несколько различных сегментов, отражающих определенные физиологические состояния организма человека.

Отмечены моменты автоматической (в реальном масштабе времени) детекции процессов скачкообразного изменения свойств ЭЭГ и перехода сигнала из одного класса физиологического состояния человека в другой. Рис. 5 иллюстрирует возможность обеспечения практической реализации предложенного подхода и гарантирует (с заданной достоверностью) вероятностные характеристики правильной классификации и точностные характеристики сегментации ЭЭГ человека.

Заключение

Предложенный вариант модернизации интегрированной системы НСЦИ для физиологического контроля состояния пилота ориентирован на создание перспективных решений в области аэрокосмических технологий современного уровня в рамках работ по созданию самолетов пятого поколения, а также на создание отдельных элементов систем управления более широкого применения.

Рассмотренные научно-технические подходы по проблемам анализа электрофизиологических показателей человека в значительной мере относятся и к проблеме создания перспективной аппаратуры общего медицинского назначения, что, в свою очередь, способствует определению эффективных путей развития отечественного здравоохранения.

Работа поддержана грантами на исследования: М99-3.5Д-260, М01-3.5К-80, М02-3.5К-127, М03-3.5К-3, А232-03 комитета по науке и высшей школе Администрации Санкт-Петербурга.

Литература

1. **Жаринов И. О.** Фоновый контроль физиологического состояния пилота летательного аппарата по его электроэнцефалограмме // Авиакосмическое приборостроение. – 2003. – № 5. – С. 46–54.
2. **Жаринов И. О.** Статистический анализ информационных сигналов от датчиков медицинских приборов (электроэнцефалография) // Датчики и системы. – 2003. – № 7. – С. 23–29.
3. **Феофанов В. К., Парамонов П. П., Суслов В. Д., Сабо Ю. И.** Нашлемная система целеуказания и индикации на базе координаточувствительного фотоприемника Мультискан // Датчики и системы. –2001. – № 8. – С. 2–3.
4. **Шепета А. П., Жаринов И. О.** Организация и обеспечение безопасности полетов методами последовательного анализа ЭЭГ пилота летательного аппарата // В кн.: Информационно-управляющие системы для подвижных объектов. Семинары ASK LAB 2001 / Под общ. ред. М. Б. Сергеева – СПб.: Политехника, 2002. – С.118–143.
5. **Zharinov I. O.** Recognition of discrete stochastic processes in space of parametric autoregression models // Preprints of 9th international student olympiad on automatic control (Baltic olympiad) – St.P.: Saint Petersburg State Institute of Fine Mechanic and Optics. – 2002. – P. 85–90.

БУБЛИКОВ Андрей Борисович

Ассистент кафедры вычислительных систем и сетей СПбГУАП.
В 2001 году окончил Санкт-Петербургский государственный университет аэрокосмического приборостроения по специальности "Вычислительные машины, комплексы, системы и сети".
Автор 4 научных публикаций.
Область научных интересов – защита потоковой информации в сетях общего доступа.

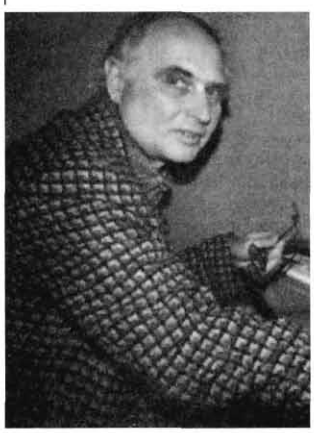
БУРАКОВ Михаил Владимирович

Доцент кафедры управления и информатики в технических системах СПбГУАП.
В 1984 году окончил Ленинградский институт авиационного приборостроения (ЛИАП) по специальности "Автоматизированные системы управления".
В 1994 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Автор около 100 научных публикаций.
Область научных интересов – системы автоматического управления, системы искусственного интеллекта.

ГОЛЛАНДЦЕВ Юрий Алексеевич

Ведущий научный сотрудник ЦНИИ "Электроприбор".
В 1967 году окончил Ленинградский институт авиационного приборостроения.
В 1978 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Является автором более 80 научных публикаций.
Область научных интересов – электромеханика и робототехника.

ГОРОДЕЦКИЙ Андрей Емельянович

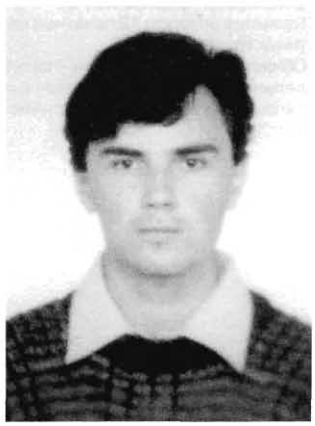


Доктор технических наук, профессор, заслуженный деятель науки и техники. Заведующий лабораторией методов и средств автоматизации Института проблем машиноведения РАН.
В 1965 году окончил Ленинградский Политехнический институт. В 1994 году защитил диссертацию на соискание ученой степени доктора технических наук. Является автором более 120 публикаций и 70 изобретений.
Область научных интересов – информационно-управляющие системы, нейронные сети, оптическое распознавание образов.

ЕРОШ Игорь Львович

Профессор кафедры вычислительных систем и сетей СПбГУАП. Академик Международной академии информатизации.
Член японской ассоциации промышленных роботов.
Награжден правительственными наградами "Изобретатель СССР", "Почетный работник высшего профессионального образования России", медалью "300 лет Санкт-Петербургу".
В 1961 году окончил Ленинградский электротехнический институт (ЛЭТИ).
В 1980 году защитил диссертацию на соискание ученой степени доктора технических наук.
Автор более 320 научных трудов, в том числе свыше 100 изобретений, соавтор 2-х учебников и 3-х монографий.
Область научных интересов – системы искусственного интеллекта, дискретная математика, распознавание образов, защита информации.

ЖАРИНОВ Игорь Олегович



Ассистент кафедры моделирования вычислительных и электронных систем Санкт-Петербургского государственного университета аэрокосмического приборостроения (СПбГУАП), ведущий инженер отдела систем авиационной индикации СПб ГУП ОКБ "Электроавтоматика".
В 2000 году окончил Санкт-Петербургский государственный университет аэрокосмического приборостроения.
Является автором более 30 научных публикаций.
Область научных интересов – обработка информации в условиях априорной неопределенности, математическое моделирование стохастических процессов и полей.

ИГНАТЬЕВ Михаил Борисович

Профессор Санкт-Петербургского государственного университета аэрокосмического приборостроения.
В 1955 году окончил Ленинградский политехнический институт.
В 1971 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 500 научных публикаций.
Область научных интересов – кибернетика, информатика, моделирование, системный анализ, вычислительная техника.

AL-KASASBEH Riad Taha

P.H.D in Biomedical Engineering, доцент Al-BALQA APPLIED UNIVERSITY.
В 1993 году окончил Ленинградский электротехнический институт.
В 1996 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Является автором 30 научных публикаций.
Область научных интересов – информационно-управляющие системы, нейронные сети, оптическое распознавание образов

КОНОВАЛОВ Александр Сергеевич

Заведующий кафедрой управления и информатики в технических системах СПбГУАП. Заведующий лабораторией автоматизации института проблем машиноведения РАН.
В 1968 году окончил Ленинградский институт авиационного приборостроения (ЛИАП) по специальности "Электрооборудование ракет и других летательных аппаратов".
В 1997 году защитил диссертацию на соискание ученой степени доктора технических наук.
Автор более 150 научных публикаций.
Область научных интересов – синтез нелинейных систем автоматического управления сложными объектами, системы искусственного интеллекта, системы автоматизированного проектирования.

КОПТЕВ Борис Анатольевич

Кандидат технических наук, начальник отдела 1 ЦНИИ МО.
В 1982 году окончил Высшее Военно-Морское училище имени М.В. Фрунзе, в 1990 году – Военно-Морскую академию имени Н.Г. Кузнецова.
В 2000 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Является автором более 20 научных публикаций.
Область научных интересов – математическая статистика.

НАУМОВ Лев Александрович

Магистрант кафедры "Компьютерные технологии" Санкт-Петербургского государственного университета информационных вычислений, механики и оптики.
Стипендиат Президента РФ.
Является автором 12 печатных научных трудов.
Область научных интересов – параллельные вычисления, клеточные автоматы, Grid – системы, вычислительные кластеры.

ПРОКАЕВ Александр Сергеевич

Преподаватель Военно-Морской академии имени Н. Г. Кузнецова.
В 1986 году окончил Военно-Морской институт радиоэлектроники по специальности "Автоматизированные системы управления".
В 2003 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Автор 9 научных публикаций.
Область научных интересов – теория поиска подвижных объектов.

РОЗОВ Алексей Константинович

Ведущий научный сотрудник 1-го ЦНИИ МО.
В 1974 году окончил Высшее инженерно-техническое училище ВМФ по специальности инженер-электромеханик.
В 1968 году защитил диссертацию на соискание ученой степени доктора технических наук.
Автор 25 научных публикаций.
Область научных интересов – применение статистических методов в задачах обнаружения, классификации и оценивания сигналов.

РОМАНОВСКИЙ Александр Феофанович

Преподаватель Военно-Морской академии.
Окончил Высшее военно-морское училище радиоэлектроники им. А.С.Попова в 1985 г. по специальности "Автоматизированные системы управления", Военно-Морскую академию им. Н.Г.Кузнецова в 1999 г.
Диссертацию на соискание ученой степени кандидата технических наук защитил в 2002 г.
Автор 15 научных публикаций.
Области научных интересов: механика двухсредных аппаратов, исследование нестационарных процессов в динамике движения объектов.

СЕРГЕЕВ Михаил Борисович

Профессор, заведующий кафедрой вычислительных систем и сетей Санкт – Петербургского государственного университета аэрокосмического приборостроения(СПбГУАП).
Главный редактор журнала "Информационно-управляющие системы".
В 1980 году окончил Ленинградский электротехнический институт (ЛЭТИ) по специальности "Электронные вычислительные машины".
В 2001 году защитил диссертацию на соискание ученой степени доктора технических наук.
Автор более 80 научных работ, в том числе двух монографий.
Под его редакцией издано пять сборников научных трудов и монографий.
Области научных интересов – теория разрядных вычислений, методы проектирования спецпроцессоров для систем контроля и управления.

СОЛОВЬЕВ Николай Владимирович

Старший преподаватель кафедры вычислительных систем и сетей Санкт -Петербургского государственного университета аэрокосмического приборостроения (СПбГУАП).
В 1979 году окончил Ленинградский политехнический институт (ЛПИ) по специальности "Автоматизация и комплексная механизация машиностроения".
Автор более 20 публикаций и 4 изобретений.
Область научных интересов – распознавание образов, компьютерная обработка и анализ изображений, техническое зрение роботов.

ТАРАСОВА Ирина Леонидовна

Старший научный сотрудник Института проблем машиноведения РАН.
В 1978 году окончила Ленинградский Политехнический институт.
В 1997 году защитила диссертацию на соискание ученой степени кандидата технических наук.
Является автором 35 научных публикаций.
Область научных интересов – информационно-управляющие системы, нейронные сети.

ШАЛЫТО Анатолий Абрамович

Заведующий кафедрой "Информационные системы" Санкт-Петербургского государственного университета информационных технологий, механики и оптики.
Ученый секретарь НПО "Аврора".
В 1971 году окончил ЛЭТИ по специальности "Автоматика и телемеханика".
В 1999 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 250 научных публикаций, 3 монографий и 70 изобретений.
Член редакционной коллегии журнала "Информационно-управляющие системы".
Области научных интересов – системы логического управления; автоматное программирование.

ШЕПЕТА
Александр
Павлович

Декан факультета Вычислительных систем и программирования Санкт-Петербургского государственного университета аэрокосмического приборостроения (СПбГУАП), заведующий кафедрой Моделирования вычислительных и электронных систем, профессор.
В 1972 году оакончил Ленинградский институт авиационного приборостроения. В 1995 году защитил диссертацию на соискание ученой степени доктора технических наук.
Имеет более 170 публикаций, соавтор двух монографий.
Академик МАНВШ.
Область научных интересов – обработка информации в условиях априорной неопределенности, математическое моделирование стохастических процессов и полей.

ПАМЯТКА ДЛЯ АВТОРОВ

Поступающие в редакцию статьи проходят обязательное рецензирование.

При наличии положительной рецензии статья редактируется и рассматривается редакционной коллегией. Принятая в печать статья направляется автору для согласования редакторских правок. После согласования автор представляет в редакцию окончательный вариант текста статьи, а также фотографию и краткое изложение сведений о себе.

Процедуры согласования текста статьи, предоставления фото (размером 4×5,5 см) и сведений об авторе могут осуществляться как непосредственно в редакции, так и по e-mail (электронный вариант фото в виде файла *.tif, *.jpg с разрешением 300 dpi).

При отклонении статьи редакция представляет автору мотивированное заключение и рецензию. При необходимости доработать статью — рецензию.

Редакция журнала напоминает, что ответственность за подбор, достоверность и точность фактов, экономико-статистических и технических показателей, собственных имен и прочих сведений, а также за то, что в материалах не содержится сведений, не подлежащих открытой публикации, несут авторы публикуемых в журнале материалов и рекламодатели.

УДК 681.1

Модель управления действиями наблюдателя при вторичном поиске

Прокаев А.Н. – Информационно-управляющие системы, 2003.- № 6. – С. 2–6.

Рассмотрено решение задачи нахождения оптимального алгоритма вторичного поиска (поиска подвижного объекта после потери контакта с ним) на основе теоретико-игрового подхода при различном характере распределения положения объекта в области неопределенности.

Список лит.: 8 назв.

UDK 681.1

The model of secondary search observer's control.

Prokaev A.N. – IUS, 2003. –N 6. – P. 2–6.

This article is devoted to solution secondary search (the mobile object search after losing) optimal algorithm problem on the search games basis with the different law of object position distribution in the uncertainty area.

Refs: 8 titles.

УДК 621.865.8

Методы коррекции пространственных искажений изображений плоских объектов в условиях действия полной аффинной группы преобразований

Соловьев Н.В.– Информационно-управляющие системы, 2003.- № 6. – С. 7–11.

Рассмотрены применяемые в системах распознавания робототехнических комплексов методы коррекции пространственных искажений изображений, описываемых полной аффинной группой преобразований. Основное внимание уделено методам, позволяющим распознавать объекты в процессе компенсации искажений. Предложен метод определения параметров преобразования, основанный на последовательном переборе ограниченного числа характерных точек предварительно центрированного изображения объекта.

Список лит.: 5 назв.

UDK 621.865.8

Methods of a correction of spatial aliasings of images of flat objects in conditions of an operation of full affine group of transformations.

Solov'ev A.N. – IUS, 2003. – N 6. – P. 7–11.

The methods of a correction of spatial aliasings of images are considered. The distortions are circumscribed by full affine group of transformations. The methods allow to recognize objects during indemnification of distortions. Area of application algorithm – robot vision.

Refs: 5 titles.

УДК 681.516.7.015.2

Прогнозирование движения объектов

Коптев Б. А., Розов А. К., Романовский А. Ф. – Информационно-управляющие системы, 2003.- № 6. – С. 12–15.

Фильтрация параметров движущихся объектов может быть осуществлена с использованием аппарата стохастических дифференциальных уравнений. Получаемые в результате решения уравнений оценки параметров позволяют определить прогнозируемое движение объекта. Приводится пример фильтрации и прогноза.

Список лит.: 3 назв.

UDK 681.516.7.015.2

Prognostication movement of objects

Koptev B. A., Rozov A. K., Romanovsky A. F. – IUS, 2003. – N 6. – P. 12–15.

In this article filtering of movement are based on stochastic difference equation. A possibility to applied its using for prognostication movement discuss. The example of applicability is considered.

Refs: 3 titles.

УДК 621(075.8)

Оценка профессиональной пригодности операторов человеко-машинных систем по результатам решения тестовых задач

Dr. Al-Kasasbeh Riad Taha, Городецкий А. Е., Тарасова И. Л. – Информационно-управляющие системы, 2003.- № 6. – С.16–24.

Анализируются методы отбора претендентов на работу в качестве операторов человеко-машинных систем. Предлагаются векторные оценки интеллекта претендентов, содержащие статические и динамические компоненты, характеризующие способности претендентов к решению задач в условиях неопределенности и к самообучению. Исследуются свойства предлагаемых оценок на примере тестирования пяти претендентов.

Список лит.: 8 назв.

UDK 621(075.8)

Test-based estimation of professional aptitude for operators of human-machine systems.

Dr. Al-Kasasbeh Riad Taha, Gorodecky A. E., Tarasova I. L. – IUS, 2003. – N 6. – P. 16–24.

Methods of selection of applicants for work are analyzed as operators of human-machine systems. Vector estimations of intelligence of the applicants, the containing static and dynamic components, describing abilities of applicants to the decision of problems in conditions of uncertainty and to self-training are offered. Properties of offered estimations are investigated by the example of testing five applicants.

Refs: 8 titles.

УДК 519.711

Конструирование интеллектуальных регуляторов
Бураков М. В., Коновалов А. С. – Информационно-управляющие системы, 2003.- № 6. – С. 25–33

В статье рассматриваются современные технологии интеллектуального управления динамическими объектами. Описывается методика синтеза регуляторов с базами знаний, опирающаяся на применение имитационного моделирования и генетического алгоритма. Приводится алгоритм нечеткого моделирования нелинейных динамических объектов. Обосновывается методика конструирования интеллектуальных регуляторов как сообщества конкурирующих интеллектуальных агентов.

Список лит.: 21 назв.

УДК 303.732

Лингво-комбинаторное моделирование плохо формализованных систем
Игнатьев М.Б. – Информационно-управляющие системы, 2003.– № 6. – С. 34–37.

Рассматривается лингво-комбинаторное моделирование плохо формализованных систем, для которых существует лишь описание на естественном языке и которое базируется на использовании ключевых слов, основных понятий, сложившихся в предметной области. Модель состоит из трех групп переменных – характеристик основных понятий, изменения этих характеристик и структурированной неопределенности в эквивалентных уравнениях, которая может быть использована для адаптации и управления. В качестве примеров рассматриваются модели города, организма и атмосферы.

Список лит.: 6 назв.

УДК 62-507

Искусство программирования лифта. Объектно-ориентированное программирование с явным выделением состояний
Наумов Л. А., Шалыто А. А., – Информационно-управляющие системы, 2003.- № 6. – С. 38–49.

Уже около сорока лет одним из примеров, на которых Д. Кнут обучает «Искусству программирования», является задача управления лифтом. На основании невнятного технического задания, используя только словесное описание алгоритмов, он представляет в окончательной форме программу на языке низкого уровня. В предисловии к третьему изданию упомянутой книги приводятся слова Б. Гейтса: «За последние двадцать лет мир изменился». Настоящая работа призвана показать справедливость этого высказывания на примере задачи управления лифтом.

Список лит.: 20 назв.

UDK 519.711

The construction of intellectual controller
Burakov M. V., Konovalov A. S.. – IUS, 2003. –N 6. – P. 25–33

The purpose of this paper is to present a modern technologies of intellectual control of dynamic objects. The technique of synthesis of regulators with knowledge bases basing on application of simulation modeling and genetic algorithm is described. The algorithm of fuzzy modelling of nonlinear dynamic objects is resulted. The technique of designing of intellectual regulators as communities of the competing intellectual agents is proved.

Refs: 21 titles.

UDK 303.732

Linguo-Combinatorial Simulation of the poorly formalized systems
Ignatyev M. B.– IUS, 2003. – N 6. – P. 34–37.

This paper discusses utilization of a combinatorial simulation approach for a complex system modeling.When dealing with complex systems one has to consider that conditions and environment are not fully determined. In the course of this paper it is discussed how a poorly formalized system can be efficiently represented and modeled by combinatorial simulation. Practical applications are demonstrated on the examples of atoms, human organism, city and wheather.

Refs:6 titles.

UDK 62-507

The Art of Lift Programming. Object-oriented Programming with Explicit States Separation
Naumov L. A., Shalyto A. A.– IUS, 2003. – N 6. – P. 38–49.

Near forty years, one of the examples, which was used by D. Knuth for teaching «The Art of Computer programming», is the task about lift control. Basing on indistinct description, using only verbal algorithms definitions, he gives source code of the program on the low-level language. In the preface to the third edition of mentioned book there are words of B. Gates: «During last twenty years the world has changed». The purpose of the present article is to show the correctness of last statement on the example of the task about lift control

Refs: 20 titles.

УДК 531.383-1:537.2

Программное обеспечение системы управления вентильным индукторно-реактивным двигателем

Голландцев Ю. А., – Информационно-управляющие системы, 2003.- № 6. – С. 50–53

Рассматриваются особенности построения и реализации аппаратной и программной части микропроцессорной системы управления вентильным индукторно-реактивным двигателем.

Список лит.: 4 назв.

UDK 531.383-1:537.2

Software for the control system of switched reluctance motors

Gollandcev Yu. A. – IUS, 2003. – N 6. – P. 50–53.

Hardware and software design and implementation for the microprocessor control system of switched reluctance motors are considered.

Refs: 4 titles.

УДК 681.391.1

Особенности использования булевых функций для организации криптографических преобразований потоковой информации

Бубликов А. Б., Ерош И. Л., Сергеев М. Б. – Информационно-управляющие системы, 2003. – № 6. – С.54–57.

Рассмотрены характеристики булевых преобразований при использовании для защиты потоковой информации. Приведены сравнительные результаты исследования программной и аппаратной реализации, отмечены преимущества и особенности рассматриваемых преобразований.

Список лит.: 6 назв.

UDK 681.391.1

Application of Boolean functions for the cryptographic transformation of stream information.

Bublikov A. B., Erosh I. L., Sergeev M. B. – IUS, 2003. – N 6. – P. 54–57.

Characteristics of the boolean transformations for protection stream information are considered. Comparative results of research of program and hardware realization are presented. Advantages and features of considered transformations are emphasized.

Refs: 6 titles.

УДК 681.3.06: 616.089.5

Перспективы применения в авиации интегрированных нашлемных систем нейрофизиологического контроля

Шепета А. П., Жаринов И. О. – Информационно-управляющие системы, 2003.- № 6. – С. 58–62.

Настоящая статья направлена на решение актуальной задачи организации и обеспечения безопасности (надежности) полетов пилотируемых летательных аппаратов. В работе рассматривается эффективный нейрофизиологический подход к комплексной оценке физиологического состояния пилота в динамике полета летательного аппарата в реальном масштабе времени.

Список лит.: 5 назв.

UDK 681.3.06: 616.089.5

Application of helmet-mounted system with neirophysiology control in aviation equipment.

Shepeta A. P., Zharinov I. O. – IUS, 2003. – N 6. – P. 58–62.

This article is directed at the solution of an actual problem of flights organization and safety control (reliability). In activity the effective physiological approach to complex estimation of pilot physiological condition in flight dynamics in real time is esteemed.

Refs: 5 titles.

СОДЕРЖАНИЕ ЖУРНАЛА «ИНФОРМАЦИОННО – УПРАВЛЯЮЩИЕ СИСТЕМЫ» за 2003 г. [№ 1 – 6]