ХАРЬКОВСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

На правах рукописи

Чепцов Михаил Николаевич

УДК 656.25:656.256:656.257

ДИНАМИЧЕСКАЯ ПОЕЗДНАЯ МОДЕЛЬ РАЙОНА ДИСПЕТЧЕРСКОГО УПРАВЛЕНИЯ

05.22.20 - Эксплуатация и ремонт средств транспорта

Диссертация на соискание научной степени кандидата технических наук

Научный руководитель:
кандидат технических наук,
доцент
Мойсеенко Валентин Иванович

Харьков- 2001

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Железнодорожный транспорт Украины, представляющий собой сложную территориально рассредоточенную систему технологических подразделений и технических средств, должен обеспечивать перевозку пассажиров и грузов с максимально возможной производительностью, с минимальной себестоимостью и гарантированной безопасностью движения.

В условиях выхода из кризисных явлений и дальнейшего наращивания экономического потенциала растет объем перевозок, все более усложняется организация перевозочного процесса. Это требует постоянного совершенствования системы управления железнодорожным транспортом, улучшения организации взаимодействия технологических подразделений и их технической оснащенности. Так, по выражению Петрова А. П. [48]: «Динамичность перевозочного процесса определяет ситуацию на полигонах сети, что требует оперативного принятия решений по управлению этим процессом. Система управления, постоянное ее совершенствование должно обеспечить высокий уровень использования технической вооруженности транспорта...».

Кроме того, в работе Грунтова П. С. [99], при формулировании принципа комплексного подхода к управлению, технологии и развитию транспортных систем, автор отмечает, что обеспечение комплексного развития всех отраслей железнодорожного транспорта возможно только на базе достижений научно-технического прогресса, внедрения в производство передовых технологий, широкого применения средств вычислительной техники, повышения уровня автоматизации всех составляющих технологического процесса.

В этой работе и в трудах Кочнева Ф. П., Сотникова И. Б. отмечалось, что в системе управления железнодорожным транспортом следует особо выделить систему оперативного управления перевозочным процессом. Так, по определению, данному в [53]: «Оперативное управление является основной функцией управления перевозочным процессом...». Реализация оперативного

управления на железнодорожном транспорте осуществляется через систему диспетчерского руководства, которая выполняет системный контроль выполнения оперативных планов и реализует ход перевозочного процесса по участкам и станциям. Естественно, что автоматизация диспетчерского управления - приоритетная задача для обеспечения дальнейшего развития железнодорожного транспорта Украины.

В то же время, методы автоматизации, математические модели, программное и аппаратное обеспечение, разрабатывалось в 70-80 годах и, в значительной мере было ориентировано на существующую техническую оснащенность. Однако за истекший период значительно возрос уровень развития техники, в первую очередь вычислительной, появились и получили дальнейшее развитие, новые математические методы. В связи с этим, становится очевидным, что повышение уровня автоматизации оперативного управления движением поездов, возможно за счет применения современных технических средств. А это, в свою очередь вызывает необходимость создания моделей, в основу построения которых положены прогрессивные математические методы, в частности, нейросетевого моделирования.

Актуальность работы подчеркивается наличием ***связи с научными программами, планами, темами.*** В первую очередь это Концепция и программа реструктуризации на железнодорожном транспорте Украины, принятая Государственной администрацией железнодорожного транспорта в 1998 году [49]. Кроме того, в соответствии с постановлением Кабинета Министров Украины №821 от 04.08.97 “Про затвердження Концепції створення й функціонування національної мережі транспортних коридорів в Україні”, автор принимал участие в научно-исследовательской работе “Розробка автоматизованої системи знімання інформації про проходження поїздів по міждержавних і міждорожних стикових пунктах” [84], а также научно-исследовательской теме “Разработка автоматических систем оповещения работающих на пути” государственный регистрационный №019717003551

Цель и задачи исследования. Цель исследования - разработка динамической поездной модели района диспетчерского управления для совершенствования автоматизированной системы отслеживания дислокации подвижных единиц на элементах путевого развития. Для достижения поставленной цели необходимо решить ряд задач:

1. Выполнить анализ структуры, технических средств и информационного обеспечения системы оперативного управления движением поездов на железнодорожном транспорте;
2. Разработать общую динамическую поездную модель района диспетчерского управления, потому что существующие не в достаточной мере обеспечивают соответствие информационной модели состоянию перевозочного процесса;
3. Разработать модель путевого развития полигона диспетчерского управления, адаптированную под функциональность общей модели;
4. Обосновать выбор математического аппарата нейронных сетей и разработать принципы построения моделей функционирования станционных устройств автоматики;
5. Разработать модель функционирования путевого участка, по информации системы диспетчерской централизации;
6. Составить алгоритмы и программную реализацию моделей на основе современных аппаратных средств и языков программирования;

Методы исследований.

В работе, при синтезе общей динамической модели и модели путевого развития района диспетчерского управления, применялись методы векторной алгебры. При моделировании работы устройств автоматики - теория конечных автоматов, методы нейросетевого моделирования. Синтез модели функционирования путевого участка осуществлялся с применением как широко распространенных математических методов, в частности интегрального исчисления, разностных уравнений, так и нейронных сетей (обратного распространения ошибки).

Научная новизна полученных результатов.

1) Впервые общая динамическая модель и модель путевого развития района диспетчерского управления, синтезирована с применением методов векторной алгебры и адаптацией под функционирование в реляционной базе данных.

2) Впервые показана возможность применения нейронных сетей для моделирования работы устройств автоматики на станции и сформулированы принципы построения нейросетевых моделей при маршрутном управлении.

3) Впервые исследован временной признак функционирования путевого участка по информации системы диспетчерской централизации во взаимодействии с графиком движения поездов.

4) Впервые синтезирована нейросетевая модель функционирования путевого участка и разработана методика ее обучения по данным графика движения поездов и информации системы АСОУП.

5) Впервые создано программное обеспечение для реализации нейросетевой модели работы путевых участков.

Обоснованность и достоверность научных положений, выводов и рекомендаций обусловлена корректностью постановки и решения задач, обоснованностью выбора математического аппарата, и использованием при моделировании реальных исходных данных. Достоверность научных положений подтверждается наличием актов о внедрении основных положений работы.

Практическое значение полученных результатов состоит в том, что на основе разработанных моделей создано соответствующее программное обеспечение и рассмотрена возможность аппаратной реализации. Основные положения работы нашли применение в производстве, что подтверждается актами о внедрении.

Личный вклад соискателя.

В работе [68] лично предложена модель функционирования станционных устройств автоматики на базе теории конечных автоматов, рассмотрены ее достоинства и недостатки, показана возможность применения нейронной сети, сформулированы принципы нейросетевого моделирования маршрутного управления станционными объектами.

В работе [69] лично произведен анализ состояния путевых датчиков, показана возможность применения нейронной сети для классификации состояния путевого участка, получены и проанализированы результаты моделирования.

В работе [70] лично предложен метод моделирования поездной ситуации на стации.

Апробация результатов диссертации.

Основные положения и результаты диссертационной работы докладывались на 4-м и 5-м Международном молодежном форуме «Радиоэлектроника и молодежь в XXI веке», проводившимся в Харьковском Государственном техническом университете радиоэлектроники в 2000 и 2001г., Алуштинской 1998, 2000 г. международной конференции, научно-технических конференциях и семинарах ХарГАЖТ 1998-2000г.

Публикации.

По содержанию диссертации опубликовано пять статей в специализированных научных и научно-технических изданиях [68-70, 106, 107].

РАЗДЕЛ 1
АНАЛИЗ СТРУКТУРЫ, ТЕХНИЧЕСКИХ СРЕДСТВ И ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ СИСТЕМЫ ОПЕРАТИВНОГО УПРАВЛЕНИЯ ДВИЖЕНИЕМ ПОЕЗДОВ

ЕЕ Структура системы оперативного управления движением поездов

В основу организации движения на железных дорогах Украины положены принципы диспетчерского управления (ДУ), осуществляемого на всех уровнях - от Укрзализницы до участка или станции [99]. В настоящий момент на дорогах Украины действует как четырехуровневая, так и трехуровневая системы управления железнодорожным транспортом (Рис. 1.1).

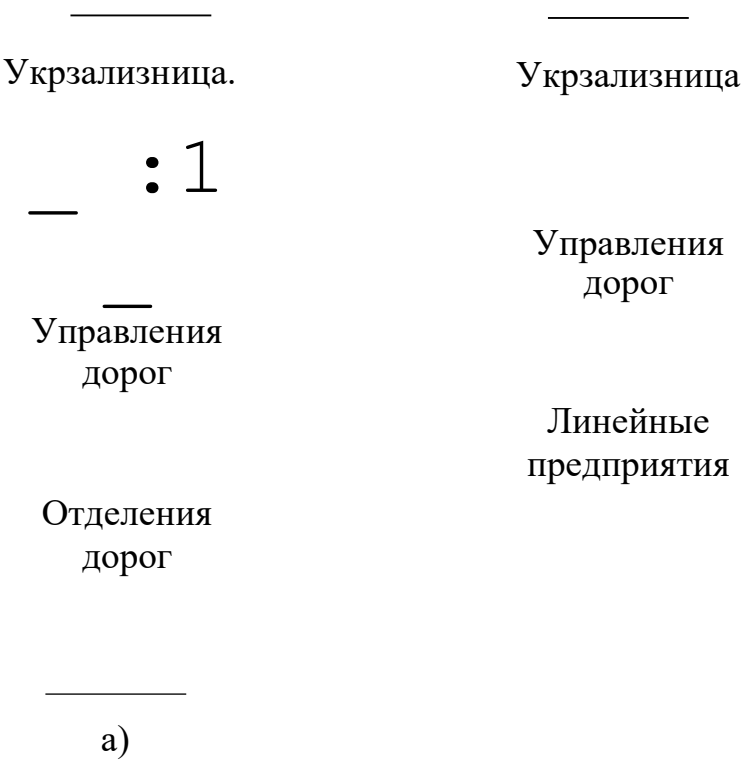


Рис. 1.1 Структуры систем управления железнодорожным транспортом: а) четырехуровневая; б) трехуровневая

Независимо от применяемой на дороге модели управления в качестве элементов системы следует выделить: а) станции; б) узлы, диспетчерские участки; в) районы управления (отделения).

и

На уровне Укрзализницы, в главном управлении движения, координация работы дорог находится в ведении дежурных помощников начальника распорядительного отдела, отвечающих за текущую работу железнодорожного транспорта Украины.

Организацией движения поездов в масштабе дороги занимается начальник оперативно-распорядительного отдела службы движения (ДГ), его сменный заместитель (ДГС), дежурный помощник по поездной (ДГП) и местной (ДГПВ) работе, дежурный локомотивный диспетчер дороги (ТДЦ). Они контролируют работу отделений и районов управления, решают вопросы, возникающие на стыках между отделениями. Основными задачами оперативного персонала управления дороги является обеспечение движения поездов в соответствии с планом, формирование составов на сортировочных и участковых станциях, рациональное использование локомотивов.

На уровне отделения дороги или района управления организацией движения занимаются:

- старший диспетчер отдела движения (ДНЦС);
- дежурный по отделению (ДНЦО);
- узловой диспетчер (ДНЦУ);
- поездной (ДНЦ) и узловой (ДНЦУ) диспетчер;
- диспетчер вагоно-распорядитель (ДНЦВ);
- локомотивный диспетчер (ТНЦ).

Поездной и маневровой работой на станциях руководят дежурные по станциям (ДСП). На крупных станциях формированием и расформированием поездов, подачей вагонов под погрузку и разгрузку, внутристанционными операциями руководят станционные и маневровые диспетчеры (ДСЦ).

- арис. 1.2 показаны управляющие связи звеньев основного оперативного персонала службы движения, непосредственно обеспечивающих движение поездов.

Как в четырехуровневой и трехуровневой моделях они примерно идентичны. Различие состоит в том, что управление перевозочным процессом при трехуровневой системе управления группируется не по отделениям дороги, а по направлениям и районам управления более логично объединяющих существующие поездопотоки и регионы зарождения и (или) погашения грузопотоков.

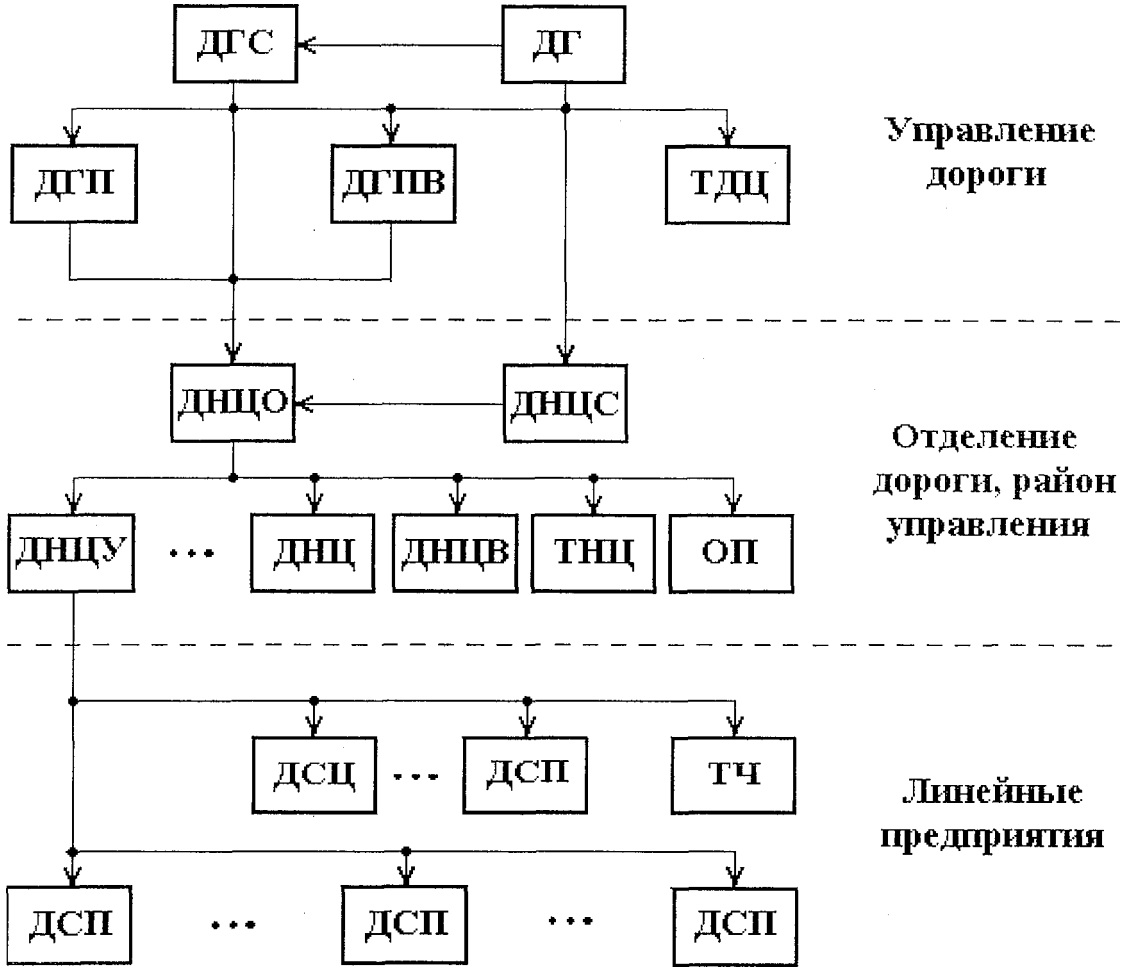


Рис. 1.2. Управляющие связи звеньев оперативного персонала службы движения

Таким образом, система диспетчерского управления имеет иерархическую структуру, а значит ей присущи следующие основные особенности [76]:

- последовательная вертикальная соподчиненность;
- правом вмешательства в действие любой системы обладает только система вышестоящего уровня;

- действия любой системы фактически зависят от исполнения своих функций управляемыми подсистемами;
- более высокие уровни управления имеют дело с более крупными подсистемами и более широкими аспектами поведения всей системы в целом, т.е. чем выше уровень, тем больше общность отображения объекта управления;
- более высокий уровень управления имеет больший период выработки управляющего решения из-за необходимости сбора и обработки большего количества разнородной информации;
- чем выше уровень управления, тем больше неопределенность в выработке математически точной программы управления объектом.

Из этого следует, что для эффективного функционирования системы диспетчерского управления необходима хорошо отлаженная технология информационного обмена между уровнями, позволяющая достаточно оперативно и полно отражать управляемый объект. А так как основной функцией системы ДУ является управление движением поездов, то необходимость в отражении достоверной информации о поездном положении становится очевидной.

В то же время, в сложной иерархической системе, каковой является оперативное управление движением поездов, это сделать трудно и поэтому все недостатки в информационном обмене оборачиваются потерей эффективности управления. Таким образом, для автоматизации управления необходимы технические средства, отражающие состояние станций, участка по всем технологическим составляющим. Поэтому, прежде всего, необходимо рассмотреть современный уровень технический оснащенности и информационного обеспечения системы диспетчерского управления на дорогах Украины.

1.2. Техническая оснащенность и информационное обеспечение системы диспетчерского управления

Оперативное управление эксплуатационной работой осуществляется с применением комплекса технических средств, которые подразделяются на три вида: 1) средства связи; 2) устройства диспетчерской централизации; 3) информационные автоматизированные системы.

В состав технического обеспечения работников диспетчерского аппарата входят следующие средства связи:

- телефонная связь (в некоторых случаях радиотелефоны);
- дорожно-распорядительная связь (ДРС);
- поездная радиосвязь (ПРС).

Диспетчерские участки и узлы оборудованы средствами диспетчерского контроля (ДК) или диспетчерской централизации (ДЦ) типа “Нева”, “Луч”, “Минск”, в настоящее время внедряются микропроцессорные (МП) ДЦ.

Канал телесигнализации (ТС) в системе диспетчерской централизации предназначен для получения информации о состоянии объектов автоматики и телемеханики на станциях и перегонах полигона диспетчерского управления, ее передачи в центр ДУ, отражения этой информации для восприятия диспетчером (рис. 1.3).

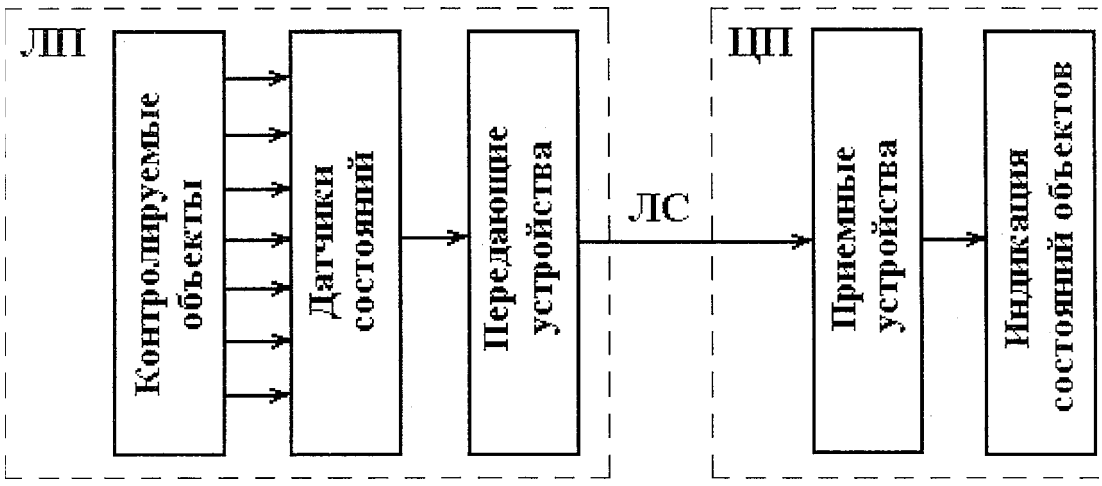
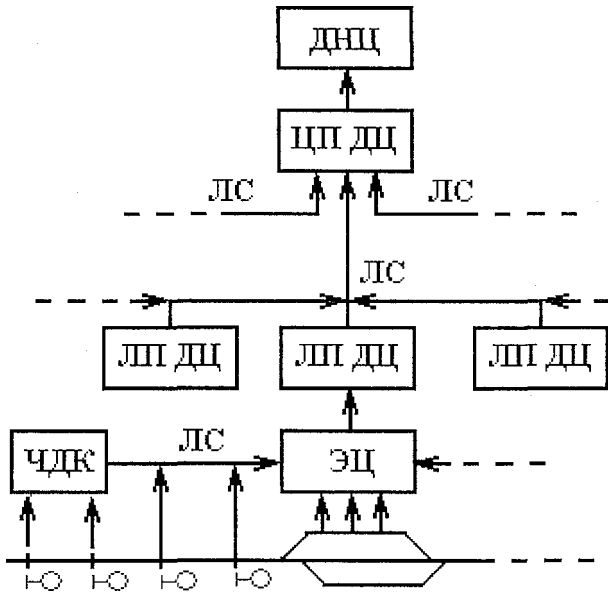


Рис. 1.3 Структурная схема канала телесигнализации системы ДЦ

Контролируемые объекты - непосредственно связанные с движением поездов станционные и перегонные устройства линейных пунктов (ЛП) ДЦ: участки пути, светофоры, а в МП ДЦ стрелки, и ряд других устройств. В качестве датчиков состояний применяются устройства электрической централизации (ЭЦ) на станциях и системы интервального регулирования на перегонах. Для контролируемых объектов первичными датчиками являются:

- рельсовые цепи (участки пути);
- сигнальные реле (светофоры);
- контакты автопереключателей стрелочных приводов (стрелки);
- устройства электрической- централизации (маршруты, установленное направление движения).

Информация, снимаемая с этих датчиков, кодируется, модулируется и передается в линию связи. В центре ДУ (центральный пост ДЦ) осуществляется прием, демодуляция, декодирование и отображение на пульте диспетчера. Система ДЦ территориально рассредоточена, устройствами телесигнализации охвачен весь полигон диспетчерского управления (рис. 1.4).



Перегон (АБ) (ітянтпті (АТІ)

Рис. 1.4 Структура устройств телесигнализации системы ДЦ

Основную долю информации о поездном положении на полигоне диспетчерского управления дают каналы телесигнализации систем диспетчерской централизации и диспетчерского контроля. Это данные о состоянии путевых участков, светофоров, положении стрелок, установленных маршрутах, направлении движения. В системах ДЦ “Нева”, “Луч”, “Минск”, которыми оборудовано большинство дорог Украины, эта информация является дискретной и поступает в центр ДУ в двоичном виде:

- путевой участок: «свободен», «занят»;
- светофор: «открыт», «закрыт»;
- маршрут: «установлен», «не установлен»;
- установленное направление движения: «четное», «нечетное».

Принцип работы устройств телесигнализации построен на циклическом опросе состояния контролируемых объектов на станциях полигона. В центре диспетчерского управления информация обновляется периодически, после приема очередного сигнала ТС, длительность каждого цикла около 5 секунд.

В микропроцессорных системах, в отличие от ДЦ «Нева», «Луч», «Минск», применяется спорадический способ формирования сигнала ТС, информация в системе обновляется при изменении состояния хотя бы одного контролируемого объекта. Кроме того, значительно увеличилась значность состояний по каждому объекту, и количество контролируемых объектов автоматики. Например, в «КСУ ДП», которая разработана государственным предприятием «ЦСМ АСУ УПП ЖТ» [67], состояние объектов может принимать одно из следующих значений:

- стрелки и съезды:

О - неопределенное положение;

1 - плюсовое положение;

2 - минусовое положение;

3 - потеря контроля после неопределенного положения;

4 - потеря контроля после плюсового положения;

5 - потеря контроля после минусового положения;

- путевые участки, приемоотправочные пути станций, блок-участки при автоблокировке:

1 - свободен;

2 - свободен, замкнут в маршруте;

3 - занят;

4 - свободен, исскуственное размыкание маршрута;

5 - занят, исскуственное размыкание маршрута;

6 - неисправность сигнальной установки;

- поездные и маневровые светофоры

О - неисправность;

1 - запрещающее показание;

2 - основное разрешающее (зеленый, белый);

3 - дополнительное разрешающее (белый, 2 белых);

4 - пригласительный поездного;

Однако в указанных системах ДЦ нет устройств идентификации подвижного состава, следовательно, и нет данных о конкретной подвижной единице, находящейся на элементе путевого развития, а при оперативном управлении движением поездов эта информация необходима в первую очередь.

Кроме систем ДЦ центры диспетчерского управления оборудуются техническими средствами для доступа к автоматизированной системе оперативного управления движением поездов (АСОУП) [96].

Система АСОУП предназначена для автоматизации выполнения диспетчерским аппаратом таких функций, как планирование пропуска поездов по участку, организации регулирования движения поездов на участке, оценки и прогноза поездного положения на станциях и перегонах участка, планирования продвижения местных поездов на участках, организации работы локомотивов, контроля работы станций участка по выполнению графика движения поездов. В структуре системы АСОУП (рис. 1.5) явно выражены несколько уровней: а) линейного предприятия; б) узлового вычислительного центра (УВЦ); в)

дорожного вычислительного центра (ДВЦ); г) главного вычислительного центра (ГВЦ).

На уровне линейных предприятий станции оборудованы абонентскими пунктами (АП) ЕС ЭВМ, где операторами вводится информация в виде сообщений системы. Сортировочные станции оборудуются мультиплексором передачи данных (МПД). Информация с уровня линейных предприятий по линиям связи поступает в узловые вычислительные центры, далее в дорожные ВЦ и затем в ГВЦ.

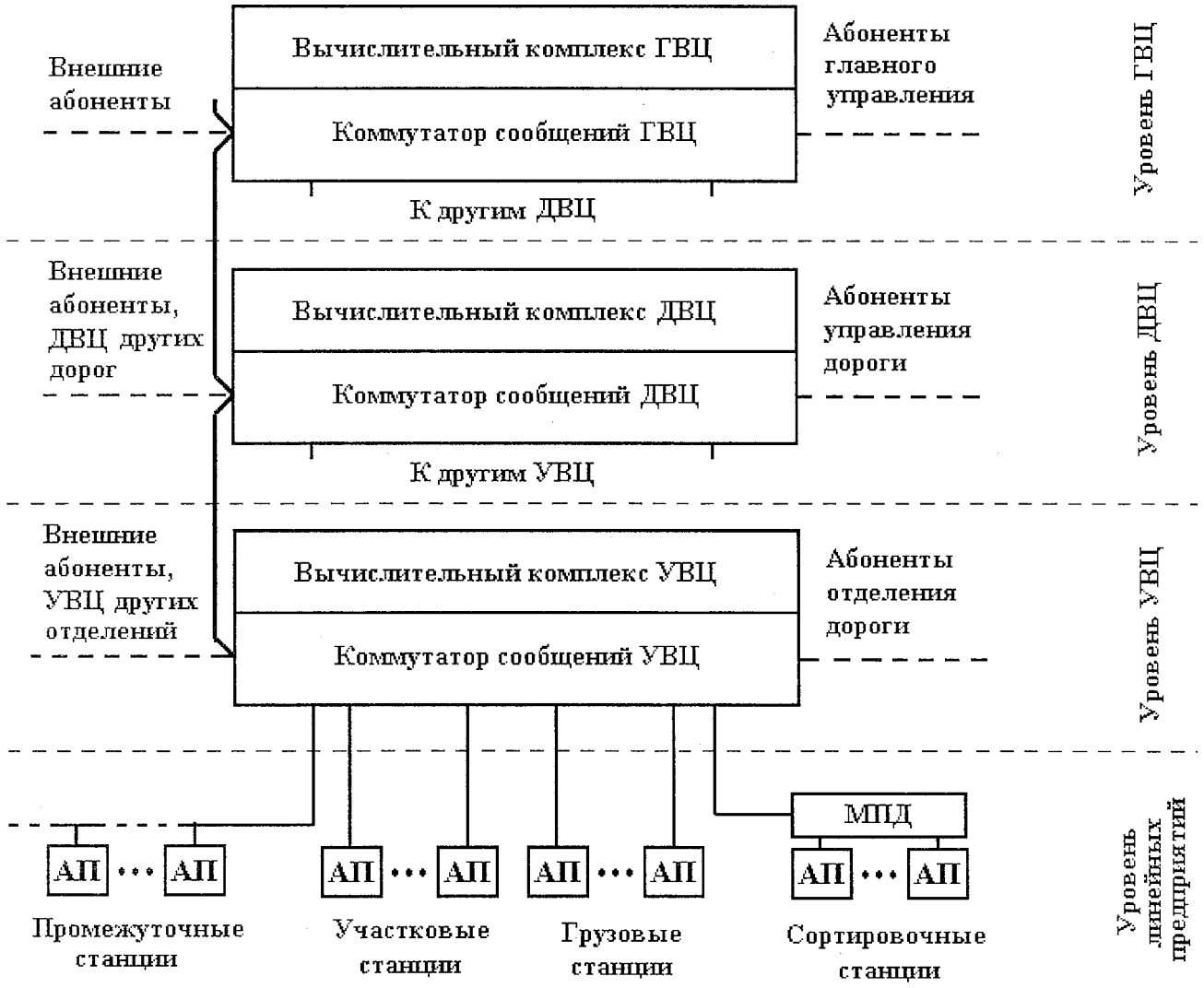


Рис. 1.5 Структура системы АСОУП

Информация о поездной работе в системе АСОУП включает вводимую в ЭВМ постоянную информацию (нормативно-справочные данные) и оперативные сообщения об операциях с поездами и вагонами, а также сообщения по

планированию прибытия и отправления поездов [109]. Нормативно-справочные данные оформляются в виде таблиц и вводятся в систему работниками дорожного информационно-вычислительного центра (ИВЦ). К оперативным сообщениям АСОУП относятся:

- телеграммы-натурные листы (ТНЛ) поезда;
- сообщения о прицепляемых и отцепляемых на станциях вагонах;
- сдача поездов по стыковым пунктам;
- прибытие с участков и отправление поездов с технических станций;
- данные о загруженных и выгруженных на станции вагонах, и ряд другой информации.

Наряду со значительным увеличением уровня автоматизации оперативного управления в системе АСОУП присутствует ручной ввод первичных данных, вследствие этого ухудшается достоверность информации, и ее оперативность. Кроме того, опыт эксплуатации системы АСОУП показал, что оперативный персонал не использует в полном объеме возможности данной системы [109].

Таким образом, анализ технической оснащенности системы оперативного управления показывает, что во первых, система АСОУП предоставляет диспетчерскому аппарату информацию о поездном положении только в дискретные периоды времени, по мере формирования поездов, при подходе к раздельному пункту, после прицепки и отцепки вагонов и т. п. Однако данные о поездной ситуации на полигоне необходимы непрерывно, так как потребность в них может возникнуть в случайный момент времени. Во вторых, информация системы ДЦ отображается для визуального восприятия диспетчером на пульте-табло или экране дисплея. Для того чтобы сопоставить индикацию о занятии участка пути с конкретной подвижной единицей диспетчеру необходимо или постоянно отслеживать дислокацию каждого поезда, с момента его зарождения на станции формирования или же воспользоваться средствами связи для получения информации со станции. Естественно, что необходимость в автоматизации этого процесса очевидна, но с другой стороны, эта задача достаточно сложная, и поэтому целесообразно рассмотреть существующие

подходы к формальному описанию диспетчерского управления как информационной системы, с целью выявления наиболее существенных параметров, влияющих на качество руководства перевозочным процессом.

1.3. Анализ информационной модели системы диспетчерского управления

Характерной чертой деятельности персонала на всех уровнях оперативного управления процессом перевозок является то, что не существует непосредственной связи с объектом управления в силу его территориальной рассредоточенности. В такой ситуации для принятия управляющих решений оперативным работникам необходимо предоставить соответствующее информационное обеспечение. Естественно, что от качества информации, ее полноты, своевременности, приспособленности к восприятию зависит уровень руководства перевозочным процессом [24].

Формальное описание и подходы к синтезу модели систем управления широко освещены в литературе (например [3, 6, И, 15-18, 31, 36, 58, 103, 104]). Рассмотрим информационную модель диспетчерского управления на основе подхода предлагаемого в работе [80].

Процесс диспетчерского управления можно представить в виде информационного цикла, включающего следующие фазы обращения информации: сбор и передача, представление человеку, обработка и хранение, принятие решения и воздействие. В общем случае систему ДУ можно представить в виде схемы (Рис. 1.6).

Информация о состоянии объекта (X) и управляющего воздействия (Y) поступает на датчики £\ и D_vсоответственно. Результат работы датчиков:

$$x., = \text{В,Ив Г,,=О,(Г)} \qquad (1.1)$$

- является входной информацией для системы диспетчерского управления. На основе этой информации система ДУ вырабатывает управляющее воздействие (U), поступающее на исполнительный механизм (ИМ) и далее на объект.

Таким образом в основе процесса управления лежит информация о сложившейся ситуации [80]:

$$l = (X_d, Y_d Y \qquad (1.2)$$

Так как основной задачей оперативного управления эксплуатационной работой является обеспечение движения поездов, то главная роль в информационном обеспечении принадлежит средствам информации о поездном положении - дислокации поездов, локомотивов и локомотивных бригад на полигоне управления. Поэтому, прежде всего, необходимо рассмотреть источники такого рода информации (У, *Y* в 1.1), и ее содержание при поступлении в систему диспетчерского управления (A'(/,Y_d в 1.2).

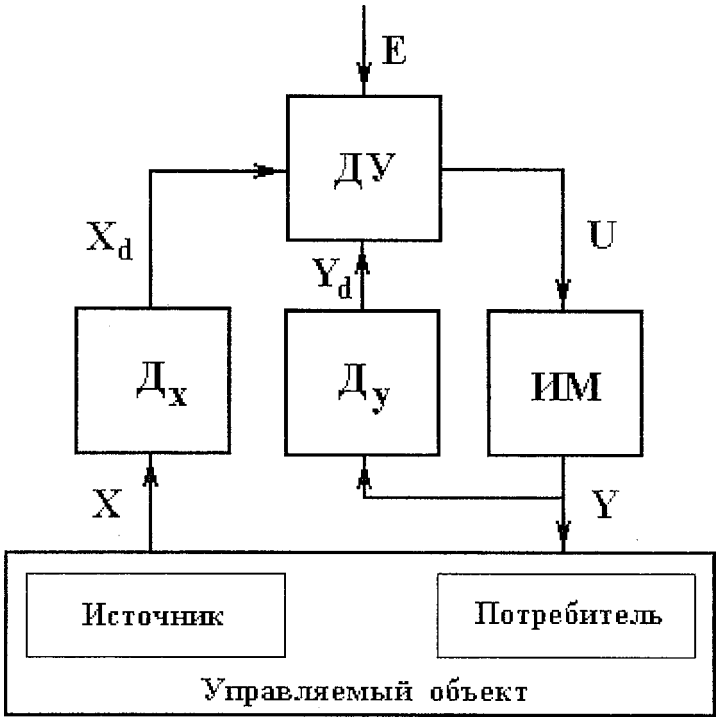


Рис. 1.6 Общая схема системы диспетчерского управления

Как показано в п.1.1, источниками информации о сложившейся поездной ситуации являются системы АСОУП, и ДЦ. Тогда, в выражении 1.2:

X_d - данные о номерах поездов на полигоне, их составе (АСОУП), контроль занятия, освобождения участков пути (ДЦ);

Г, - контроль установки маршрутов, открытия светофоров (ДИ), контроль выполнения технических операций (АСОУП).

Информация, получаемая от датчиков, по определению любой системы управления, всегда является неполной, что доказано в работе [80]. Эта неполнота, прежде всего, связана с ограниченными возможностями всякой системы сбора информации, кроме того, в ней могут присутствовать искажения (Е на рис. 1.6), источники возникновения которых следует рассмотреть более детально.

Так, любые сигналы телемеханических систем состоят из той или иной совокупности импульсов, передаваемых по каналу связи [76]. Это утверждение справедливо как для системы диспетчерской централизации (рис. 1.4) так и АСОУП (рис. 1.5). Правильное опознание сигналов на приемной стороне означает верный (достоверный) прием переданного сообщения. Это возможно в том случае, если принимаемые импульсы искажены не настолько, чтобы приемное устройство не различало импульсных признаков. В противном случае исходная информация принимается с искажениями. Такие искажения в канале связи чаще всего являются результатом наложения внешних помех или определяются физическими свойствами канала передачи данных. Вопросы возникновения искажений в каналах передачи данных достаточно хорошо освещены в литературе (например [76], [93], [109]).

Кроме линии связи искажения информации могут возникать в результате сбоев в работе источника первичной информации, устройств формирования сигнала, его передачи, приемных устройств, средств отображения информации и в результате неверного восприятия оператором. В общем случае воздействие искажений информации на принятие решений в системе управления (рис. 1.4) выражается в виде суммы контролируемых искажений $E_к$ и неконтролируемых £,, [80]:

$$£ = £,+£,,, \qquad (1.3)$$

где под $E_к$ подразумеваются искажения сигнала, которые можно выявить и исправить аппаратным или программным способом. Относительно £,, могут выдвигаться лишь определенные предположения, но непосредственно они не выявляются. Естественно, что чем больше значение $E_к$ в суммарном значении £,

тем больше возможностей в интерпретации входной информации алгоритмом работы системы управления. В идеальном случае $E = E_к$, когда все возможные искажения сигнала выявляются системой управления.

Практика эксплуатации системы АСОУП показала, что наибольшее количество искажений информации происходит во время ввода первичных данных оператором. В связи с этим был проведен ряд организационных мероприятий, по повышению качества ввода в систему первичных документов, разработана и реализована система программного контроля достоверности поступающей информации. Программа ее обработки позволяет проверить зависимость исходных сообщений друг от друга, проконтролировать формат передаваемых сообщений, а также логически сравнить отдельные реквизиты сообщений. Таким образом достоверность данных в АСОУП доведена до приемлемого уровня (99-99,8%) [109].

В системе диспетчерской централизации параметр $E_к$ в выражении 1.3 (контролируемые искажения) - это искажения, возникающие в процессе прохождения информации внутри системы. Для выявления таких искажений в устройствах ДЦ применяются специальные меры [76], в результате чего их идентификация и обработка не является сложной задачей. Основные из таких искажений в системах “Нева”, “Луч”, “Минск” - пропадание в текущем сигнале информации об объекте контроля. За счет циклического способа формирования и передачи информации, такого рода искажения исправляются в следующем цикле работы системы. Естественно, такой способ коррекции приводит к нарастанию временной ошибки, чего нельзя не учитывать. Следует отметить, что в микропроцессорных системах ДЦ такие явления отсутствуют. Большинство искажений в МП ДЦ исправляются программно-аппаратным способом [67]. Можно утверждать, что значение контролируемых искажений $E_к$ в микропроцессорных системах диспетчерской централизации имеет приемлемый уровень.

Неконтролируемые искажения $E_п$ (1.3) в системах ДЦ главным образом связаны с поступлением первичной информации. В ее формировании принимает

участие значительное количество устройств, в каждом из которых не исключена возможность возникновения сбоев. Так, например, рассмотрим прохождение информации о занятии поездом блок-участка на перегоне (рис. 1.7). Вступление первой колесной пары поезда на изолированную рельсовую цепь блок-участка приводит к ее шунтированию, после чего обесточивается реле путевого датчика.

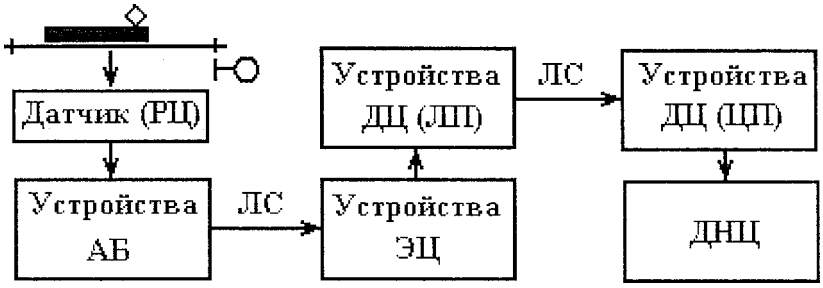


Рисунок 1.7 Прохождение информации о занятии блок-участка

После обработки этого сигнала релейной схемой устройств автоблокировки, происходит передача информации в линию связи. Станционные устройств автоматики осуществляют прием и дешифрацию этих данных. Далее в устройствах линейного поста системы ДЦ происходит формирование и передача сигнала в линейную цепь на центральный пост. После приема, дешифрации, отображения на пульте информация о занятии поездом блок-участка воспринимается диспетчером.

Известно, что каждое из перечисленных устройств автоматики имеет не нулевую вероятность возникновения искажений. По аналогии с подходом применяемым в [80], для рассматриваемого примера неконтролируемые искажения информации о состоянии блок-участка можно выразить как сумму искажений в устройствах АТ, которые задействованы в прохождении информации от первичного датчика до ее потребителя:

$$E_{п!'ц} \sim E/Ч1 + E.iii + E.нц.чп + E.'ic + Едцц_П + E_{,(Н/1}, \quad (1.4)$$

где, $E_{I>ц}$ - искажения, вносимые датчиком путевого участка;

E — искажения, возникающие в сигнальной точке автоблокировки (устройства ЧДК);

1 .щлп - искажения информации при ее формировании на линейном пункте ДЦ;

1.1 (. - искажения, вносимые линией связи;

E:ЦЩIП - искажения информации при приеме и дешифрации на центральном пункте ДЦ;

E ит - искажения при восприятии информации диспетчером.

Уровень неконтролируемых искажений информации о поездном положении на полигоне диспетчерского управления, будет зависеть от количества устройств, принимающих участие в формировании такого рода информации, и от уровня искажений, возникающих в каждом устройстве.

Таким образом, при синтезе динамической поездной модели района диспетчерского управления, особое внимание следует уделить возможным искажениям первичной информации, так как эта проблема в настоящее время не решена.

1.4. Постановка задачи исследования

Создание автоматизированной системы управления железнодорожным транспортом предполагает реализацию динамической модели, соответствующей реальному перевозочному процессу на полигоне дороги во всех его существенных признаках. Как показано в работе Грунтова П. С. [24], такая модель должна представлять собой периодически обновляемые данные о поездном положении, локомотивах, вагонах, контейнерах и грузах, находящихся на обслуживаемом полигоне. Традиционное решение этой проблемы предусматривает использование информации, получаемой от системы диспетчерской централизации (ДЦ), оперативной телефонной связи, в сочетании с фиксацией данных на графике, и в виде приложения к графику исполненного движения. Однако, при таком подходе, большая часть рабочего времени диспетчера тратится на получение и обработку информации, а не на оперативное вмешательство в управление эксплуатационной работой. Так, например, анализ

показал, что большая часть рабочего времени ДГП (около 56%) тратится на получение информации и оформление сокращенного графика движения поездов, а на регулирование вагонными и локомотивными парками только 4,5%, на оперативное вмешательство в управление работой станций и отделений всего 0,2 -9,0% [24].

Кроме того, в работах [24-27], [53], [99] неоднократно указывалось, что модель перевозочного процесса является одним из основных компонентов информационного обеспечения в задачах оптимизации управления эксплуатационной деятельностью железнодорожного транспорта, и реализация ее только традиционными методами, без использования возможностей современных технических средств, не эффективна.

К этому следует добавить, что при решении проблемы создания высокоэффективных технологических систем на железнодорожном транспорте особое значение приобретает развитие теоретических и прикладных исследований, с пересмотром, а зачастую и отказом от традиционных подходов, концепций, методологий [71].

Так, повышение эффективности оперативного управления - одна из приоритетных задач развития техники и технологий на железнодорожном транспорте в мировой практике. На железных дорогах США продолжаются исследования систем управления движением поездов, основанных на применении современных технологий спутниковой связи. При этом ставится задача найти экономичное техническое решение, способное обеспечить повышение безопасности перевозочного процесса и способствовать росту производительности железных дорог [142].

Железнодорожные компании Conrail, CSX и Norfolk Southern (NS) в 1996 г. начали сотрудничать в области систем управления движением поездов на основе радиосвязи (CBTC), считая существовавшие в то время системы CBTC недостаточно эффективными. Разрабатываемая система позволит получить значительный экономический эффект за счет более точного скрещения поездов. Кроме того, GE Harris Railway Electronics рассчитывает, что благодаря

применению СВТС железные дороги смогут увеличить пропускную способность на 20 - 30 % без строительства дополнительной путевой инфраструктуры и отказаться от значительной части напольных устройств СЦБ [142].

На железной дороге Саутерн (США) с 1982 года эксплуатируется система автоматизированного оперативного планирования поездной работы на базе микро-ЭВМ (CAD). Основные задачи, решаемые системой - повышение точности выполнения графика движения и снижение эксплуатационных затрат [24]. Дорога использует CAD как дополнение к традиционной системе диспетчерской централизации.

Большое значение автоматизации диспетчерского управления придается при модернизации и строительстве новых линий на железных дорогах Великобритании, Франции и Италии. Так в 1983 году компанией Ансальдо для Итальянских государственных железных дорог была разработана типовая автоматизированная система диспетчерской централизации на базе микро-ЭВМ, строящаяся по модульному принципу [24]. К числу основных функций системы относятся:

- сбор информации о занятости участков от систем сигнализации и автоблокировки, а также от низовых подсистем централизации;
- вывод на цветные дисплеи информации о поездном положении;
- ведение в реальном масштабе времени графика исполненного движения, его анализ и оповещение диспетчера об отклонениях;
- автоматическое управление установкой маршрутов и показаний светофоров в соответствии с графиком движения или указаниями диспетчера;
- автоматический перерасчет нормативного графика на случай сбоев в работе;
- автоматическая регистрация и документирование графика исполненного движения;
- контроль функционирования системы и другие функции.

Система базируется на центральном вычислительном комплексе, включающем две мини-ЭВМ типа PDP-11-60 и две коммуникационные микро-ЭВМ КМС-11. Периферийные подсистемы централизации типа ТО-10 построены на современной элементной базе и обеспечивают высокий уровень достоверности информации.

Железными дорогами Германии в сотрудничестве с фирмой “AEG TELEFUNKEN” разработана система ZNA-800 индикации на пультах-табло диспетчеров номеров поездов [24]. Система пользуется исходной информацией с пульта-табло, осуществляет слежение за передислокацией поездов и индицирует номера поездов. Аппаратура индикации номеров поездов рассматривается как составная часть системы автоматизации диспетчерского управления с использованием ЭВМ.

В описанных выше системах широко применяется микропроцессорная элементная база для построения центральных устройств обработки и хранения информации, модульный принцип построения оборудования, автоматическое отображение номеров поездов и другой информации, автоматическая регистрация графика исполненного движения. Общим для всех систем является использование диспетчерской централизации и создание центров управления, охватывающих значительный полигон железнодорожной сети. Причем создание таких дорогостоящих систем (например, спутниковые) вполне оправдано, так как направлено для достижения цели - оптимизация диспетчерского управления. Естественно, если такая цель будет достигнута с меньшими затратами, то экономическая эффективность такого решения очевидна.

В отечественной практике, реализация модели для полигона диспетчерского управления, с ограниченной степенью детализации процесса впервые имела место на Белорусской дороге. Научное обоснование выбора структуры модели, необходимых элементов информационного обеспечения, математического аппарата отражено в [24-27], а в работе [24], дано определение динамической модели района диспетчерского управления (ДМР) и сформулированы требования, предъявляемые к ней.

Так, главным содержанием ДМР являются данные о местонахождении и краткие сведения обо всех поездах, находящихся на районе управления, состоянии напольных устройств автоматики и телемеханики, график исполненного движения. Динамическая модель поездного положения должна формироваться в реальном масштабе времени на базе информации о состоянии участков и станций, о номерах поездов, занимающих конкретный блок-участок или станционный путь. Она должна быть достоверна и объективна, первичная информация должна поступать без участия человека.

Существующие в настоящее время методы отслеживания движения поезда основаны на событийном подходе. В модели фиксируются моменты времени при отправлении, проследовании по перегону, прибытии на следующую станцию и так далее для каждого поезда, находящегося на полигоне диспетчерского управления. Тогда модель движения поезда представляет собой последовательность событий:

C (1.5)

где 6’"(/) - отправление поезда со станции *m,* - проследование поезда по перегону *тк , S*- прибытие на станцию *к*. В свою очередь каждое из событий в модели движения является функцией от состояния объектов телесигнализации системы ДЦ. Например, отправление со станции *m :*

s;;W=fc.5',’.si.s;,sr,s;,)> (i.в)

где S". - открытие выходного светофора, S? - занятие поездом горловины станции, - закрытие выходного светофора, 6Ц - занятие поездом участка удаления, - освобождение горловины, - освобождение участка удаления.

На основе этих принципов Кейзером А. П. была разработана и реализована на базе ЭВМ работающая в реальном масштабе времени динамическая модель поездного положения, дающая информацию о дислокации и номерах подвижных единиц на полигонах диспетчерских участков и района управления в целом [46]. Однако данная система имела ряд недостатков, основными из них являются:

- привязка модели к конкретным устройствам (пульт-табло ДЦ), а следовательно и невозможность реализации на более современных технических средствах;
- недостаточный уровень достоверности информации (полностью определяется применяемой системой ДЦ);
- ориентация на ручной ввод номеров поездов.

Существующие системы управления движением поездов определяют местоположение поезда по сигналам систем СЦБ о занятии и освобождении рельсовых цепей на перегонах и станциях. Номер поезда, вводимый в систему на станции формирования или входному стыковому пункту «перемещается» в памяти ЭВМ в соответствии с изменением этих сигналов. Такой косвенный способ является единственно возможным способом построения поездной модели полигона дороги при существующих устройствах железнодорожной автоматики. В [24] отмечены его следующие основные недостатки:

- возможность «потери» поезда при отказах рельсовых цепей и аппаратуры сигнальных точек автоблокировки;
- необходимость ручного ввода номеров поездов;
- необходимость хранения большого числа информационных массивов о положении поездов в памяти ЭВМ;
- при длительных отказах вычислительных средств возникает необходимость в ручном восстановлении соответствия номеров поездов их действительной дислокации.

Более рациональным, по сравнению с косвенным способом построения и ведения динамической поездной модели, является метод, основанный на применении напольных устройств идентификации подвижного состава. Однако по ряду причин, такие устройства не нашли применения на сети железных дорог Украины. Кроме того, в работе Петрова А. П. [48] показано, что существующие технические средства не позволяли создать информационную систему с надежной обратной связью, благодаря которой может быть обеспечено соответствие информационной модели состоянию перевозочного процесса.

Таким образом, возникает необходимость синтеза модели, ориентированной на технические средства, находящиеся в эксплуатации на полигоне диспетчерского управления и возможностью перехода, с минимальными затратами, на более совершенные.

Наиболее приемлемым, в такой ситуации, является синтез модели ДМР с использованием первичной информации от находящихся в эксплуатации на большинстве дорог Украины систем АСОУП и ДЦ. Первая из них предоставляет данные о номере поезда, станции формирования, времени отправления, составе поезда, проследовании поездом стыкового пункта дороги, а по информации от системы ДЦ (последовательное занятие и освобождение путевых участков) отслеживать его перемещение по элементам путевого развития полигона диспетчерского управления. Естественно, что в этом случае необходимо создание модели функционирования путевых участков, по информации системы диспетчерской централизации. В связи с этим, наиболее существенным вопросом является недопущение потери информации или же не превышение этой потери определенных заданных допусков. Обеспечить это условие возможно как за счет надежной работы устройств, так и правильным построением системы опроса источников информации [30]. Исследованные Дороховым Е. А. потоки информации от телемеханических устройств железнодорожной автоматики, сведены к простейшим потокам, обладающим свойствам ординарности и отсутствия последействия. Такой подход позволил определить потери информации, как для отдельного контролируемого объекта, так и для системы в целом, а также выразить возможную потерю в процентах. На этой основе дается методика расчета оптимального периода съема информации при циклическом опросе контролируемых объектов. Естественно, что такой расчет производится для конкретных устройств, но в любом случае, значение возможных потерь информации не будет нулевым.

Кроме этого, возможны потери информации за счет искажений, возникающих в системе диспетчерской централизации. В работе Переборова А.С.

[76] приведены количественные данные, методика расчетов возможной потери информации при возникновении искажений.

Следует отметить, что для систем диспетчерской централизации источником первичной информации о местонахождении подвижных единиц служит датчик - рельсовая цепь (РЦ). По данным ряда авторов надежность функционирования РЦ, достоверность информации, получаемая от датчика, не в достаточной мере удовлетворяет поставленным требованиям. Так, по данным Меньшикова Н. *Я.* [75], на отказы РЦ приходится 25-30% общего количества отказов устройств СЦБ, а в работе Ягудина Р. Ш. [111] приведены данные, характеризующие интенсивности отказов рельсовых цепей по видам неисправностей. Кроме того, рельсовые цепи подвержены влиянию дестабилизирующих факторов, что также негативно сказывается на достоверности информации. В частности, в работе [79], приводятся данные о влиянии тягового тока на функционирование путевого приемника, в условиях возникновения электрической дуги на токоприемнике электровоза.

Следует также отметить что, при интерпретации информации от рельсовой цепи, в задаче моделирования движения поезда, несколько смещается акцент, по сравнению с применением РЦ в станционных и перегонных устройствах автоматики. Так, в устройствах СЦБ, в первую очередь должно выполняться требование безопасности - не допущение индикации свободного состояния при нахождении подвижной единицы в зоне действия датчика. Реализация этого требования приводит к тому, что при любой неисправности рельсовой цепи должна быть индикация состояния - «занято». В то же время, при моделировании движения поезда должны различаться ситуации: а) РЦ индицирует «занято», а поезда в зоне действия датчика нет; б) РЦ - «занято» и поезд находится на рельсовой цепи. Эта задача выдвигает дополнительные требования к моделированию работы путевого участка на основе информации, получаемой от системы диспетчерской централизации.

Кроме того, реализации приведенных выше моделей позволяли диспетчеру получить ответ на вопрос: «Какая ячейка табло индицирует занятость, и какой

номер поезда высвечивается при этом?». Более логичен в этой ситуации, например такой ответ: «Поезд номер 3604 находится на 14-м километре перегона А-Б». Реализация модели поездного положения, имеющая такую функциональность, позволила бы снизить загрузку диспетчера. Естественно, что при этом необходимо моделировать путевое развитие полигона диспетчерского управления. Так, в работе Мирошниченко В. М. [64] показана принципиальная возможность применения аппарата теории графов для выработки стандартной записи, позволяющей представить структуру схемы железнодорожной станции и производить необходимые операции с этой информацией. Однако подход, описанный автором, в большей степени ориентирован на решение задач транспортного проектирования, а не на реализацию модели поездного положения.

На основе изложенного можно сформулировать цель настоящей работы - совершенствование автоматизированной системы отслеживания дислокации подвижных единиц на элементах путевого развития района диспетчерского управления.

Для достижения этой цели необходимо решить ряд задач:

1. Выполнить анализ структуры, технических средств и информационного обеспечения системы оперативного управления движением поездов на железнодорожном транспорте;
2. Разработать общую поездную модель района диспетчерского управления, так как существующие в настоящее время реализации ДМР не в должной мере обеспечивают соответствие информационной модели состоянию перевозочного процесса;
3. Разработать модель путевого развития полигона диспетчерского управления, адаптированную под функциональность общей модели;
4. Обосновать выбор математического аппарата нейронных сетей и разработать прицины построения моделей функционирования станционных устройств автоматики;

5. Разработать модель функционирования путевого участка, по информации системы диспетчерской централизации;
6. Исследовать модели на соответствие состоянию перевозочного процесса;
7. Составить алгоритмы и программную реализацию моделей на основе современных аппаратных средств и языков программирования высокого уровня;

Выводы по разделу

В разделе произведен анализ структуры, технических средств и информационного обеспечения системы оперативного управления движением поездов. Сделан вывод о том, что при существующей технической оснащенности железных дорог Украины возможно совершенствование автоматизированной системы отслеживания дислокации подвижных единиц на элементах путевого развития района диспетчерского управления. На основе анализа литературы показана необходимость синтеза модели ДМР с использованием первичной информации от находящихся в эксплуатации систем АСОУП и ДЦ. Поставлена цель и сформулированы задачи исследования.

РАЗДЕЛ 2
СИНТЕЗ ДИНАМИЧЕСКОЙ ПОЕЗДНОЙ МОДЕЛИ РАЙОНА ДИСПЕТЧЕРСКОГО УПРАВЛЕНИЯ

2.1. Синтез общей динамической поездной модели и путевого развития района диспетчерского управления

Под моделью понимается физический объект или некоторое описание, отображающее те свойства и отношения оригинала, которые в данном исследовании признаны существенными [48]. Среди возможных моделей можно выделить объектные (физические объекты-макеты и др.), словесно-описательные и математические. Объектное моделирование поездного положения на районе диспетчерского положения трудно реализуемо, и его применение не имеет смысла. Словесно-описательная модель находит применение в процессе оперативного управления движением поездов, однако, модели такого типа невозможно использовать для исследований и формальных методов анализа. Таким образом, наиболее предпочтительнее использовать математическую модель, где свойства полигона управления описываются переменными, условными знаками, соотношениями.

Как показано в 1-м разделе, главным содержанием динамической модели района диспетчерского управления являются данные о местонахождении и краткие сведения обо всех поездах, находящихся на районе управления.

Введя обозначения, общую модель можно представить в виде выражения:

$$Щ = (лг,,м_2,...ж,,), \qquad (2.1)$$

где под M_n понимается информационная характеристика каждого поезда, находящегося в районе диспетчерского управления, n - общее количество поездов.

В связи с тем, что поездная ситуация изменяется во времени, то и общая модель ДМР будет функцией времени:

$$^(0 = 174,(0,тИ_2(0,..,М_п(0), \qquad (2.2)$$

где М,,(z) - изменение информации о *и* - ом поезде в течении времени.

На рис. 2.1 показано содержание ДМР, где $М_п(/)$ - изменение дислокации *n* -го поезда по путевому развитию полигона диспетчерского управления. В связи с этим, в функции должны быть соответствующие параметры, учитывающие координаты головы поезда. Остановимся на этом вопросе более подробно.

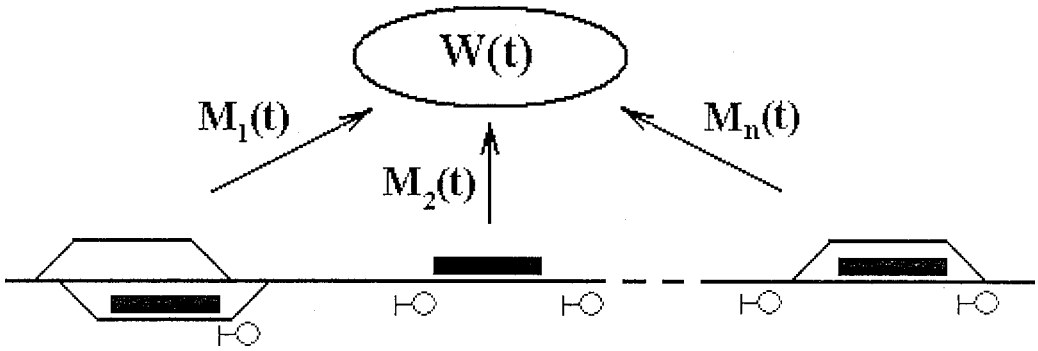


Рис. 2.1 Содержание динамической модели региона

Так, для определения местоположения движущегося объекта относительно земли применяют такое количество переменных, сколько степеней свободы движения для него существует. Например, на воздушном транспорте, для описания местоположения самолета, применяются географические координаты (долгота, широта) и высота. В морском транспорте, для определения дислокации корабля, две - долгота и широта. На железнодорожном транспорте применение географических координат не нашло широкого применения. При диспетчерском управлении движением поездов, для определения местонахождения поезда, используют две координаты: номер пути и расстояние от оси станции.

В связи с этим, любой подвижной единице (поезду) можно поставить в соответствие вектор:

$$M(t,x,y), \qquad (2.3)$$

где x - расстояние от оси станции, до первой колесной пары головы поезда, y - номер пути, на котором находится поезд.

При движении поезда по станции, координата x изменяет свое значение от x_n (ордината входного светофора для маршрутов приема или выходного для маршрутов отправления), до $x_к$ (ордината выходного или маневрового светофора с того пути, на который осуществляется прием поезда, для маршрутов приема, или ордината входного светофора противоположного направления для маршрутов отправления).

Координата y изменяется в зависимости от положения стрелок и направления движения. Для стрелки любого типа определено множество входных (уд) и выходных (у,,ыл.) значений y. У одиночных стрелок эти множества состоят из двух элементов (Рис. 2.2) у,(л. = {1,2}, $y_{кых}$ = {0,2}, где **0** используется для описания

недопустимого перемещения, т.е. при плюсовом положении стрелки и указанном направлении движения, перемещение осуществляется с пути 2 на путь 2, с пути 1 перемещение недопустимо, при минусовом - с пути 1 на путь 2, с пути 2 -

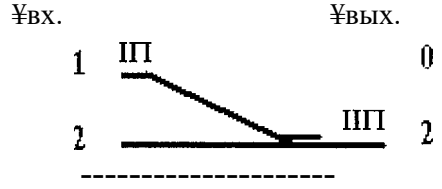


Рис. 2.2 Множества значений у(м и y... для одиночной стрелки

Если при плюсовом положении стрелки и указанном направлении движения, перемещение осуществляется с пути 2 на путь 2, с пути 1 перемещение недопустимо, при минусовом - с пути 1 на путь 2, с пути 2 - недопустимо. При изменении направления движения множества $y_{Г1X}$ и $y_{Г1m}$ меняются местами.

Спаренная стрелка состоит из двух одиночных, множество выходных значений $y_{ных}$ для первой по ходу движения стрелки, является множеством ВХОДНЫХ $y_{1Я}$ для второй, и оно будет множеством выходных значений спаренной

стрелки. При изменении направления движения изменяются только множества входных значений. Например, для спаренной стрелки 2/4 (Рис. 2.3) определены множества $y_{i;x2}$, $y_{,,x4}$, $y_{tlh,x2}$,y_{llhlx4}, тогда х,,,,-2 = Z«4 = Для четного направления Ч (обозначено стрелкой) $y_{11X} = y_{1a2}$, для нечетного - $т_{бч.} = J/_{вмv4}$.

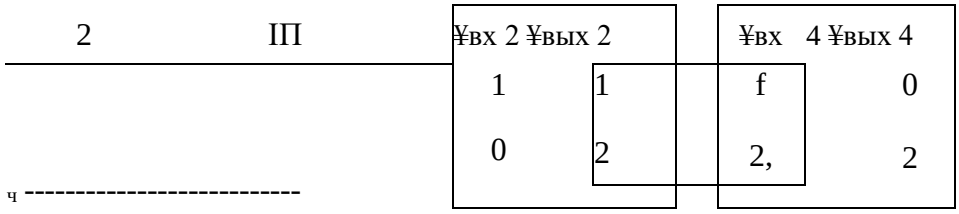
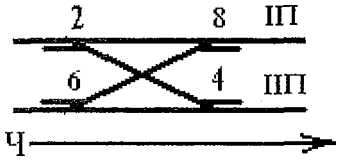


Рис. 2.3 Множества значений y~ и y_{an}. для спаренной стрелки

Перекрестная стрелка состоит из двух спаренных, множества выходных значений $у_{КЫЛ}$. для двух первых по ходу движения стрелок, являются множествами входных $y_{(:x}$ для вторых, значения этих множеств равны и эти значения составляют множество выходных значений перекрестной стрелки. При изменении направления движения изменяются только множества входных значений. Например, для перекрестной стрелки 2/4-6Z8 (Рис. 2.4) определены множества , Лг4 > *У^х2 ,* > *У«хб , У^i > У^б , У вых»*> тогда $у_{Льи2} = y_{Г1x4} = y_{г,ыхб} = = y_{flhlx}$. *Для*

множеств $y_{,:x} = y_{,:x2}$ $о_{Л,б}$, для нечетного $у_{вх}$ =



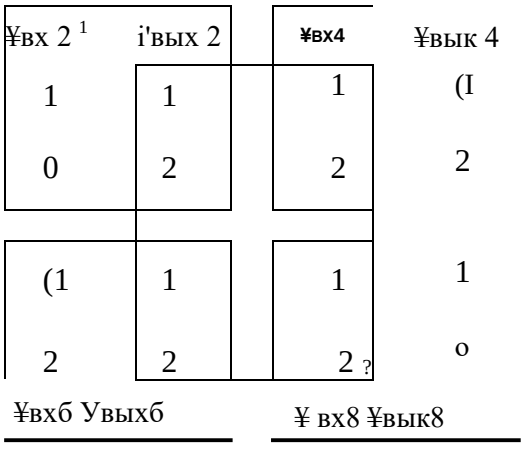


Рис. 2.4 Множества значений $y._{Г)}$ и $y._{ЛЙ}$ для перекрестной стрелки

Таким образом, для определения множеств входных и выходных значений стрелок *(yra.,y,lh,x),* исходной информацией является однониточный план станции. Для станции любого типа можно составить таблицы соответствия значений *y* в зависимости от направления и положения стрелок.

В модели движения поезда (2.3) при равенстве значения *x* ординате какой-либо стрелки, происходит присвоение координате *y* значения из множества согдасно выражения:

$$y' = У_{11Ы}.^{\wedge если}\text{-}У^{\epsilon}\ y_{l<x},и\_y * 0,при\_x = x_c, \qquad (2.4)$$

где x_c - ордината соответствующей стрелки.

Для всех типов стрелок, при выполнении условия

$$если\_y\ \epsilon\ y_{вх},и\_y\ Ф\ 0,при\_x = x_c$$

выражение (2.4) примет следующий вид:

$$У = \begin{cases} \backslash_{Ы.Г}| > если\ \_y - y_{BV1}\ \_\ и\ \_\ P = О \\ , Унесли\ \_y = y,_{аУ}\_и\_P = \backslash \\ > {}^{еСЛи}\text{-}У = У«_{Х}2 \quad = ® \\ У,_{!Ы}Л\text{-}1, если\ \_y = y_{sv2}\ \_\ и\ \_\ P = 1 \end{cases} \qquad (2\text{-}5)$$

, где P = 0- плюсовое положение стрелки, P = 1 - минусовое.

Например, для станции (Рис. 2.5) составляется таблица значений *x* (ординаты светофоров, стрелок) и множеств y_{11X} и y_{mx} для каждой стрелки. При приеме поезда на путь Ш, в момент занятия участка ЧАП координатам вектора *M* (2.3) присваиваются начальные значения: x = 1190, *y* = 2. При дальнейшем движении значение *x* уменьшается, и в момент его равенства ординате соответствующей стрелки, происходят проверки условий из (2.4) и (2.5), значению *y* присваивается номер того пути, на который переведена стрелка.

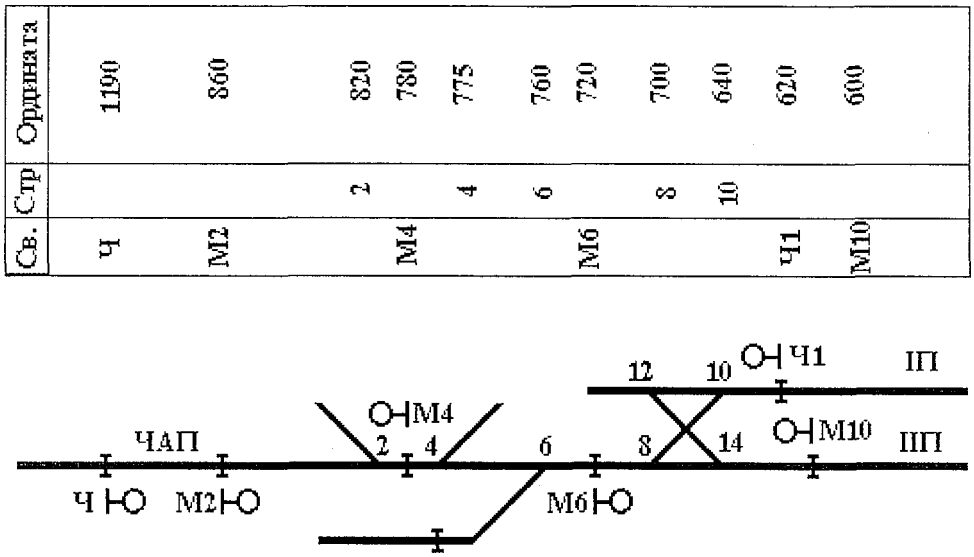


Рис. 2.5 Фрагмент однониточного плана станции.

Так, для перекрестной стрелки 8/10-12/14 ординаты 8-й и 12-й стрелки равны, при x = 700 происходит проверка множества входных значений каждой из них. Так как y = 2, то условие (2.4) выполняется только для стрелки 8, и при минусовом положении 8/10 происходит присвоение координате y значения 1 (первый путь) $y' = y_{<lhai}$, потому что $y = y_{вд,2}$ и P = 1 (2.5).

После окончания движения поезда (прибытие на первый путь) координаты вектора M принимают значения x = 620 (ордината светофора 41) $y = 1$ (первый ПУТЬ).

Аналогичным образом происходит функционирование модели и при отправлении. После вступления головы поезда на первый участок удаления координата x вектора M принимает значение ординаты входного светофора встречного направления. Далее, в зависимости от того, какими устройствами оборудован прилегающий перегон, значению x присваиваются значения относительной координаты. Так, если перегон оборудован устройствами автоматической блокировки (АБ) (Рис. 2.6), в момент проследования головой поезда сигнальной установки 4, координата x принимает значение 12400, и так далее, до входного светофора следующей станции.

Рис. 2.6 Схема движения поезда по перегону,
оборудованному устройствами автоматической блокировки

Таким образом, общая динамическая модель района диспетчерского управления будет являться множеством векторов подвижных единиц, находящихся в определенный момент времени на элементах путевого развития полигона и примет вид:

$$T(/) = (M_{|}фx,y),M_2(Цx,y),...,Л/',,(Г,x,y)), \qquad (2.6)$$

В связи с тем, что существующие устройства автоматики оперируют только с дискретной информацией, точное значение времени может быть определено только в некоторых точках, а именно в моменты занятия или освобождения поездом участков пути, стрелочных секций, блок-участков на перегоне.

Таким образом, используя векторный подход при моделировании движения поезда, можно контролировать передвижение подвижных единиц, по их основным параметрам - номер пути и расстояние от оси станции.

Функционирование рассмотренной модели основано на предположении, что все возможные искажения устраняются вне модели, и оперирование происходит только с достоверными данными. Такой подход наиболее применим для микропроцессорных систем диспетчерской централизации, которые, по своей природе, являются информационными и достоверность информации обеспечивается как на схемном, так и на программном уровне. Однако, для большинства находящихся в эксплуатации систем ДЦ, моделирование должно производиться с учетом возможности возникновения искажений во входной информации. А так как источником такого рода данных являются устройства

автоматики на станциях и перегонах (Рис. 1.4), то возникает необходимость в синтезе соответствующих математических моделей.

2.2. Первичные модели функционирования станционных устройств автоматики

Известно, что в настоящее время синтез устройств автоматики осуществляется на базе теории конечных автоматов [15, 85]. С позиции этой теории работа устройства управления рассматривается как «черный ящик» (Рис. 2.7), у которого существует некоторое множество входного алфавита $A = \{a_1,a_2,a_3,\ldots,a_n\}$, выходного алфавита $Z = \{z_1,z_2,z_3,\ldots,z_n\}$, внутренних состояний S, начального состояния $s_t eS$, функции переходов $\delta(a,s)$ и функции выходов /(а,л) [15]:

$$M = (A,S,Z, 5, ,8\{a, б'), \qquad ,.\$\blacksquare))$$

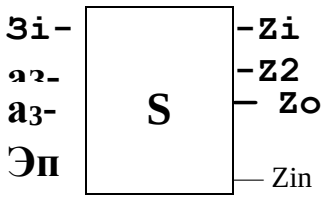


Рис. 2.7 Абстрактный конечный автомат.

Входной и выходной алфавиты, а так же множество внутренних состояний автомата предполагаются конечными, функции переходов и выходов δ и % однозначны и всюду определены. Абстрактный анализ осуществляется с использованием таблиц или графов переходов [15].

Например, для горловины станции (Рис. 2.8) устройства централизации должны осуществлять управление стрелкой и изменять показания светофора в зависимости от управляющих команд и поездной ситуации.

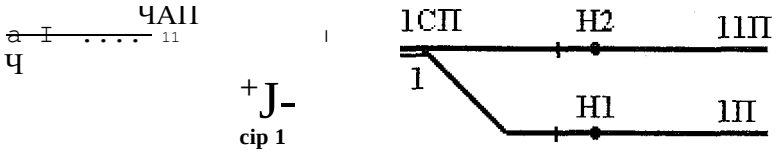


Рис. 2.8 Схематический план горловины станции.

Исходя из теории конечных автоматов, устройство, осуществляющее эти функции можно представить в виде абстрактного автомата 5 (Рис. 2.9). Входной алфавит *A* определяется множеством значений состояний датчиков напольных устройств и кнопок управления:

- путевые датчики: ЧАП, 1СП, 2П, Ш;
- положение стрелки: ПК 1, МК1;
- кнопки управления: светофоры - Ч, Hl, Н2, стрелка - «+»,«-».

Выходной алфавит Z - это множество значений выходов, или управляющих команд, подаваемых на напольные устройства (стрелочные управляющие: ПУ1, МУ 1; лампы огней светофоров: Ж1, К, Ж2).

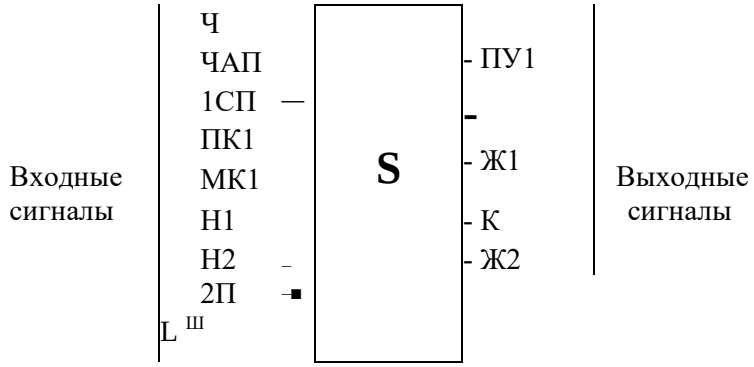


Рис. 2.9 Абстрактный автомат, осуществляющий управление станционными устройствами.

При решении задачи абстрактного синтеза конечного автомата входным условием является регулярное выражение события *E,* для его определения необходимо рассмотреть процесс управления передвижением подвижных единиц на станции, который может быть как с ручным управлением стрелками и сигналами, так и с маршрутным. В первом случае происходит ручной перевод стрелки в необходимое положение и нажатие кнопки соответствующего

светофора. После этого системой централизации должны провериться все условия безопасности и, при их выполнении, произойти смена показания светофора. При движении поезда, после вступления его на первый участок за светофором, должно смениться разрешающее показание на запрещающее (красный огонь). При маршрутном управлении команда на установку подается, как правило, нажатием двух кнопок на пульте управления. Так, для маршрута приема на 2-й путь, первой нажимается кнопка входного светофора Ч. После этого нажимается кнопка выходного светофора Н2. Затем, система централизации должна проверить положение стрелок, входящих в маршрут, и если оно не соответствует требуемому, то формируется команда на перевод. Дальнейшие действия такие же, как и в системе с ручным управлением стрелками и светофорами. Эти процессы являются регулярными, формально задаются в виде таблиц или графов состояний.

Последовательность событий для раздельного и маршрутного управления показана соответственно в таблице 2-Ій таблице 2.2, где единицам соответствует свободное состояние рельсовых цепей, контроль положения стрелки, нажатое состояние кнопок. За исходное состояние ($Л_о$) принято минусовое положение стрелки, показание светофора Ч - красный, маршрут задается на 2-й путь.

Таблица 2.1

Последовательность событий при раздельном управлении

Событие	Ч	**ЧАП**	**1СП**	**ПК1**	**МК1**	+	-	**Выход**
Л	**0**	**1**	**1**	**0**	**1**	**0**	**0**	
4	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**ПУ1**
4	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**Ж1**
$А_Л$	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**К**

Таблица 2.2

Последовательность событий при маршрутном управлении

Событие	Ч	ЧАП	1СП	ПК1	МК1	Н2	2П	Выход
А	**0**	**1**	**1**	**0**	**1**	**0**	**1**	
А	**1**	**1**	**1**	**0**	**1**	**0**	**1**	
A_2	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**ПУ1**
А	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**Ж1**
А	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**К**

В таблицах 2.1, 2.2 приведены только те события, в результате наступления которых формируются управляющие команды. При раздельном управлении (Таблица 1) каждое входное событие *(A_1-A_3)* приводит к формированию определенной команды управления, а при маршрутном нет (например событие A_1 - нажатие кнопки начала маршрута), однако для формирования правильного управляющего сигнала необходима вся последовательность событий. В этом случае можно говорить о синтезе автомата с памятью. Более подробно эти вопросы рассмотрены в работах [15, 85].

Ограничением применения теории конечных автоматов является то, что они работают только в том случае, когда условия работы описываются регулярными выражениями. Когда же имеют место случайные, или не учтенные на этапе проектирования события, то система не может нормально функционировать. Однако создание стохастической модели работы устройств автоматики весьма затруднительно. Действительно, вероятность определения конкретного значения времени перехода дискретного устройства из одного состояния в другое стремиться к нулю, на промежутке времени стремящемся к бесконечности:

Д(<М- (2.8)

Но, в связи с тем, что в процессе движения поездов присутствует цикличность (параметр Дг в формуле (2.8) равен 24 часам), и в течение этого промежутка распределение вероятности перехода $P_n(f)$ не равномерно. Последнее

утверждение можно доказать, совместив по значению времени изменение состояния объектов автоматики с нитками на графике движения поездов. Так, вероятность срабатывания (переход из состояния свободно в состояние занято) путевого датчика блок-участка на перегоне (Б), оборудованного автоблокировкой, возрастает при совпадении текущего времени с ниткой на графике (Рис. 2.10). Очевидно, что в моменты времени t_1, t_2, t_3 значение функции $p(t)$ максимально.

Аналогичные графики можно построить и для других объектов автоматики, непосредственно связанных с движением поездов, естественно, если информация об их функционировании присутствует в системе диспетчерской централизации.

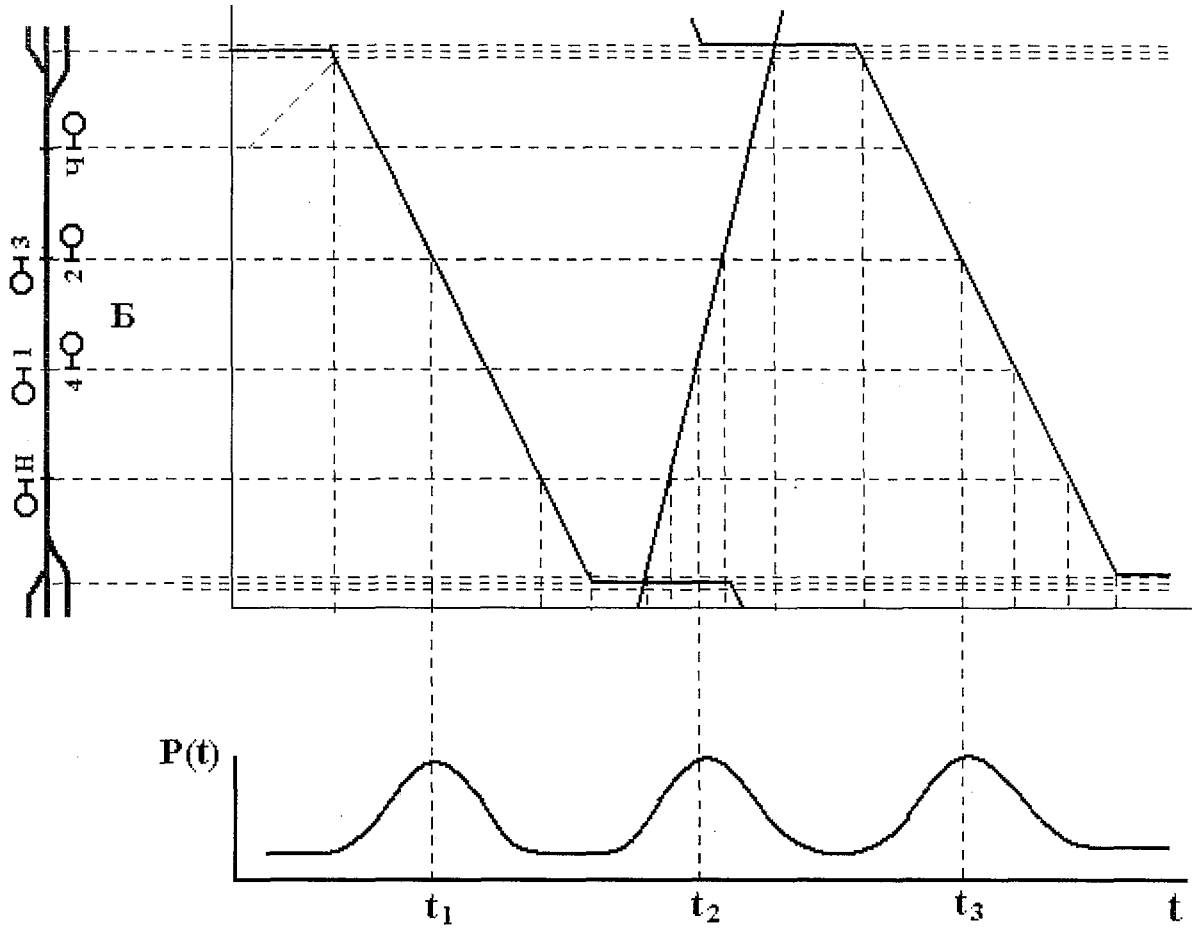


Рис. 2.10 Совмещение ниток на графике движения поездов
с изменением состояния объектов автоматики

Таким образом, существует возможность синтеза стохастической модели работы объектов автоматики, но для этого, в первую очередь, необходимо иметь функцию распределения вероятности $p(t)$ для каждого объекта. Аналитически

получить такую функцию не представляется возможным по ряду причин, к основным из них можно отнести:

- отсутствие значений длин участков на графике движения и, следовательно, значительная временная погрешность при совмещении ниток на графике с изменением состояния объектов автоматики;
- не по всем ниткам на графике осуществляется пропуск поездов;

Вид функции *P(i)* можно получить статистически, однако, для этого необходимо иметь значительное количество априорной информации, и даже при ее наличии, аппроксимация представляет определенную трудность. Это утверждение сделано на основе анализа статистических данных работы устройств автоматики, по информации системы диспетчерской централизации, за различные периоды времени (результаты частично опубликованы в работе [106]).

В ходе исследований измерялся промежуток времени, в течение которого объект типа участок пути находился в состоянии «занято», определялось количество таких случаев за один месяц 1999г. (Приложение А). Для различных объектов телесигнализации систем ДЦ (станции Мандрыкино, Донецк, Рутченково Донецкой ж.д. и станции Жуляны Юго-Западной ж. д.) были получены, с помощью созданного соответствующего прогриммного обеспечения (Приложение Б), гистограммы (Рис. А. 1-А.6), на которых ось / - длительность индикации состояния «занято», *n* - количество таких случаев за месяц. Даже беглый просмотр гистограм показывает определенную трудность аппроксимации функции Р(/), а если принять во внимание, что таких объектов в районе диспетчерского управления около 2000, то становится очевидной невозможность применения традиционного математического аппарата для создания стохастической модели функционирования объектов автоматики. К этому следует добавить тот факт, что необходима периодическая корректировка функции связаная с изменением параметров работы объектов (например, сезонные).

В то же время синтез модели работы объектов автоматики может осуществляться с использованием математического аппарата нейронных сетей (НС). Теоретической основой этого предположения служит теорема Хехт-

Нильсена [134] о возможности представления любой многомерной функции нескольких переменных с помощью двухслойной нейронной сети с заранее известной функцией активации.

В настоящее время существует достаточно большое количество различных видов нейронных сетей (например [4, 12, 21, 29, 37, 50, 51, 94, 118] и др.), различающихся как по строению, так и по принципу функционирования. В связи с этим возникает необходимость выбора какой-либо из них, для чего необходимо рассмотреть особенности функционирования некоторых, наиболее распространенных типов нейронных сетей.

2.3. Особенности функционирования некоторых типов нейронных сетей

Существуют различные виды нейронных сетей, для оптимального выбора типа, необходимо дать краткую характеристику наиболее известных из них.

2.3.1. Полносвязанная нейронная сеть без скрытых нейронов

Схема рассматриваемой нейронной сети [22, 37, 118, 119, 123, 148] представлена на рис. 2.11. В сети отсутствуют скрытые нейроны - внешний входной сигнал подается на входы всех нейронов сети, выходы всех нейронов образуют выходной сигнал сети. Сеть функционирует в течение нескольких тактов. Начиная со второго такта, выходной сигнал каждого из нейронов подается на входы всех остальных нейронов.

Уравнения, которые описывают функционирование сети, имеют вид:

$$y_т(0)=0_?\ m= \quad .$$

$$лЫ,А') = А-)У].vpVjy(А-^{-}1)y_{;т} + ,\ i — 1,2,...,Af,\ к = 1,2,...АГ$$

$$у, (£) = f\ (\$\ (£)),\ тїї = 1,2,...,\ Af,\ к = 1,2,...\ К$$

где у - выход нейрона, ш, i - номера нейронов, М - число нейронов в сети, s - выход сумматора сети, j - номер элемента входного сигнала, N - размерность

входного сигнала, x - элемент входного сигнала, w - синаптический вес, v - вес связи от выхода нейрона к входу, к - номер такта функционирования сети, К - число тактов функционирования, b - смешение нейрона, ц(к) - коэффициент затухания, например:

|Ч-

где р - некоторая константа.

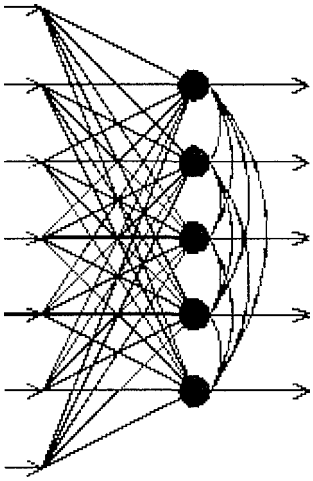


Рис. 2.11 Полносвязная нейронная сеть без скрытых нейронов

Кроме данной, в литературе (например [51]) рассматриваются и другие полносвязные нейронные сети. В общем случае число нейронов и размерность выходного сигнала сети могут не совпадать.

Обучающая выборка имеет вид:

E ___E _____E _____ E . ,-■>!! ,>'E <-€
ь и'Ъ-З ? ' i J"'"' 2d'

5

где а и - элемент входного и выходного сигнала соответственно, е=1,2,...,Р, Р - число обучающих примеров в выборке. На каждый входной сигнал сеть должна выдать соответствующий выходной сигнал.

Если в необученную нейронную сеть ввести входной сигнал одного из примеров обучающей выборки, то выходные сигналы будут отличаться от требуемых, которые определены в обучающем примере. Функция ошибки

определяет степень близости выходных сигналов к требуемым при решении всей совокупности примеров обучающей выборки:

-I ɔP Af

E-I i-1

Перед началом обучения сеть инициализируется - синаптическим весам и смещениям присваиваются некоторые случайные значения из заданного диапазона. На каждой итерации обучения выполняются модификации значений синаптических весов и смещений, уменьшающие функцию ошибки:

$$v_{im}(f+l) = v_{ira}(O + dv_{im}(f) = v_{jra}(O+-—C), \qquad 1,2,..., Af$$

где t - номер итерации обучения;

$$Д(ґ+1) = \quad + \quad = Д(O + \underset{сщ}{—-}(f), i = 1,2,..., M$$

$$v_{ira}\ (t+1) = (f) + 3\ v_{itlt}\ (0 = v_{jre}(f) + \quad f),\ i,m = 1,2,..., M$$

Для НС такого типа справедливо утверждение, что не для любого входного значения сеть сможет сформировать правильное значение выхода [63]. Таким образом, НС такого типа, хоть и проста в реализации, но не может быть использована для моделирования работы устройств автоматики. В болшей мере эти недостатки устранены в сети Хопфилда [135-137].

2.3.2. Нейронная сеть Хопфилда

Одна из первых предложенных моделей сети Хопфилда используется как ассоциативная память (рис. 2.12). Исходными данными для расчета значений синаптических весов сети являются векторы - образцы классов. Сеть функционирует циклически. Выход каждого из нейронов подается на входы всех остальных нейронов. Нейроны сети имеют жесткие пороговые функции (рис. 2.13).

Критерий останова: итерации сети завершаются после того, как выходные сигналы нейронов перестают меняться. Тип входных и выходных сигналов: биполярные (+1 и -1).

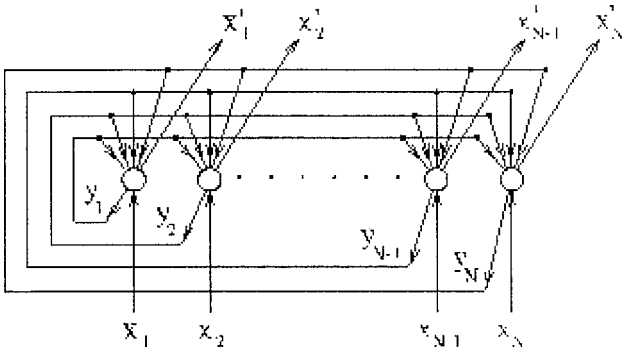


Рис. 2.12 Схема функционирования сети Хопфилда

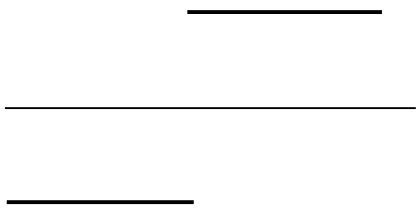


Рис. 2.13 Передаточная функция сети Хопфилда

Размерности входа и выхода ограничены при программной реализации только возможностями вычислительной системы, на которой моделируется нейронная сеть, при аппаратной реализации - технологическими возможностями. Размерности входных и выходных сигналов совпадают. Емкость сети: сеть, содержащая N нейронов, может запомнить не более M=0.15*N образов. При этом запоминаемые образы не должны быть сильно коррелированны. Мера коррелированное™:

$$Д-\underline{У_{.Ух}}Х$$

Число синапсов в сети: M*(M-1). Обозначения на рис. 2.12: му - i-й синаптический вес j-го нейрона, х,. - i-й элемент входного сигнала сети, $х_{;/}$ - i-й элемент j-го вектора-образца, *x′* - i-й элемент выходного сигнала сети, jy - выход

j-го нейрона, N - количество элементов (размерность) входного сигнала, количество нейронов в сети, M - количество векторов-образцов. Формирование синаптических весов сети:

$$LU_{..} = \sum_{\text{и г-1}} {}^{1}\blacksquare' \quad {}^{j}_{J}$$

$$\mathbf{o},$$

Функционирование сети:

$$.y,Ф) = X, 1 - 7 -$$

$$, 1\text{-}У\text{-}$$

$$_{1\text{-}i} \qquad _{J}$$

Функционирование заканчивается, если на некотором шаге T для всех j:

$$y.\ б?) = y/T\text{-}1)$$

Выходной сигнал сети:

$$*7 = .У,(T>$$

В процессе функционирования уменьшается энергетическая функция:

$$E = - o \sum_{1\text{-}1\ J\text{-}1} | JV /У'$$

Нейронная сеть Хопфилда имеет всего один слой и функцию возбуждения, в виде единичного скачка. Для НС такого типа, хоть и меньшей мере, чем для полносвязанной сети, справедливо утверждение Минского-Пайперта [63] об ограниченном круге задач, ею решаемой. Кроме того, в работе [102] раскрыты и другие недостатки такого типа сетей, в частности, функционирование НС с одним слоем нейронов может рассматриваться как решение разностного уравнения. Таким образом, НС такого типа не может быть использована для моделирования работы устройств автоматики. Эти недостатки в большей степени устранены в сети обратного распространения ошибки (back propagation).

2.3.3. Сеть обратного распространения ошибки (back propagation)

Алгоритм обратного распространения [21, 22, 45, 50, 51, 57, 116, 118, 119, 122, 123, 132, 133, 141, 145, 148] т это итеративный градиентный алгоритм, который используется с целью минимизации среднеквадратичного отклонения текущего выхода многослойной сети и желаемого выхода.

Алгоритм обратного распространения используется для обучения многослойных нейронных сетей с последовательными связями. Нейроны в таких сетях делятся на группы с общим входным сигналом - слои. На каждый нейрон первого слоя подаются все элементы внешнего входного сигнала. Все выходы нейронов m-го слоя подаются на каждый нейрон слоя m+1. Нейроны выполняют взвешенное суммирование элементов входных сигналов. К сумме элементов входных сигналов, домноженных на соответствующие синаптические веса, прибавляется смещение нейрона. Над результатом суммирования выполняется нелинейное преобразование - функция активации (передаточная функция). Значение функции активации есть выход нейрона.

На рис. 2.14 показана схема многослойной сети (многослойного персептрона [81-83, 143, 144]). Нейроны представлены кружками, связи между нейронами - линиями со стрелками.

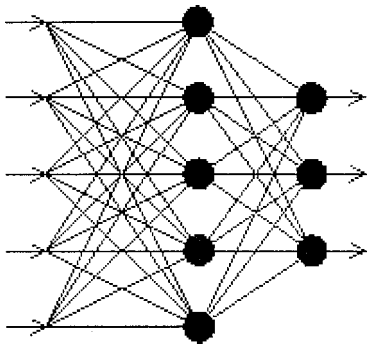


Рис. 2.14 Сеть обратного распространения ошибки

Тип входных сигналов - целые или действительные, тип выходных сигналов - действительные из интервала, заданного передаточной функцией нейронов, тип

передаточной функции - сигмоидальная. В нейронных сетях применяются несколько вариантов сигмоидальных передаточных функций.

Функция Ферми (экспоненциальная сигмоида):

$$\text{од—Ь.} \quad \blacksquare\ 1 + e^{-'*} \qquad (2.8)$$

,где s - выход сумматора нейрона, а - некоторый параметр.

Рациональная сигмоида:

$$|s| + c_i$$

Гиперболический тангенс:

$$^{a} \quad e''+e^{\wedge}$$

Перечисленные функции относятся к однопараметрическим. Значение функции зависит от аргумента и одного параметра. Также используются многопараметрические передаточные функции, например:

$$\text{ЛЭ} = \text{Р1 гз}\sim\!\!\!-\!\!\!- {}^{+} \text{А}$$
$$\text{И}^{+}\ \text{р}_{\text{г}}$$

Сигмоидальные функции являются монотонно возрастающими и имеют отличные от нуля производные на всей области определения. Эти характеристики обеспечивают правильное функционирование и обучение сети. Таким образом, наиболее эффективной передаточной функцией является рациональная сигмоида.

Функционирование многослойной сети выполняется в соответствии с формулами:

$$5_{;} = \text{Х'щ.- У/ “ Д , i}_{„„.}{}^{=} 1.2\ /\text{V1,2,...,}\ L \qquad (2.9)$$

$$\text{-1.2 ....}L, \qquad (2.10)$$

где s - выход сумматора, w - вес связи, y - выход нейрона, b - смещение, i - номер нейрона, N - число нейронов в слое, m - номер слоя, L - число слоев, f- функция активации.

Обучение сети разбивается на следующие этапы:

1. Инициализация сети. Весовым коэффициентам и смещениям сети присваиваются малые случайные значения из диапазонов и соответственно.

2. Определение элемента обучающей выборки. (<Текущий вход>, <желаемый выход>). Текущие входы (xO, xl... xN-I), должны различаться для всех элементов обучающей выборки. При использовании многослойной сети в качестве классификатора желаемый выходной сигнал (dO, dl... dN-1) состоит из нулей за исключением одного единичного элемента, соответствующего классу, к которому принадлежит текущий входной сигнал.

3. Вычисление текущего выходного сигнала. Текущий выходной сигнал определяется в соответствии с традиционной схемой функционирования многослойной нейронной сети.

4. Настройка синаптических весов. Для настройки весовых коэффициентов используется рекурсивный алгоритм, который сначала применяется к выходным нейронам сети, а затем проходит сеть в обратном направлении до первого слоя. Синаптические веса настраиваются в соответствии с формулой:

$$£ + 1) = ш,-,,■(£) + rg\ д,,- \qquad (211)$$

где я; - вес от нейрона i или от элемента входного сигнала i к нейрону j в момент времени t, x' - выход нейрона i или i-ый элемент входного сигнала, г - шаг обучения, g_i - значение ошибки для нейрона j.

Если нейрон с номером j принадлежит последнему слою, то

$$3‘./^{=}\ Г\text{-}0\ '\ УЖ/^{-}\ Г? \qquad (2.12)$$

где d_y - желаемый выход нейрона), y_t - текущий выход нейрона).

Если нейрон с номером j принадлежит одному из слоев с первого по предпоследний,то

(2.13)

где к изменяет свое значение по номеру нейрона слоя на единицу больше, чем у того, которому принадлежит нейрон j. Внешние смещения нейронов b настраиваются аналогичным образом.

Моделирование работы объектов автоматики основывается на предпосылке, что в условиях нормального функционирования существует некоторое количество априорной информации. В процессе эксплуатации могут возникать случайные явления, приводящие к искажениям в первичной информации, поэтому необходимо выбрать нейронную сеть, позволяющую производить первоначальный процесс обучения на ограниченной выборке, а затем дообучаться при появлении неопределенностей. Для этого наиболее подходит нейронная сеть с обратным распространением ошибки (back propagation), однако, прежде всего, необходимо более детально рассмотреть возможность ее применения для моделирования работы устройств автоматики.

5. 4. Синтез моделей функционирования устройств автоматики с применением нейронной сети обратного распространения ошибки

Основными параметрами нейронной сети является количество слоев, число нейронов в каждом слое, функции возбуждения нейронов. Как и при синтезе конечного автомата (Рис. 2.7) количество входных и выходных нейронов сети определяется исходя из множеств ВХОДНОГО *(A = {$a_л$,C1$_2$,a-.,...,a_n}')* и выходного (Z = -$_2$,zj) алфавитов соответственно. Такие параметры конечных автоматов, как множество внутренних состояний 5, начальные состояния .у, є S , функции переходов j(a,i), функции ВЫХОДОВ /(щл), в нейронных сетях не применяются. Определение числа нейронов в скрытых слоях представляет собой нетривиальную задачу и в настоящее время не существует универсальной методики ее решения. Однако, для приблизительной оценки этого числа можно воспользоваться известной теоремой Колмогорова-Арнольда и следствием из нее. В соответствии с этими теоретическими результатами, в двухслойной нейронной сети для реализации произвольного отображения потребуется $2^Л$ нейронов в

скрытом слое, где N- размерность выходного сигнала или число нейронов последнего слоя.

Для более точной оценки числа нейронов в скрытых слоях можно воспользоваться формулой для оценки необходимого числа синаптических весов N_w в многослойной сети с сигмоидальными передаточными функциями [50]:

$$\frac{N}{1+\log_2(7V_{/7})} < N_w < \frac{}{7V_v} + 1\ 7V + 77,. + 1_7 + N$$

где N- размерность выходного сигнала, N - число элементов обучающей выборки, N_x - размерность входного сигнала.

Оценив необходимое число весов, можно рассчитать количество нейронов в скрытых слоях. Для двухслойной сети число нейронов в скрытом слое составит:

$$N_x + N$$

С учетом вышесказанного, для приведенной на рис. 2.8 станции нейронная сеть будет иметь 7 нейронов на входе (количество входных сигналов) и 3 на выходе (количество управляющих выходов), один скрытый слой. Нейроны первого слоя А (Рис. 2.15) осуществляют масштабирование входных сигналов в диапазон {0,1} сигмоидальной функцией (2.8). Каждый нейрон этого слоя связан со всеми нейронами скрытого слоя S, причем эта связь имеет численное выражение, которое вычисляется в процессе обучения сети. Нейронами слоя S вычисляется значение выхода, которое подается на последний слой Z, с пороговой функцией активации.

На входы и выходы сети подаются все элементы обучающей выборки. После проведения процесса обучения проверяется правильность работы сети и подбирается порог срабатывания нейронов последнего слоя (Таблица 2.3), который в данном случае равен 0,7 (событие А2, выход Ж1).

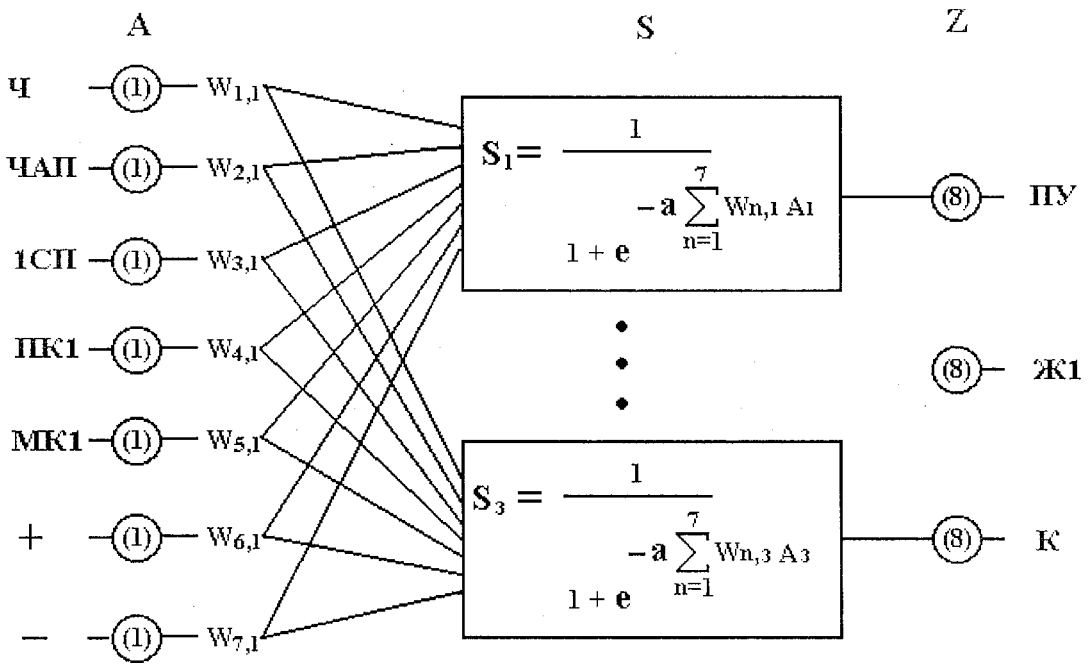


Рис. 2.15 Схема модели нейронной сети

Таблица 2.3

Расчитанные значения выходов нейронной сети

Событие	**Выходы**		
	ПУ1	**Ж1**	**К**
А0	**0.080500**	**0.106499**	**0.004772**
А1	**0.923900**	**0.086823**	**0.000002**
А2	**0.001428**	**0.787447**	**0.093767**
А3	**0.000948**	**0.131921**	**0.874473**

Таким образом, нейронная сеть настроена на выполнение основной функции - формирование команд управления при соответствующих манипуляциях на пульте управления. Однако по условиям безопасности необходимо сменить показания светофора на запрещающее, при сбоях в работе устройства, неисправностях, и т.д. Для выполнения этого требования необходимо обучить сеть на всех оставшихся комбинациях входного сигнала так, чтобы на выходе появлялась команда включения красного огня светофора. В этом случае

обучающая выборка будет состоять из $2^7 = 128$ пар (<текущий вход>, <желаемый выход>).

Задача несколько усложняется, при реализации маршрутного управления (Таблица 2.1). Событие А1 (нажатие кнопки начала маршрута) не приводит к формированию команды управления, однако информация о нем должна сохраниться в системе до прихода события А2 (нажатие второй кнопки), что необходимо для определения направления и трассы маршрута. Сформулируем принципы решения этой задачи на основе применения нейронной сети обратного распространения ошибки:

1. Установка на входе нейронной сети счетчика событий.

Процесс маршрутного передвижения является циклическим, следовательно, каждому из событий можно присвоить номер по порядку. Тогда при наступлении очередного события инкрементируется счетчик и его значение, вместе с полезными сигналами поступают на вход НС (Рис. 2.16), а после полной реализации маршрута счетчик сбрасывается.

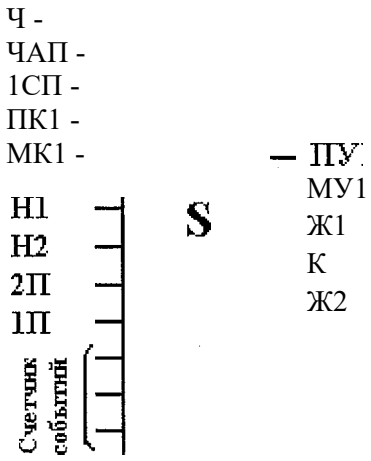


Рис. 2.16 Нейронная сеть со счетчиком событий

Таким образом, на входе сети представлена информация о текущем событии, во взаимосвязи с предыдущими и последующими.

К недостаткам этого метода следует отнести наличие внешнего устройства - счетчика, причем отдельный счетчик событий необходим для каждого маршрута.

Кроме того, в сети построенному по такому принципу, происходит возврат к идее конечного автомата, и сеть не может функционировать, если события нерегулярны.

2. Введение в нейронную сеть обратных связей.

Событие А1 (Таблица 2.2) не приводит к формированию управляющего сигнала, однако без этого события невозможна правильная классификация сетью последующих. Поэтому необходимо добавить по одному нейрону в первый и последний слои нейронной сети (Рис. 2.17, нейроны АЮ и Z6), выход нейрона Z6 соединяется с входом АЮ.

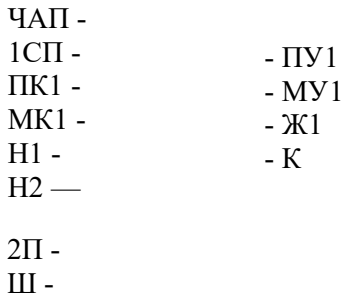


Рис. 2.17 Нейронная сеть с обратной связью

Недостатком этого метода является необходимость детального анализа работы системы централизации, что необходимо для разворачивания обратных связей при проектировании НС. Топология сети в таком случае будет индивидуальной для каждой станции. Кроме того, из литературы (например [4, 21,22]) известно о нестабильности сети с обратными связями.

3. Подача на вход нейронной сети всей последовательности событий.

В этом случае применяется подход, напоминающий работу стека в ЭВМ. После прихода очередного события предыдущее сдвигается «вверх» внешним устройством, а на его место записываются текущие данные, и все они подаются на входы нейронной сети (Рис. 2.18). Работа сети сводится к выделению из всей

последовательности событий только тех, для которых необходима генерация сигналов управления. Такой тип сети имеет значительно большее число входов (для приведенного примера 9*5=45), причем для реальных станций это количество может быть больше на 1-2 порядка, как следствие, значительно возрастает как время работы, так и процесса обучения.

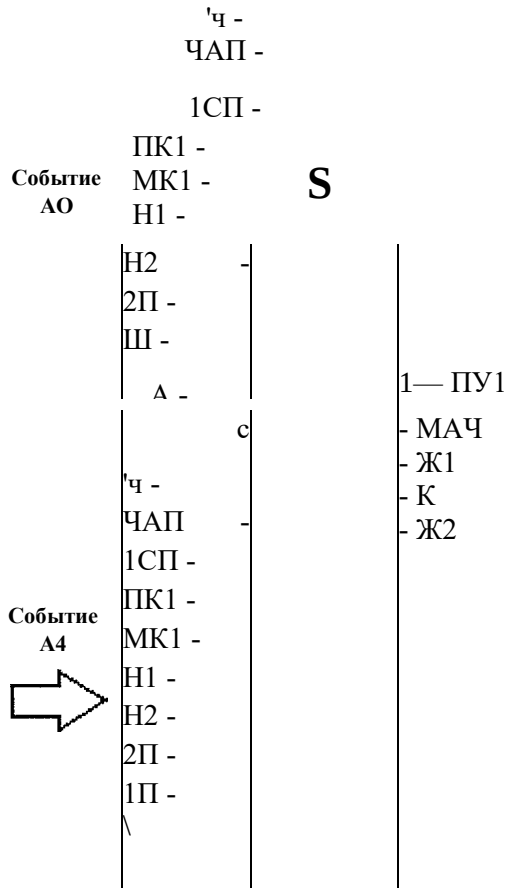


Рис. 2.18 «Стековая» нейронная сеть

Следует заметить, что из приведенных выше различных типов конфигураций, эта сеть является наиболее универсальной. При ее проектировании нет необходимости в установке счетчика для каждого из возможных маршрутов или обратных связей, как в приведенных выше примерах. При разработке модели нейронной сети, ориентированной на реальный полигон диспетчерского управления, априорной информацией являются данные о количестве входов и выходов, внутреннее строение сети остается одинаковым для станций с различным путевым развитием, конечно за исключением первого и последнего слоя нейронов.

В то же время известно, что информация о нажатии кнопок (Н, 41, 42 в рассмативаемом примере) мало подвержена искажениям, что нельзя сказать о контроле положения стрелок (ПК1, МК1) и состоянии путевых участков и стрелочных секций (4АП, Ш, 2П, 1СП). Приняв во внимание данные о надежности работы этих устройств [75, 111], следует детально остановиться на функционировании рельсовых цепей, т. к. именно эта информация наиболее подвержена искажениям (см. п. 1.3), а для решения задачи определения местоположения подвижных единиц она является наиболее важной. В связи с этим возникает необходимость в синтезе модели работы путевого участка, по информации системы диспетчерской централизации.

Выводы по разделу

В разделе произведен синтез общей динамической поездной модели и путевого развития района диспетчерского управления, определено содержание, выделены переменные. Рассмотрены первичные модели функционирования станционных устройств автоматики на основе теории конечных автоматов. Показаны ограниченные возможности применения этой теории. Обоснован выбор математического аппарата нейронных сетей. Произведен анализ функционирования наиболее распространенных типов НС и сделан выбор сети с обратным распространением ошибки (back propagation). Синтезированы модели функционирования устройств автоматики с применением сети такого типа. Рассмотрены принципы нейросетевого моделирования станционных устройств автоматики с маршрутным набором. Показана необходимость моделирования работы объекта типа путевой участок, по информации системы диспетчерской централизации.

РАЗДЕЛ 3
СИНТЕЗ ИНФОРМАЦИОННОЙ МОДЕЛИ ФУНКЦИОНИРОВАНИЯ ДАТЧИКА РЕЛЬСОВОЙ ЦЕПИ

3.1. Анализ временных характеристик источников первичной информации

Источником первичной информации о местонахождении подвижной единицы на элементе путевого развития, в системах электрической и диспетчерской централизации, является датчик - рельсовая цепь (РЦ). Как показано в предыдущем разделе, работоспособность моделей автоматики, а следовательно и общей динамической модели района диспетчерского управления зависит от достоверности данных, предоставляемых этим датчиком.

Известно, что рельсовая цепь по своей сути является двоичным устройством и может индицировать состояния «свободно» или «занято». Если рассматривать функционирование датчика во взаимосвязи с движением поезда (Рис. 3.1), то при идеальных условиях работы, длительность индикации состояния

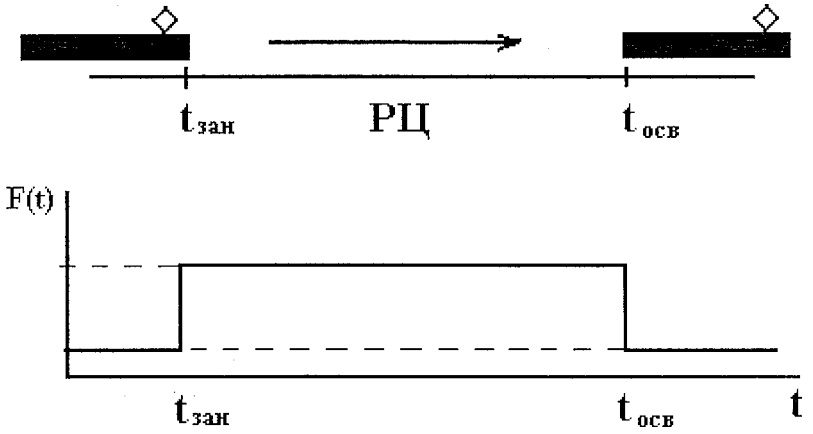


Рис. 3.1 Функционирование РЦ во взаимосвязи с движением поезда

«занято» соответствует времени проследования подвижной единицы по данной рельсовой цепи:

$$ИИ = ИП >$$ (3-й

где $1_П$ - длительность индикации состояния «занято», t_n - время нахождения поезда на рельсовой цепи. В свою очередь время t_n является разностью между

значением времени освобождения РЦ последней колесной парой поезда и временем занятия ее первой (л,бш):

$$l_{г,}\text{-}t_{,} \qquad (3.2)$$

Выразив $t_п$, по известной формуле, через скорость движения (Ц), длину поезда $(1_П)$ и длину рельсовой цепи ($/_т$) получим:

$$t_П \quad П$$

$$П$$

Однако, в формировании информации, поступающей в центр диспетчерского управления, кроме первичного датчика, принимают участие устройства ЭЦ, ДЦ, линии связи (Рис. 1.2, 1.3). В работе этих устройств, могут присутствовать временные искажения, обусловленные их принципом функционирования, элементной базой, а также возможными сбоями в нормальной работе (Рис. 3.2). С учетом этого, выражение (3.1) можно представить следующим образом:

$$_П \pm \Delta l\ , \qquad (3.4)$$

где А/ - разброс временных параметров.

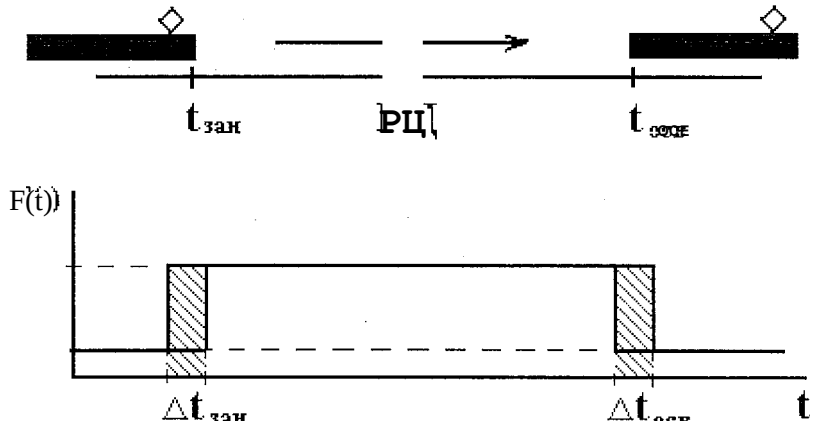


Рис. 3.2 Функционирование РЦ при наличии временных искажений

Параметр А/ является суммой временных искажений, возникающих при изменении показания датчика:

$$kt = kt + \text{АГ} \qquad (3\text{-}5)$$

где Дл.$_{о}$,, - искажения, возникающие при изменении индикации с «свободно» на «занято», $M_{жк}$ - при изменении с «занято» на «свободно».

На основе вышесказанного можно получить формулу для определения суммы временных искажений *At,* для чего подставим в формулу (3.4) выражение (3.3):

$$|\Delta t| = \frac{V_{П}}{l_{П} + l_{РЦ}} - t_{И} . \qquad (3.6)$$

Величина параметра Az связана с техническим состоянием устройств, принимающих участие в формировании сигнала от путевого датчика. Ее увеличение может служить показателем ухудшения временных характеристик работы элементов систем ЭЦ и ДЦ. Более детальное рассмотрение этого вопроса приведено ниже.

С другой стороны, если известен этот параметр для конкретного путевого датчика, то можно повысить достоверность информации о наличии подвижной единицы на данном элементе путевого развития. Для доказательства этого утверждения рассмотрим реальные условия движения поезда.

Известно, что длина блок-участка на перегоне, оборудованном автоматической блокировкой, составляет 1-1,5 км, и известна максимальная скорость движения, в качестве примера остановимся на значении 120 км/ч. Подставив эти значения в формулу (3.3), для короткой подвижной единицы (Ц=5 м) получим t_{n} « 30с.

Таким образом, если в результате измерения длительности состояния «занято» для блок-участка АБ, по информации системы ДЦ (/,, в формуле 3.6), полученное значение меньше 30 сек, то можно утверждать о наличии искажений. Аналогичные расчеты справедливы для всех РЦ, физическая длина которых не слишком мала. В этом случае значение t_{n} будет сравнимо с величиной разброса временных параметров работы элементов систем ЭЦ и ДЦ.

В области значений i_{n}, сравнимых по величине со временем

проследования поезда, выявление полезного сигнала по временному признаку является достаточно сложной задачей. Однако работа устройств автоматики

напрямую связана с движением поездов, и процесс изменения поездной ситуации на участке диспетчерского управления, представлен на графике исполненного движения (ГИД). Сравнив эту информацию с данными, получаемыми от системы ДЦ можно выявить наличие временных искажений в работе устройств ЭЦ и ДЦ, и следовательно повысить достоверность информации поступающей от системы диспетчерской централизации.

3.2. Анализ методов выявления временных искажений в работе рельсовой цепи по информации системы диспетчерской централизации

Основой организации движения поездов является график. График движения поездов объединяет в единое целое работу станций, локомотивных депо, тяговых подстанций, пунктов обслуживания и ремонта вагонов, дистанций пути, сигнализации и связи и других подразделений железных дорог, связанных с движением поездов, и обеспечивает их согласованную работу [53].

Графиком движения устанавливаются время прибытия, оправления или безостановочного проследования поездов по каждому раздельному пункту, время следования каждого поезда по перегонам, продолжительность нахождения локомотивов на конечных станциях их следования.

График движения представляет собой графическое изображение следования поезда. Введем обозначения и представим график как функцию от времени:

$$G(p_l,p_2,...,p,,r), \qquad (3.7)$$

где p, - местонахождение *i* -го поезда на участке железной дороги, причем *i* є [l,m], *m* - количество поездов за сутки. В свою очередь сигнал телесигнализации - функция состояния объектов контроля по времени:

$$S(n_{}),n,,. \qquad (3.8)$$

где /7, - состояние f-го объекта в момент времени /, *i* є [l,m], *m* - количество объектов. Тогда оценка достоверности информации в сигнале от путевых датчиков, состоит в определении степени сходства этих двух функций.

Эта задача может быть решена, если рассматривать каждую нитку на графике, как процесс занятия и освобождения участков пути, а для соответствующего объекта в сигнале ТС - как изменение его состояния.

В качестве примера рассмотрим фрагмент графика движения поездов (Рис. 3.3), где 6 - время отправления поезда со станции А соответствует значению времени изменения состояния первого участка удаления, /2 - второго участка удаления, и так далее для всех блок-участков на перегоне.

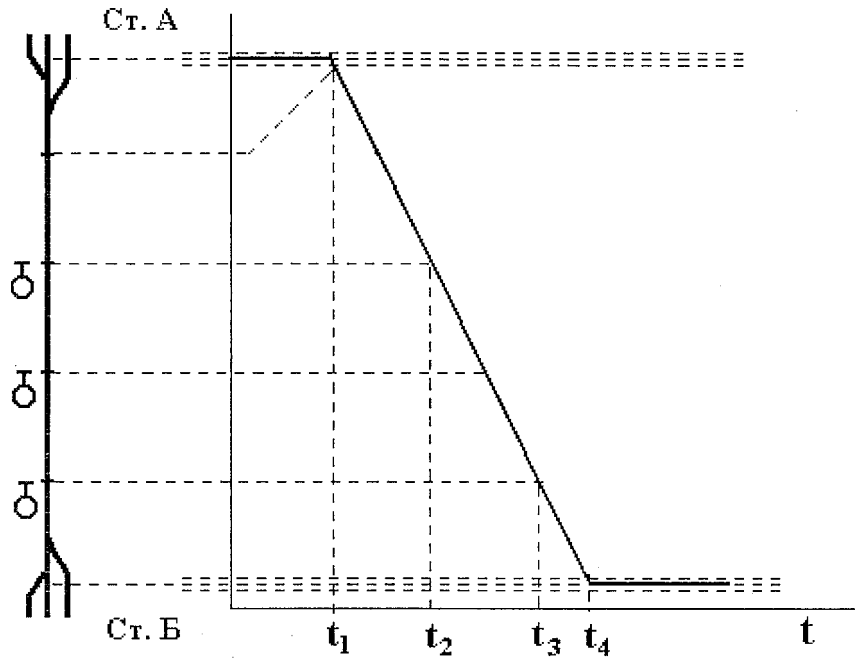


Рис. 3.3 Совмещение нитки на графике движения
с изменением состояния объектов в сигнале ТС

Время прибытия поезда на станцию Б (г4) соответствует времени занятия приемо-отправочного пути. При совмещении времени изменения состояния каждого объекта телесигнализации по всем ниткам на графике, можно определить степень сходства функций G и S. Обозначим ее как д, и выразим по известной формуле как разность квадратов интегралов этих функций:

$$\mu(t_i) = \sqrt{\left(\int_{t_1}^{t_m} G(t_1, t_2, ..., t_m)dt\right)^2 - \left(\int_{t_1}^{t_m} S(t_1, t_2, ..., t_m)dt\right)^2} \qquad (3\text{-}9)$$

Решением этого уравнения, на основе данных графика движения и сигналов ТС, за этот же период времени, будет множество значений времени. Сумма значений элементов этого множества может являться суммарной оценкой временных искажений по всем объектам телесигнализации типа путевой датчик:

$$(з.\overset{m}{\underset{/=|}{ю}})$$

В идеальном случае /ц = 0, что произойдет при полном совпадении данных графика с информацией, полученной от системы ДЦ, в противном случае решением выражения (3.10) будет суммарное время рассогласования этих данных по каждому из объектов телесигнализации. Вычисления по формулам 3.9 и 3.10 могут найти применение при оценке временных искажений в целом, по всем устройствам за 24 часа. Однако практическое применение такого подхода затруднительно, что связано, в первую очередь, с разным количеством объектов телесигнализации, на различных полигонах диспетчерского управления. Наиболее предпочтительно, в такой ситуации, вычисление среднее значение временных искажений:

где *m* - количество объектов.

Ограничением при расчетах по формуле (3.11) является то, что при сопоставлении с графиком движения поездов каждый объект телесигнализации типа путевой участок, должен находиться в состоянии «занято» такое же количество раз, сколько ниток представлено на графике за такой же период времени.

Рассмотрим графическое представление зависимости длительности индикации состояния «занято» и количества таких случаев по информации системы ДЦ (Рис. 3.4), где *t* - время нахождения объекта ТС в состоянии «занято», *n* - количество таких случаев за сутки. Пусть кривая, проходящая через эти точки, является графиком функции

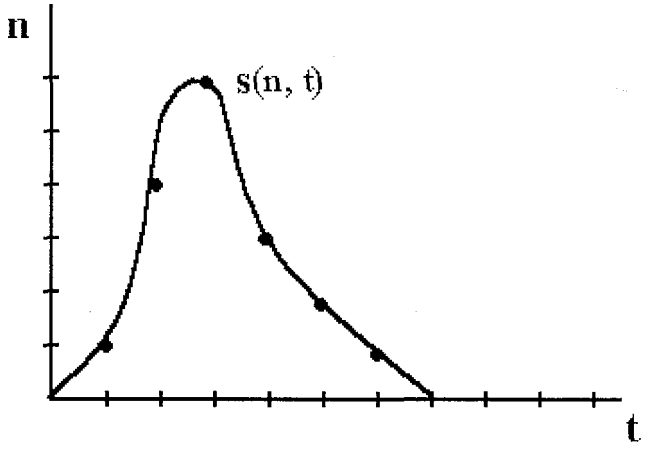


Рис. 3.4 График функции *s(n,t)*

График функции, в таких же координатах, можно построить и по информации графика движения поездов. В этом случае, по каждой нитке на графике можно определить как время занятия приемо-отправочных путей на станциях, блок-участков на перегоне, так и время их освобождения (формула 3.3).

Например, по нитке графика движения (Рис. 3.3), для участка удаления станции А время занятия Г совпадает с временем отправления, а время освобождения определяется по формуле:

$$I + 1$$

(3-12)

где 1_n - длинна поезда, 1_m - длинна участка удаления, V - средняя скорость движения поезда по перегону, которую можно определить по данным графика движения поездов (Рис. 3.3):

г/;

где /, - время отправления со станции А, $г_4$ - время прибытия на станцию Б, $/_{1/;}$ - расстояние между станциями А и Б).

Результатом вычислений, которые производятся для каждой нитки на графике движения будет множество *n* - пар значений времени

Разность этих значений $t^n = t'' - t''$ - время, в течении которого объект находился в состоянии «занято». Тогда, для каждого объекта существует функция g(/?,r),

отражающая его функционирование по данным графика движения, и оценка достоверности информации о состоянии i-го объекта в сигнале ТС (/./,) будет являться степенью сходства функций *g(n,t)* и .$■(«,/). Аналогично формуле (3.9), д, может быть найдена как разность квадратов интегралов этих функций:

$$A = \int_{V} \partial K \, JgOvM)^2 \sim(\qquad > \qquad (3\text{-}14)$$

Диапазон значений д, - от 0, при полном совпадении данных графика движения поездов и информации от системы ДЦ, до площади фигуры, ограниченной сверху графиком функции снизу осью 0 - *t,* что может произойти при отсутствии данных об объекте.

Следует отметить, что такой диапазон значений /у не совсем удобен в связи с тем, что нет универсальности в оценке искажений информации для различных объектов телесигнализации, и как следствие невозможность применения формулы (3.14) для системы в целом. К недостаткам так же следует отнести достаточную сложность вычислений, что в большей степени связано с аппроксимацией функций *g(n,l)* И

Частичным решением может быть сравнение непосредственно координат, и применение в качестве меры сходства среднеквадратичного расстояния между точками, которыми служат значения времени нахождения объекта в состоянии «занято» по данным графика движения и сигнала ТС. В этом случае оценка искажений в информации о i-том объекте:

$$\sqrt{\sum_{j=1}^{n} \left(t_{\Gamma j} - t_{T j} \right)^2}$$

где - значение интервала времени по данным графика движения, /7. - по информации ДЦ, *n* - количество интервалов. Очевидно, что такая оценка возможна при совпадении количества случаев изменения состояния объекта телесигнализации и числа ниток на графике движения поездов за такой же период времени. В то же время известно, что в сигнале ТС возможно как пропадание информации о движении поезда, так и появление ложного показания. В связи с

этим, в расчетах необходимо учитывать степень важности или весовой коэффициент для каждого значения t_r и t_T. Тогда выражение (3.15), с учетом вышесказанного, примет следующий вид:

$$\sqrt{\text{ЕШ-чХ}} \qquad \text{(зле)}$$

где n - количество интервалов времени по данным графика движения, m - по данным сигнала ТС, $к_у$ - весовой коэффициент.

Основная трудность при расчете по формуле (3.16) состоит в подборе весовых коэффициентов, количество и значения которых могут варьироваться в широких пределах, в зависимости от исходных данных, и они индивидуальны для каждого объекта телесигнализации. В настоящее время известны достаточно эффективные математические методы расчета такого рода коэффициентов. К таковым относится метод градиентного спуска, используемый в алгоритме обучения нейронных сетей (НС) (например [21]). Тогда задачу оценки временных искажений, в информации о состоянии объекта телесигнализации типа участок пути, можно сформулировать как распознавание (классификацию) образа кривой ,s(«,/) по эталону g(»,r) и использовать математический аппарат нейронных сетей. В этом случае, функционирование сети состоит из двух этапов:

- обучение, когда на вход нейронной сети подаются значения функции за предыдущий период, и по методу обратного распространения ошибки вычисляются значения весовых коэффициентов;
- нормальное функционирование (прямое распространение). В этом случае на вход сети подается текущее значение длительности состояния «занято», и вычисляется значение выхода, при рассчитанных на первом этапе весовых коэффициентах.

Естественно, что качество функционирования сети во многом зависит от проведения процесса обучения. Как показано в работах [4, 21,50, 51, 63, 102] некоторые типы сетей вообще не могут обучиться при некоторых значениях входов. Так, в частности, известна так называемая проблема «исключающего ИЛИ» и в работе [63] доказано, что однослойный персептрон имеет ряд

ограничений на круг решаемых задач. В связи с этим, прежде всего, необходимо рассмотреть процесс обучения нейронной сети, после чего, сформировать обучающую выборку по значениям функции получаемой из графика движения поездов.

3.3. Обучение нейронной сети участка пути по методу обратного распространения ошибки

Среди различных структур нейронных сетей (НС) одной из наиболее известных является многослойная структура, в которой каждый нейрон произвольного слоя связан со всеми аксонами нейронов предыдущего слоя или, в случае первого слоя, со всеми входами НС [21]. Когда в сети только один слой, алгоритм ее обучения очевиден, так как правильные выходные состояния нейронов единственного слоя заведомо известны, и подстройка синаптических связей идет в направлении, минимизирующем ошибку на выходе сети. По этому принципу строится, например, алгоритм обучения однослойного персептрона [83, 144]. Однако, как показано выше (п. 2.3.1), сеть такого типа не подходит для решения задачи выявления искажений в сигнале ТС. Это связано с тем, что на вход сети предполагается подача значений времени нахождения путевого датчика в состоянии «занято», и диапазон таких значений не должен быть ограничен. В связи с этим возникает необходимость в применении многослойной нейронной сети. Однако в многослойных сетях оптимальные выходные значения нейронов всех слоев, кроме последнего, не известны, и такую сеть уже невозможно обучить, руководствуясь только величинами ошибок на выходах.

Один из вариантов решения этой проблемы - разработка наборов выходных сигналов, соответствующих входным, для каждого слоя НС, что, конечно, является очень трудоемкой операцией и не всегда осуществимо [51]. Второй вариант - динамическая подстройка весовых коэффициентов, в ходе которой выбираются, как правило, наиболее слабые связи и изменяются на малую величину в ту или иную сторону, а сохраняются только те изменения, которые

повлекли уменьшение ошибки на выходе всей сети [21]. Очевидно, что реализация данного метода, несмотря на свою кажущуюся простоту, связана с значительным количеством вычислений.

Наиболее приемлемым, в такой ситуации вариант - распространение сигналов ошибки от выходов НС к ее входам, в направлении, обратном прямому распространению сигналов в обычном режиме работы, т.е. алгоритм обратного распространения ошибки.

Согласно методу наименьших квадратов, минимизируемой целевой функцией ошибки нейронной сети является величина [21]:

= (3.17)

где - реальное выходное состояние нейрона j выходного слоя N нейронной сети при подаче на ее входы p-го образа; d_{jp} - идеальное (желаемое) выходное состояние этого нейрона.

Суммирование ведется по всем нейронам выходного слоя и по всем предъявляемым на вход сигналам. Минимизация ведется методом градиентного спуска, что означает подстройку весовых коэффициентов следующим образом:

= (3.18)

где Wy - весовой коэффициент синаптической связи, соединяющей I-ЫЙ нейрон слоя п-1 с j-ым нейроном слоя п, /у - коэффициент скорости обучения, 0<?7<1.

Как показано в [146],

щ = (3.19)
^'4 ^./ ds .i d v u

где*yj* выход нейрона], - взвешенная сумма его входных сигналов, то есть аргумент активационной функции. Так как множитель *dyjldsj* является производной этой функции по ее аргументу, из этого следует, что производная активационной функция должна быть определена на всей оси абсцисс. В связи с

этим функция единичного скачка и прочие активационные функции с неоднородностями не подходят для рассматриваемой задачи.

В такой ситуации возможно применение гладких функций, таких как гиперболический тангенс или классический сигмоид с экспонентой. В случае гиперболического тангенса:

$$\wedge = 1-?. \qquad (3.20)$$
$$ds$$

Третий множитель в выражении 3.19 (ds/cW,;), равен выходу нейрона предыдущего слоя $y^{\wedge l)}$, а для первого множителя раскладывается следующим образом:

$$\text{ж}\ \_\text{у}\ \text{ж} \qquad =\text{у}\ \text{ж}\ 4\text{y}_{/\text{f н}},(\text{»}_{+}\text{p} \qquad (3\ 21)$$
$$tyj\ к\ \delta y\backslash\ ds_k\ \text{У}\ <\text{fy}_\text{k}\ ds_k\ ^{Jk}$$

где суммирование по к выполняется среди нейронов слоя п+1.

Введя новую переменную

$$\text{Т}'' = \text{—} \qquad (3.22)$$
$$dSj$$

можно получить рекурсивную формулу для расчетов величин $\delta\text{-}^{n>}$ слоя п из величин $\delta^{\wedge'+1)}$ более старшего слоя п+1:

$$= 2\text{L}^{\wedge}.''^{+\text{I})} \bullet \text{w'y}'' \qquad (3.23)$$

а для выходного слоя:

$$dSj \qquad (3.24)$$

Теперь можно записать (3.18) в раскрытом виде:

$$\text{Дм2}''' = -m\rfloor\text{-}\delta^{\wedge}\text{-}y^{\wedge} \qquad (3\text{-}25)$$

Таким образом, полученное выражение можно применять для подстройки весовых коэффициентов в процессе обучения сети. Однако для придания процессу коррекции некоторой инерционности, сглаживающей резкие скачки при перемещении по поверхности целевой функции, (3.25) можно дополнить значением изменения веса на предыдущей итерации [21]:

$$(\text{О} = \text{-}77\ \text{О}\ \text{Л1Ц}''(?\text{-}1) + (!\text{-}/<)\bullet\ \ \text{-X}^{\text{м}}), \qquad (3.26)$$

где /7 - коэффициент инерционности, t - номер текущей итерации.

Таким образом, полный алгоритм обучения НС с помощью процедуры обратного распространения строится так:

подать на входы сети значения функции §■(«,/), заранее полученных из графика движения поездов;

в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних (выражения 2.28, 2.29);

- рассчитать $^{N)} для выходного слоя по формуле (3.24);
- рассчитать по формуле (3.25) или (3.26) изменения весов $Aw^{(N)}$ слоя N.
- рассчитать по формулам (3.24) и (3.25) (или (3.24) и (3.26)) соответственно $^{п)} и для всех остальных слоев, n=N-l,...l;

скорректировать все весовые коэффициенты в нейронной сети:

$$4)(0 = ^G-1) + Aw^(0 \qquad (3.27)$$

рассчитать значение ошибки, и если она существенна - перейти на начало.

Если же значение ошибки удовлетворяет поставленным требованиям, то подстройка весовых коэффициентов закончена.

Из выражения (3.25) следует, что когда выходное значение стремится к нулю, эффективность обучения заметно снижается. При двоичных входных векторах в среднем половина весовых коэффициентов не будет корректироваться, поэтому область возможных значений выходов нейронов (0,1) желательно сдвинуть в пределы (-0.5,+0.5), что достигается простыми модификациями логистических функций. Например, сигмоид с экспонентой преобразуется к виду

$$/(x) = -0.5 + -Д^. \qquad (3.28)$$
$$1 + e$$

Кроме вышеизложенного необходимо оценить емкость нейронной сети, то есть число предъявляемых на ее входы данных, которые она способна научиться классифицировать. Для сетей с числом слоев больше двух, этот вопрос остается открытым. Как показано в [21], для НС с двумя слоями, то есть выходным и одним скрытым слоем, емкость сети Cd оценивается так:

(3.29)

где N_w - число подстраиваемых весов, N_y - число нейронов в выходном слое. Следует отметить, что данное выражение получено с учетом некоторых ограничений. Во-первых, число входов N_x и нейронов в скрытом слое N_h должно удовлетворять неравенству $N_x+N_h>N_y$. Во-вторых, $N_w/N_y>1000$. Однако вышеприведенная оценка выполнялась для сетей с активационными функциями нейронов в виде порога, а емкость сетей с гладкими активационными функциями, например (3.28), обычно больше [123]. Кроме того, полученная оценка емкости подходит абсолютно для всего возможного набора входных данных, которые могут быть представлены N_x входами. В действительности распределение входных данных, как правило, обладает некоторой регулярностью, что позволяет НС проводить обобщение и, таким образом, увеличивать реальную емкость. Так как распределение образов, в общем случае, заранее не известно, мы можем говорить о такой емкости только предположительно, но обычно она раза в два превышает емкость, полученную из выражения (3.29).

Рассматриваемая нейронная сеть имеет несколько "узких мест". Во-первых, в процессе обучения может возникнуть ситуация, когда большие положительные или отрицательные значения весовых коэффициентов сместят рабочую точку на сигмоидах многих нейронов в область насыщения. Малые величины производной от логистической функции приведут в соответствие с (3.22) и (3.23) к остановке обучения, что парализует НС. Во-вторых, применение метода градиентного спуска не гарантирует, что будет найден глобальный, а не локальный минимум целевой функции. Эта проблема связана еще с одной, а именно - с выбором величины скорости обучения. Доказательство сходимости обучения в процессе обратного распространения основано на производных, то есть приращения весов и, следовательно, скорость обучения должны быть бесконечно малыми, однако в этом случае обучение будет происходить неприемлемо медленно. С другой стороны, слишком большие коррекции весов могут привести к постоянной

неустойчивости процесса обучения. Поэтому, как показано в работе [21], значению 77 (3.25) обычно присваивается число меньше 1, но не очень маленькое, например, 0.1, и оно, может постепенно уменьшаться в процессе обучения. Кроме того, для исключения случайных попаданий в локальные минимумы иногда, после того как значения весовых коэффициентов застабилизируются, *г/* кратковременно увеличивают, чтобы начать градиентный спуск из новой точки. Если повторение этой процедуры несколько раз приведет алгоритм в одно и то же состояние НС, можно более или менее уверенно сказать, что найден глобальный максимум, а не какой-то другой.

Выбор конфигурации нейронной сети, количество слоев, нейронов, весовых коэффициентов, происходит ИСХОДЯ ИЗ необходимой емкости Cd (3.29). В связи с

этим, и на основе изложенных принципов обучения, необходимо рассмотреть возможные виды функций получаемых из графика движения поездов, и определить количество эталонных образов, на которых необходимо проводить процесс обучения.

3.3.Г. Анализ вариантов построения функции *g(nj.)* в зависимости от возможных искажений информации

Диапазон значений коэффициента искажений информации *i* -ого объекта //, (3.15, 3.16) находится в пределах от 0 до 1. Как правило, для такого рода коэффициентов, допустима точность до второго знака, в этом случае количество классов, подаваемых на вход нейронной сети в процессе обучения, должно быть не менее 10. Следовательно, после расчета данных по графику движения для эталонных значений, подаваемых на вход сети, необходимо производить генерацию еще девяти зашумленных [21]. Смысл таких значений заключается в следующем: как бы изменился эталонный образ, если степень искажений информации о состоянии объекта телесигнализации будет равным 0; 0,1; 0.2;...:0.9. Таким образом, в начале необходимо получить функцию *g(n.t)* по

данным графика движения поездов и затем, на основе анализа искажений канала телесигнализации, зашумленные сигналы.

Рассмотрим фрагмент графика движения поездов для перегона А-Б (Рис. 3.5). На графике представлено три нитки, допустим для двух нечетных грузовых и четного пассажирского.

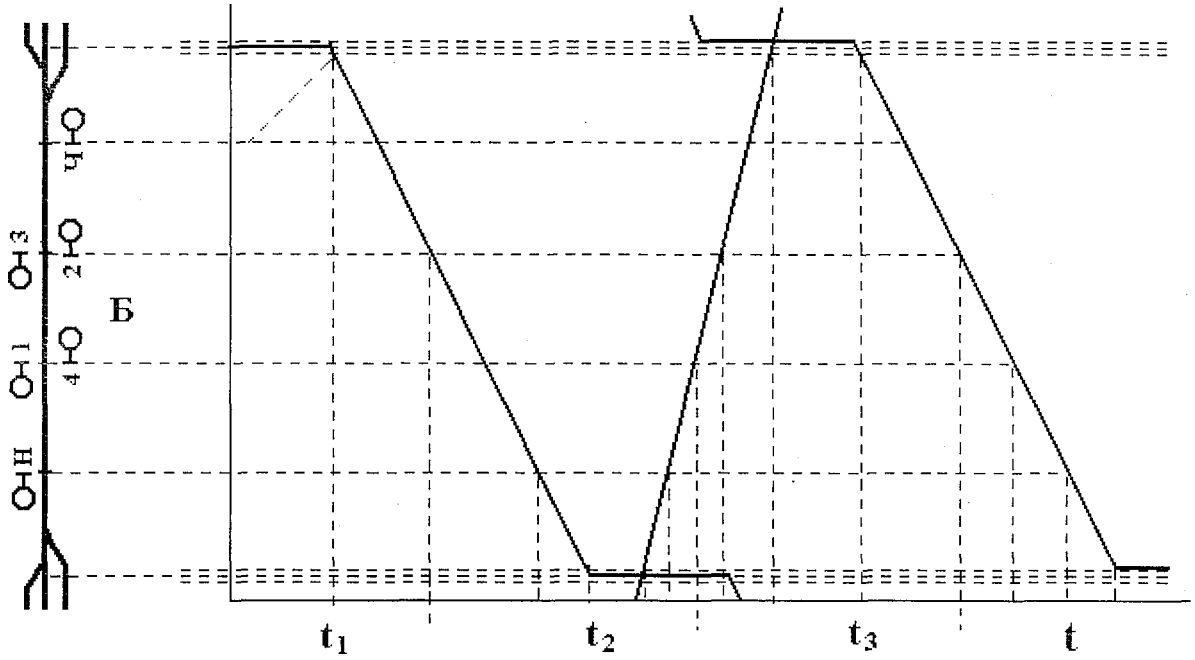


Рис 3.5. Совмещение изменения состояния блок-участков перегона А-Б с нитками на графике движения поездов

Пусть время занятия участка удаления станции А в первом случае составляет 10 мин, во втором 5 мин, тогда изображение функции g(/?,/) (эталонный образ), построенный по данным ГИД, состоит из двух точек (Рис. 3.6), и так как искажения об объекте отсутствуют, то коэффициент /./, = 1.

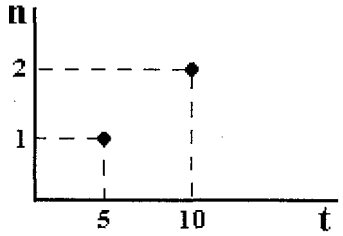


Рис. 3.6 Вариант изображения функции

Если в сигнале ТС, по какой-либо причине, отсутствуют данные об объекте, то графиком является прямая / = 0, и для этого случая коэффициент ц, = 0.

При возникновении искажений в сигнале ТС временного характера возможны различные варианты графика функции *g(n,t)* (Рис. 3.7 - 3.9).

Рис. 3.7 Вариант графика функции при наличии искажений
в информации о времени проследования пассажирского поезда

Рис. 3.8 Вариант графика функции *g(n,t)* при наличии искажений
в информации о времени проследования пассажирского и грузового поезда

Рис. 3.9 Вариант графика функции *g(n,t)* при наличии искажений
в информации о времени проследования всех поездов

Кроме того, в сигнале ТС возможно появление ложных показаний. Например, ложная занятость участка пути, несовпадающая по времени с

проследованием поезда, приведет к появлению на графике еще одной точки (Рис. 3.10).

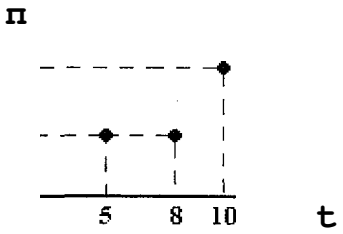


Рис. 3.10 Вариант графика функции *g(n,t)* при наличии ложной занятости участка в течении 8-й минут.

Не исключены случаи возникновения ложной занятости участка пути после проследования поезда. Так, например, если после проследования пассажирского поезда участок находился в состоянии занято 10 мин, то в этом случае график будет выглядеть следующим образом (Рис. 3.11).

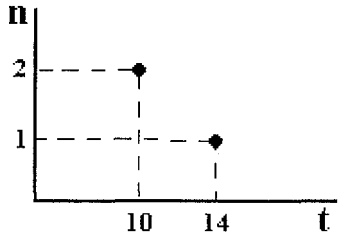


Рис. 3.11 Вариант графика функции *g(n,t)* при наличии ложной занятости участка после проследования пассажирского поезда в течении 10-и минут.

В случае возникновения более длительной неисправности, которая по времени совпадает с движением следующего поезда, информация о нем может быть искажена. Например, если ложная занятость возникла после проследования пассажирского поезда, и грузовой был отправлен со станции А без открытия выходного светофора (по приказу), то данные о времени занятия участка грузовым поездом искажаются (Рис. 3.12).

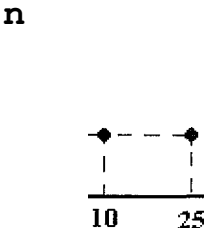


Рис. 3.12 Вариант графика функции *g(n,t)* при наличии ложной занятости участка, совпадающей по времени с проследованием грузового поезда

При движении поезда существует не нулевая вероятность возникновения искажения информации - ложная свободность участка пути. В этом случае, например, если при движении пассажирского поезда возникло такое искажение в сигнале ТС, длительностью 1 мин, то на графике это приведет к появлению точки (Рис. 3.13).

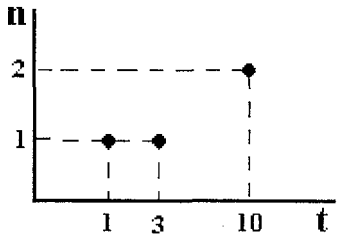


Рис. 3.13 Вариант графика функции *g(n,t)* при наличии ложной свободности участка во время движения пассажирского поезда

Ложная свободность, такой же длительности и во время движения грузового поезда, приведет к «дроблению» данных о времени проследования, и как следствие, к появлению дополнительных точек на графике функции *g(n,i)* (Рис. 3.14).

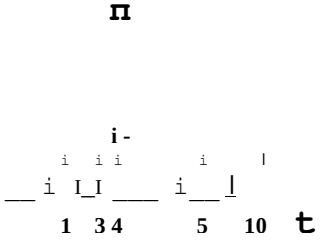


Рис. 3.14 Вариант графика функции *g(n,t)* при возникновении ложной свободности участка во время движения и пассажирского и грузового поездов

Исходя из этого, можно сделать вывод о достаточной степени сложности анализа, даже для такого простого примера с тремя поездами. Кроме того график движения представляет собой графическое отображение следования поезда, условно изображаемое прямыми линиями, соответствующими равномерному движению между раздельными пунктами. В действительности поезда следуют по перегону не равномерно с ускорением и замедлением на отдельных отрезках пути, особенно при отправлении поезда после стоянки или подходе к станции, где он должен остановиться [53].

В связи с этим, необходимо рассмотреть различные типы графиков и вид функции для возможных значений скорости следования поезда между раздельными пунктами.

3.3.2. Анализ вариантов построения функции *g(n,t)* в зависимости от различных типов графика движения поездов

Как показано в предыдущем разделе, для каждого объекта телесигнализации типа участок пути существует функция *g(n, t)*, отражающая его функционирование по данным графика движения поездов. Для определения вида этой функции необходимо рассмотреть различные типы графиков и возможные значения скорости следования поезда между раздельными пунктами.

В связи с тем, что поезда следуют по перегону не равномерно, а с ускорением и замедлением на отдельных отрезках пути, определим возможные значения скоростей движения при следовании поезда без отклонений от графика движения.

Для одной нитки на графике (Рис. 3.15) известными являются значения времени отправления со станции А (/„) и прибытия на станцию Б (/„), обозначим эти точки как M и N соответственно.

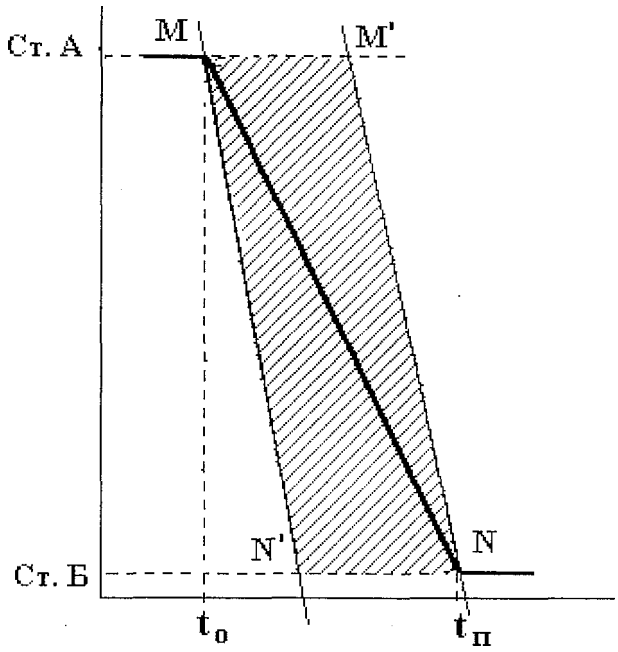


Рис. 3.15 Диапазон возможных скоростей
движения поезда по перегону А-Б

Проведем через эти точки прямые, соответствующие максимально возможной скорости движения на данном участке, так как поезд не может ее превысить на данном участке. Точки пересечения этих прямых с линиями осей станций обозначим как *M'* и IV'. В результате получим параллелепипед (заштриховано) возможных вариантов расположения нитки на графике и значений скоростей движения поезда по перегону А-Б, при следовании без отклонений от графика движения.

Совместим полученный график с путевым развитием при оборудовании перегона автоматической блокировкой (Рис. 3.16). Пересечение линий сигнальных установок, и прямых максимальных скоростей движения (M,A'), (Л/'.ЛД соответствует точкам, проекция которых на ось абсцисс является возможными значениями времени занятия блок-участка на перегоне. Например, для второго участка удаления от станции А, это точки 5, и S',, а их проекция - /, и $/_2$.

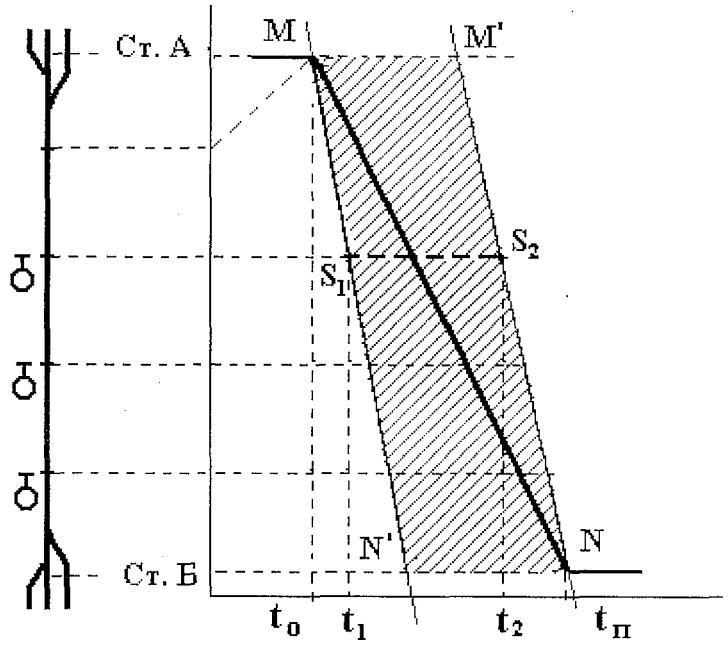


Рис. 3.16 Совмещение нитки на графике движения с изменением состояния объектов при возможных скоростях движения

При движении реального поезда без отклонения от графика движения, возможны различные скорости движения. Наибольшее ее значение (р,,.,х) не может быть больше максимально возможной на данном участке, а наименьшее (р,,,,,) близко к нулю, и тогда их разность:

$$^{Д}P = Cпах \text{ “} Pnin- \qquad (3.30)$$

В свою очередь, диапазон значений времени занятия участка пути можно определить по формуле (3.3), подставив значения $p_{,1ax}$ и p_{inn} из выражения (3.30).

При известной длине поезда (/я) и длине рельсовой цепи) получим:

$$\sim m;|X\,n\,\text{ТМ} \sim \frac{/_{\_}{}^{+}\,{}^{\wedge}РЦ}{y_{min}} \frac{{}^{+}\,{}^{\wedge}РЦ_{\_}}{y_{max}} \sim \frac{{}^{+}\,Ь{>}Ц\ \text{ХСпах} \sim \text{Спін})}{V_{>\min\max}} \quad ' \quad \frac{63311}{\text{С} \bullet\text{- } 7}$$

С учетом параметра А/, изображение функции g(>v), например, для фрагмента графика (Рис. 3.5) будет выглядеть следующим образом (Рис. 3.17).

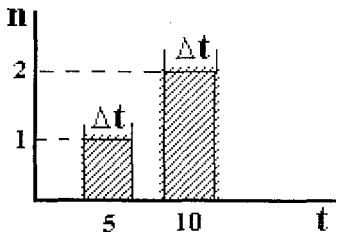


Рис. 3.17 Изображение функции *g(n,i),* с учетом А/

Для различных типов графиков изображения функции *g(n,t)* будут различаться. Известно, что при параллельном графике все поезда имеют одинаковую ходовую скорость, и линии хода поездов располагаются параллельно (Рис. 3.1 8 а).

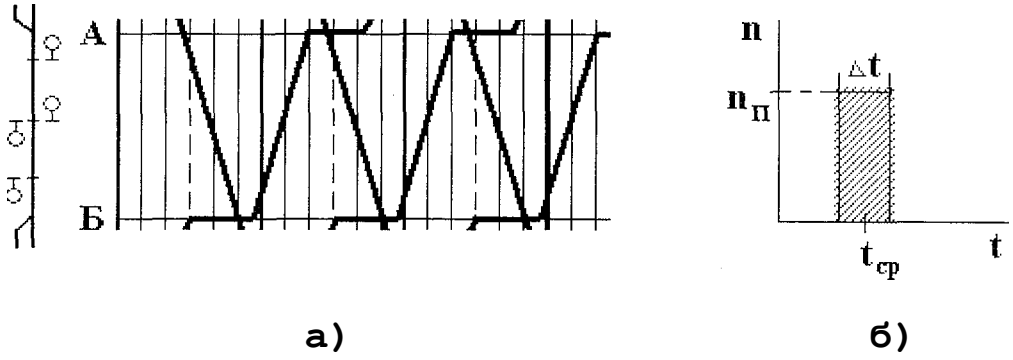


Рис. 3.18 а) Фрагмент параллельного парного графика однопутного участка; б) изображение функции *g(n,t)*

Для такого типа графика функция *g(n,t)* имеет вид прямоугольника (Рис. З. I 8 б) со сторонами:

- А/, диапазон значений времени занятия блок-участка;
- n_n, количество поездов, проследовавших по данному блок-участку.

При непараллельном графике предусматривается обращение пассажирских и грузовых поездов с разными ходовыми скоростями, причем каждый из этих видов поездов может быть одной или нескольких категорий (скорые, пассажирские, грузовые нормальной скорости, грузовые ускоренные и др.), имеющих разные скорости движения [53]. В этом случае функция будет совокупностью прямоугольников со сторонами А/ и n_n, различными для каждой

категории поезда. Так, на рис. 3.19 изображен фрагмент графика движения с тремя категориями поездов, условно обозначенными цифрами. Пусть имеется два поезда категории 1, один поезд категории 2 и два поезда категории 3. Соответственно график функции *g(n,t)* будет иметь вид трех прямоугольников, причем возможны различные варианты их взаиморасположения (Рис. 3.20).

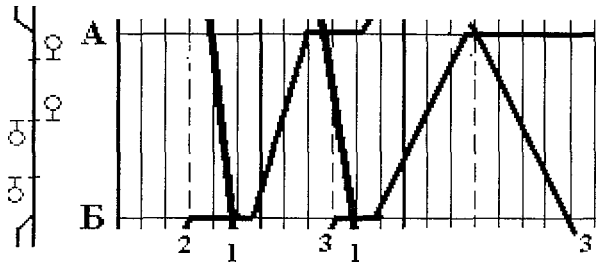


Рис. 3.19 Фрагмент непараллельного графика

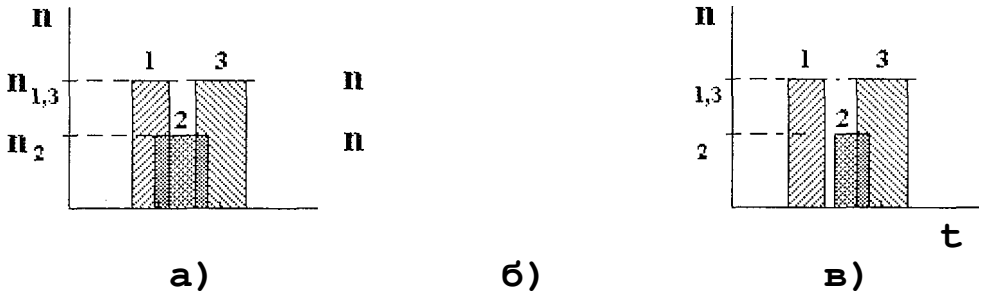


Рис. 3.20 Варианты изображения функции *g(n,i)*

Таким образом, для построения функции *g(n,t),* по данным графика движения, прежде всего, необходимо найти диапазон значений времени занятия путевого участка (Дг в формуле 3.31) для различных категорий поездов и ее среднее значение.

Из приведенного анализа следует, что существует воозможность построения функции *g(n,t)* по нормативному графику, однако этот процесс достаточно сложный и значения функции зачастую не будут соответствовать реальному поездному положению. В то же время известно, что реальная поездная ситуация на участке диспетчерского управления отражается на графике исполненного движения (ГИД). На основе этой информации, а также с использованием данных системы АСОУП, возможно построение графика функции *g(n,t),* и следовательно обучения нейронной сети каждого участка пути. Для этого необходимо

рассмотреть возможность применения нейронной сети для задачи классификации состояния блок-участка по информации, снимаемой с путевого датчика, определить оптимальную топологию и разработать методику ее первоначального обучения по данным ГИД.

3.3.3. Оценка возможности применения нейронной сети для классификации состояния блок-участка и определение ее топологии

Как показано выше, по длительности событий можно осуществить классификацию состояния объекта и произвести оценку достоверности информации, снимаемой с двоичного датчика. Следует заметить, что графики функции *g(n,t)* (рис. 3.17, 3.18, 3.20) справедливы только для одного конкретного блок-участка, в принципе эти данные можно обобщить, например, для всех рельсовых цепей типа блок-участок на перегоне, но при этом придется пойти на некоторые допущения, что неблагоприятно отразится на результатах моделирования. В любом случае необходимо производить набор значительного количества статистических данных о состоянии исследуемого устройства, что как правило связано с большими затратами времени и средств.

Для того чтобы оценить возможность применения нейронной сети, для задачи классификации состояния блок-участка по информации, снимаемой с путевого датчика, был проведен ряд экспериментов. В качестве программного обеспечения эмулятора нейронной сети применялся пакет «Back Propagation Neural Net Engine vl.33 by Patrick Ko Shu-pui» [121]. Было введено ограничение на диапазон значений времени, для чего проведено масштабирование входных значений. Это сделано со следующей целью: во первых, уменьшить число обучающих примеров, во вторых, для того, чтобы в режиме нормального функционирования проверить работоспособность сети на всех примерах, подаваемых на вход. Результаты работы эмулятора приведены в таблице 3.1

В качестве входных данных использовались масштабированные двоичные значения интервалов времени: (0..tl)=(0..4), (tl..t2)=(5..9), (t2. .tn)=(13.. 15),

значения выхода: «О» - ложная занятость блок-участка, «1» - проследование подвижной единицы.

Таблица 3.1

Результаты работы эмулятора нейронной сети

Вход	НС 4х5х3х1		НС 4х5х3х1		НС 4х15х5х1	
	Обучающая последов.	**Выход**	Обучающая последов.	Выход	Обучающая последов.	Выход
0000	0	**0.032000**	0	0.050314	0	0.016471
0001	-	**0.943829**	0	0.048776	0	0.014663
0010	0	**0.029994**	0	0.036411	0	0.009410
ООН	-	**0.935908**	0	0.042806	0	0.011144
0100		**0.105950**	-	0.738741	-	0.795947
0101	1	**0.922489**	1	0.877976	1	0.871350
ОНО	-	**0.061711**	-	0.740391	-	0.767491
0111	1	**0.889916**	1	0.874595	1	0.850020
1000	-	**0.041331**	1	0.871933	1	0.847187
1001	1	**0.862650**	1	0.918514	1	0.898396
1010	-	**0.033494**	--	0.848272	-	0.816589
1011	-	**0.772290**	-	0.899947	-	0.868122
1100	-	**0.056437**	-	0.072149	-	0.102127
1101	0	**0.137859**	0	0.80555	0	0.112749
1110	-	**0.052308**	0	0.067327	0	0.093350
1111	0	**0.103633**	0	0.078367	0	0.104402

Обучение производилось по неполной выборке, а результаты снимались для всех возможных состояний входа сети. В первом эксперименте обучающая последовательность состоит из 7-й пар значений <вход>,<выход>, нейронная сеть состоит из 4-х входов, 1-го выхода, 2-х скрытых слоев по 5 и 3 нейрона

соответственно в 1-м о во 2-м слое. Как видно из результатов работы, НС дает достаточно точное значение выхода для случаев, которые присутствовали в обучающей выборке и для диапазона значений t2-tn, но в остальных результат не правильный (достоверность 87%), что связано с недостаточным количеством обучающих примеров. Во втором эксперименте их количество увеличено до 11, в результате сеть дает правильные значения для всех возможных значениях входных сигналов (достоверность 93%). В третьем эксперименте увеличена емкость сети: в первом слое 15 нейронов во втором - 5, обучающая последовательность осталась та же. Результатом работы сети стало уточнение выходных значений.

Для применения в реальных условиях необходимо увеличить количество входов нейронной сети и произвести побитовое кодирование значения времени (Рис. 3.21), и значение каждого бита подавать на отдельный вход сети. Таким образом, возможна оценка состояния рельсовой цепи в реальном масштабе времени. Топология сети в данном случае будет зависеть от решаемой задачи. Если необходимо получать только оценку состояния одной РЦ, то достаточно 16-ти входов и 1 -го выхода.

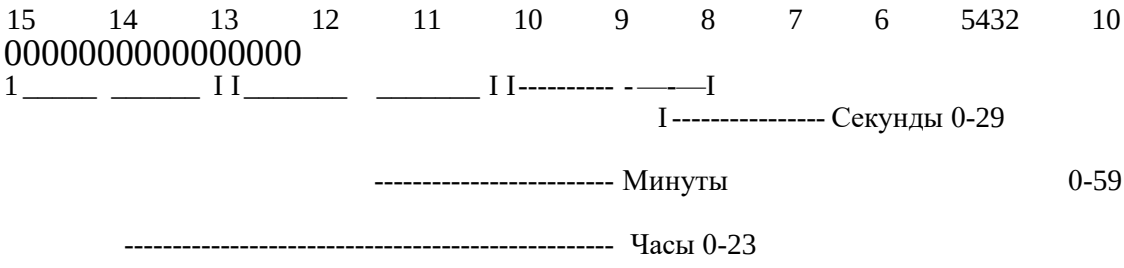


Рис. 3.21 Побитовое кодирование значения времени

Количество выходов такой сети должно соответствовать числу распознаваемых классов, а так как необходимо производить оценка степени искаженности сигнала, снимаемого с датчиков, то вполне достаточно одного выхода. Тогда если в результате функционирования выходное значение близко к единице, то произошло занятие участка пути подвижной единицей, в противном

случае - ложная занятость. Вполне естественно, что для согласования с логическими элементами по выходу необходимо предусмотреть пороговое устройство. Порог срабатывания элемента должен выбираться в процессе первоначального периода функционирования. Так, например, для полученных результатов работы эмулятора (Таблица 3.1) такой порог равен 0,7.

Таим образом, применение нейронной сети вполне оправдано. Далее, с учетом выбранной топологии, необходимо рассмотреть методику обучения нейронной сети по данным графика исполненного движения поездов.

3.4 Методика обучения нейронной сети по данным графика исполненного движения поездов и информации системы АСОУП

Известно, что для непрерывного контроля за движением поездов диспетчер ведет график исполненного движения (ГИД), на котором наносятся линии хода поездов по перегонам, отмечается время прибытия, отправления и проследования поездов по каждой станции, номера поездов, а также некоторые другие данные. Считается, что ГИД отражает реальную поездную ситуацию на участке диспетчерского управления (диспетчерском круге). На основе информации, предоставляемой графиком исполненного движения, а также по данным системы АСОУП, возможно вычисление значений времени занятия участка пути по формулам (3.2), (3.3), (3.30), (3.31), а следовательно и составление обучающей выборки нейроннойсети, возможность применения которой показана выше (п. 3.3.3). Наиболее целесообразно рассмотреть методику на примере фрагмента ГИД для одного из участков железных дорог Украины.

Двухпутный участок А-Б (Приложение В) оборудован автоматической блокировкой. Каждый перегонный путь разделен на два блок-участка. Расстояние между станциями - 5000 м (по осям станций). Рассмотрим участок удаления станции Б, имеющий физическую длину 1200 м.

1) Определение времени хода по перегону. По отметкам времени на графике исполненного движения определяется промежуток между отправлением со

станции и прибытием на соседнюю. Так, например, для поезда 6372 время хода по перегону составляет 10 минут.

2) Вычисление ходовой скорости. По полученным на предыдущем этапе данным времени хода каждого поезда и заданном расстоянии между станциями определяется ходовая скорость, например для поезда 6372:

3) Расчет длины поезда. На основе данных системы АСОУП, где имеется информация о количестве вагонов в каждом поезде, и исходя из длины вагона, рассчитывается длина каждого поезда. Так, длина поезда 6372, имеющего в своем составе 8 вагонов составит 136 метров (длина пассажирского вагона 17 м).

4) Расчет длительности занятия участка пути по данным ГИД. По ходовой скорости движения поезда, его длине, применяя формулы (3.2), (3.3) вычисляется значения длительности занятия участков пути. Например, занятие участка удаления станции Б поездом 6372:

$$t_{з} = \frac{l_{п} + l_{гц}}{V_{п}} = \frac{136 + 1200}{8{,}3} = 161(\text{с}).$$

предыдущих этапах составляется последовательность, состоящая из пар «вход-выход» нейронной сети данного путевого участка. Для примера (Приложение В) обучающая последовательность состоит из значений длительностей занятия участка удаления станции Б (Таблица 3.2). При этих значениях входа НС, выход сети должен равен «1» (проследование поезда). К выборке добавляется значение времени занятия - 0, значение выхода при этом будет равно «0» (ложная занятость).

Так как на начальном этапе обучения не известны длительности состояний ложной занятости участка пути, то все значения времени, которые больше максимального времени занятия подвижным составом, считаются искаженными и выход НС при этом равен «0» (строка 16 в таблице 3.2).

Таблица 3.2

Первоначальные значения обучающющей последовательности

№	Г, *(сек)*	Г, *(мин)*	Значения входов	Знач. выхода
1	161	3	0000000000000011	1
2	161	3	0000000000000011	1
3	108	2	0000000000000010	1
4	111	2	0000000000000010	1
5	118	2	0000000000000010	1
6	97	2	0 00 000 0 0 0 0 0 0 0 0 1 0	1
7	148	3	0000000000000011	1
8	206	3	0000000000000011	1
9	89	1	0000000000000001	1
10	88	1	0000000000000001	1
И	190	3	0000000000000011	1
12	96	1	0000000000000001	1
13	145	3	0000000000000011	1
14	404	7	0000000000000111	1
15	0	0	0000000000000000	0
16	max	max	1111111111111000	0

6) Исключение повторяющихся записей в обучающей выборке. Строки таблицы 3.2, в которой совпадают значения входа и выхода не несут новой информации в при обучении нейронной сети, следовательно их можно исключить из обучающей выборки. С учетом этого таблица пар значений вход-выход будет состоять из 6-й строк (таблица 3.3).

Таблица 3.3

Преобразованная последовательность значений обучающющей выборки

Значения входов	Значение выхода
0000000000000011	1
0000000000000010	1
0000000000000001	1
0000000000000111	1
0000000000000000	0.
1111111111111000	0

7) Обучение нейронной сети путевого участка. В процессе обучения нейронной сети на входы последовательно подаются значения из таблицы 3.3 и по методу обратного распространения ошибки (п. 3.3.3) происходит настройка весовых коэффициентов в соответствии с формулами (3.21 - 3.26).

После проведения процесса обучения нейронной сети она работоспособна. Однако применение нейромоделирования будет более эффективным, если в процессе эксплуатации производить дообучение сети. В этом случае будут учитываться все возможные значения времени занятия участков пути подвижными единицами. Данный механизм дообучения может быть реализован исходя из технологического процесса. Так, график исполненного движения составляется за 12-и часовую рабочую смену поездного диспетчера. Вполне естественно обучать нейронную сеть каждые 12 часов, по мере готовности ГИД.

С учетом вышесказанного возможно рассмотрение функционирования общей динамической модели района диспетчерского управления.

3.5 . Анализ функционирования общей динамической поездной модели района диспетчерского управления с учетом моделирования работы рельсовых цепей

Функционирование динамической поездной модели района диспетчерского управления состоит из нескольких этапов. На первом происходит обучение нейронной сети каждого путевого участка, что становится возможным после окончания рабочей смены диспетчера и составления окончательного графика исполненного движения. В процессе обучения НС используется информация ГИД, системы АСОУП, информации о неисправностях рельсовых цепей за текущий период работы (АРМ СЦБ) по методике описанной выше (п. 3.4).

Второй этап - текущее функционирование модели (Рис. 3.22).

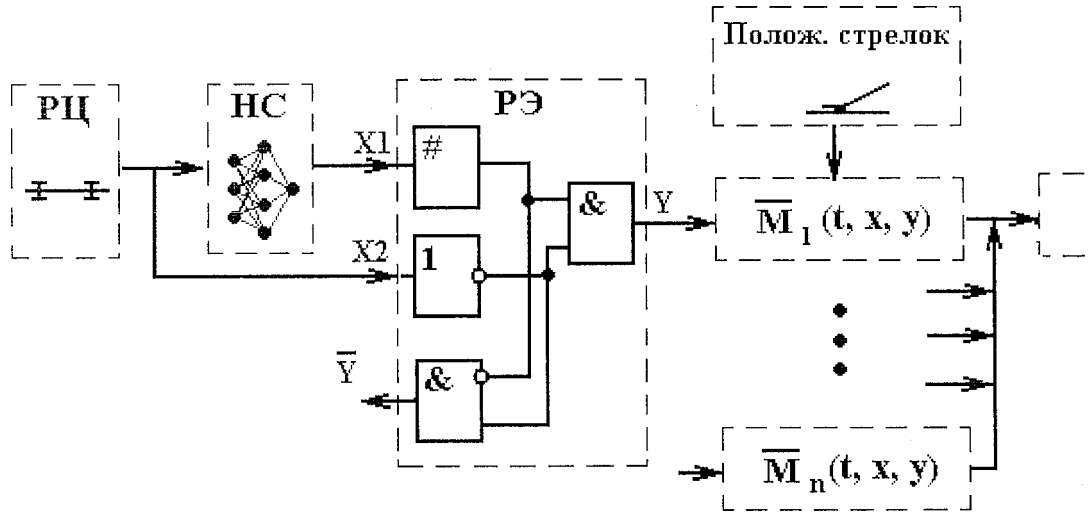


Рис. 3.22 Схема функционирования общей поездной модели

Сигнал занятия путевого участка, получаемый от датчика рельсовой цепи одновременно поступает на нейронную сеть (НС) и на решающий элемент (РЭ). Механизм функционирования и выбор топологии сети приведен выше (п. 2.3.3, п. 2.4, п. 3.3.3). Решающий элемент осуществляет операцию логического сложения двух сигналов - выхода нейронной сети и датчика рельсовой цепи:

$$У = А, л\ X_2.$$

Значения выхода НС действительные числа из диапазона [0,1], а на вход элемента «И» необходимо подавать логические уровни, следовательно возникает вопрос их согласования. Наиболее просто этот вопрос решается установкой порогового элемента, функционирование которого описывается следующей системой:

$$\backslash\backslash, ecnu U_{,lx} > U_{mp}$$
$$^{l\,rblx} \sim [Q, ecnu U_{BX} < u_{m>}/$$

где U - уровень срабатывания (порога). Пороговый элемент может быть реализован или в виде отдельного элемента (как показано на рис. 3.22) или модификацией функции возбуждения нейронов последнего слоя (выбор значения порога рассмотрен в п. 3.3.3).

Рассмотрим временную диаграмму функционирования решающего элемента при проследовании подвижной единицы по данному элементу путевого развития (Рис. 3.23).

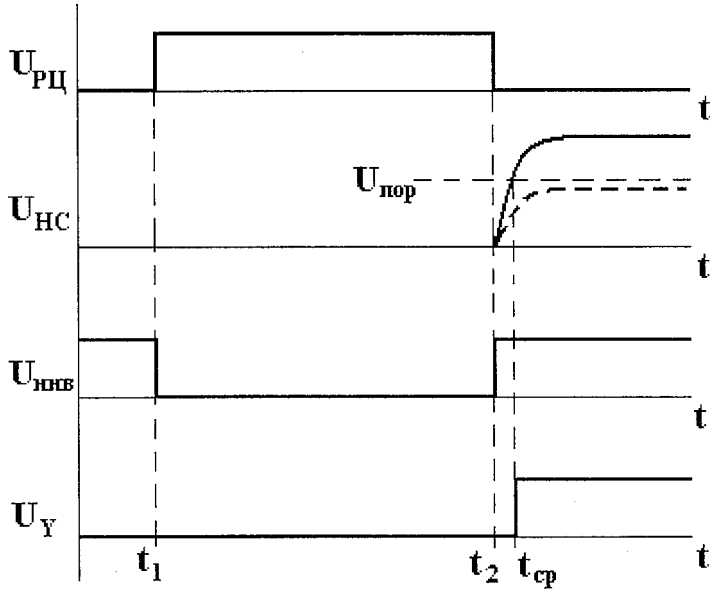


Рис. 3.23 Временная диаграмма функционирования решающего элемента при проследовании подвижной единицы

При срабатывании датчика рельсовой цепи уровень $(u_{1>u})$ на входе решающего элемента изменяется с «0» в «1» ф,). В этот момент времени запускается счетчик реального времени нейронной сети. После проследования

поезда, в момент времени у, уровень изменяется с «1» в «О». В случае распознавания полезного сигнала, на выходе НС появляется уровень, превышающий *U*, срабатывает пороговый элемент, и на выходе решающего элемента (C_y) появится уровень «1» в момент времени $1_{ср}$. В противном случае, если сигнал распознан как ошибочный, то пороговый элемент не срабатывает (штриховая кривая) и $U_Y = 0$. Этот сигнал можно использовать для выявления сбойных ситуаций в работе системы диспетчерской централизации (У на рис. 3.22)

Наличие сигнала =1 является основной предпосылкой функционирования модели движения поезда *M,,(t,x,y)* (2.3). Так, в момент времени / происходит присвоение значений переменным *t,X,y* исходя из физических координат рельсовой цепи, текущего времени. При движении поезда по станции, кроме информации о занятии путевых участков, необходимы данные о положении стрелок.

Таким образом, динамическая модель *W(t)* (2.6) содержит данные о местонахождении всех поездов, находящихся на районе управления, причем эти данные изменяются в соответствии с развитием реальной ситуации на полигоне.

Выводы по разделу

В разделе произведен анализ временных характеристик источников первичной информации о состоянии путевых участков, получены формулы для определения временных искажений в сигнале ТС системы ДЦ, рассмотрены ограничения при расчетах. На основе анализа методов выявления временных искажений в работе рельсовой цепи, сделан вывод о возможности применения нейронной сети для классификации состояния путевого участка по информации системы ДЦ. Произведено моделирование работы рельсовой цепи на основе НС и рассмотрено функционирование общей модели ДМР с учетом этой модели.

РАЗДЕЛ 4
СПОСОБЫ РЕАЛИЗАЦИИ ДИНАМИЧЕСКИОЙ ПОЕЗДНОЙ МОДЕЛИ РАЙОНА ДИСПЕТЧЕРСКОГО УПРАВЛЕНИЯ

4.1. Способы реализации общей модели ДМР в вычислительном центре управления железной дороги

В настоящее время центры диспетчерского управления в достаточной степени оснащены средствами вычислительной техники, в большей степени ЭТО универсальные ЭВМ. Как показано в первом разделе работы, специализированных вычислительных устройств, предназначенных для моделирования поездной ситуации района диспетчерского управления в полном объеме, в эксплуатации не находится. Так, существующая система АСОУП спроектирована в 80-90 годах на базе ЭВМ третьего-четвертого поколения, и сейчас не в достаточной степени отвечает возросшим требованиям, предъявляемым к ней. Эта проблема частично решается заменой аппаратных средств, однако наращивание вычислительных мощностей используется не достаточно эффективно, так как основное ядро - математическая модель, алгоритмы функционирования, остаются прежними или незначительно изменяются.

С другой стороны, системы диспетчерской централизации также является устаревшими на большинстве железных дорог Украины. В то же время опыт внедрения современных микропроцессорных систем ДЦ показывает, что кроме отображения информации в более удобном виде, записи журнала событий, и некоторых других, дополнительных средств, по сути ничего не изменяется в принципах обработки информации.

В предыдущих разделах работы показано, что уровень автоматизации оперативного управления процессом перевозок можно повысить за счет интеграции систем АСОУП и ДЦ, а в качестве математического аппарата

использовать предложенную динамическую модель района диспетчерского управления.

Естественно, что реализация предлагаемой модели должна производится с учетом имеющихся в центрах диспетчерского управления программно-аппаратных средств. Это операционные системы - Windows, UNIX, Novell, базы данных - Oracle, Anywhere. Структура программно-аппаратных средств реализации модели состоит из тех уровней: линейных предприятий (полигон ДУ), информационно-вычислительного центра (ИВЦ), центра диспетчерского управления (Рис. 4.1).

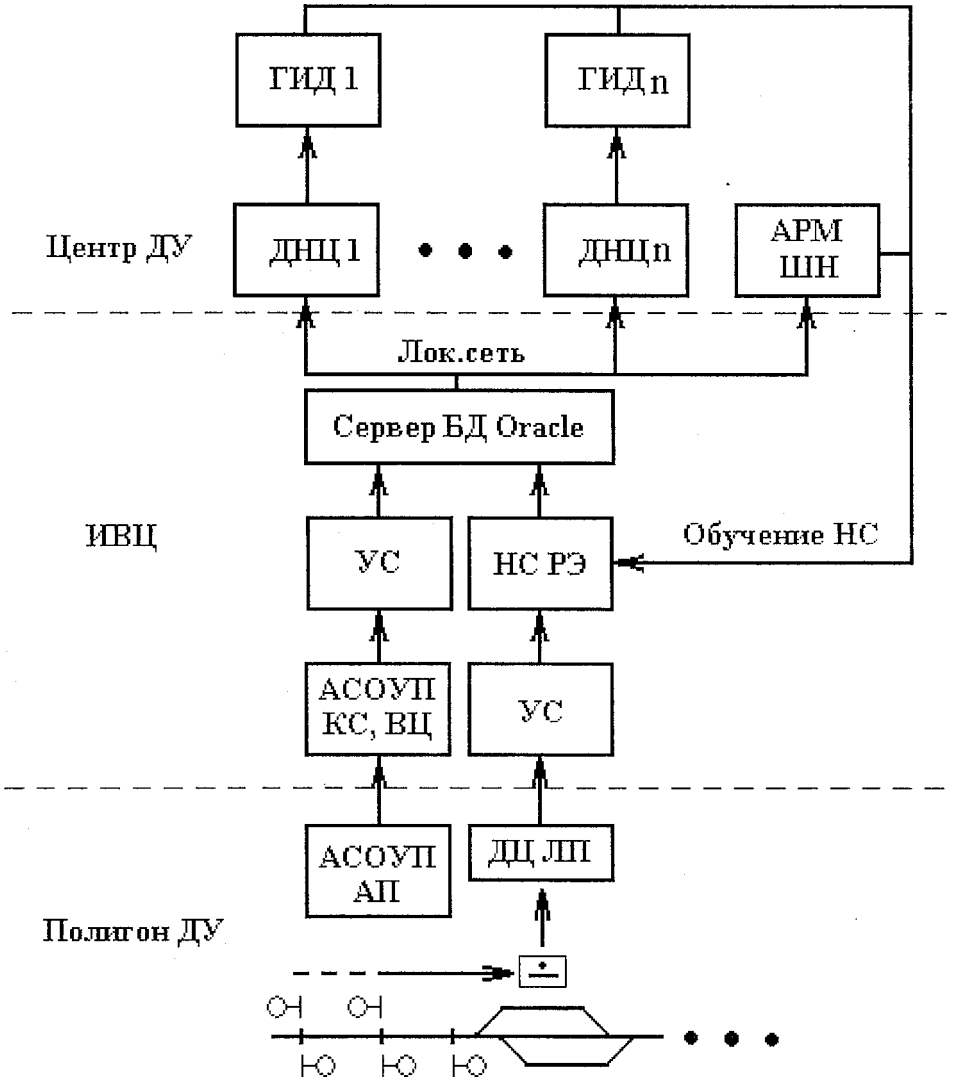


Рис. 4.1 Структурно-функциональная схема реализации модели ДМР

Оборудование полигона диспетчерского управления системами АСОУП и ДЦ приведено выше (п. 1.1,1.2). Вычислительные центры железных дорог Украины имеют достаточные вычислительные мощности, используемые в системе АСОУП (Рис. 1.5). Техническим средством, необходимым для реализации динамической модели, является устройство согласования, позволяющее осуществить взаимодействие с сервером баз данных (БД Oracle). Основная функция такого устройства - преобразование сообщений системы АСОУП в запрос базы данных. Например, для натурного листа поезда (ТНЛ) (сообщение № 2 АСОУП) используется только данные из служебной фразы, а именно номер поезда, код станции формирования, код станции назначения, дата и время передачи информации, условная длина. Запрос БД на языке SQL выглядит следующим образом:

а) Создание таблицы

```
CREATE TABLE BaseASOUP

        NUMER INTEGER,
        KODSTFORM INTEGER,
        KOD_ST_NAZN INTEGER,
        DATA TIME DATE,
        LEN INTEGER
■);
```

б) Изменение данных (Например, поезд 3012 Приложение В)

```
INSERT INTO BaseASOUP
(N UMER,KOD_ST_FORM,KOD_ST_N AZN, D AT A TIME,LEN)
VALUES(3012,4856,4800, ”08.07.1999 17:36”,48);
```

Результат отработки вызова - создание, а затем заполнение полей таблицы данных (Таблица 4.1) на сервере БД Oracle. После этого информация становится доступной пользователям, подключенным к локальной сети (Рис. 4.1).

Таблица 4.1

Содержание записи в таблице BaseASOUP БД Oracle

NUMER	KOD_ST_FORM	KOD_ST_NAZN	DATA_TIME	LEN
3012	4856	4800	08.07.1999 17:36	48

Кроме информации АСОУП на сервере БД находится данные, получаемые от системы диспетчерской централизации, как результат работы нейронной сети, решающего элемента (НС РЭ на рис. 4.1). Принцип работы такого узла описан выше (п. 3.5), а его реализация требует более подробного рассмотрения и приведена ниже (п. 4.2).

По сформированному в НС РЭ сигналу занятия путевого участка Y (Рис. 3.22), происходит формирование запроса БД:

```
INSERT INTO BaseDC
(KOD_SIGN,SOST)
VATUES( 121.01.04,0);
```

Где KOD_SIGN - код сигнала телесигнализации системы ДЦ. Для системы «ЛУЧ», наиболее удобным является формат записи:

район_круг_канал.группа.сигнал.

Например: 121.01.04 - 1-й район, 2-й круг, 1-й канал, 1-я группа, 4-й сигнал. В микропроцессорных системах ДЦ более предпочтительна сплошная нумерация объектов телесигнализации, так как в них, как правило, не существует разделения по группам, каналам. В любом случае, должна существовать справочная таблица базы данных, связывающая код сигнала ТС, с наименованием контролируемого объекта автоматики. Для системы «Луч» наиболее рациональной будет организация справочной таблицы в виде иерархической структуры (Рис. 4.2). По сути, в этой таблице находит отражение путевое развитие района диспетчерского управления, которое, в свою очередь, необходимо для функционирования общей динамической модели.

Как показано в п.2.1 изменение параметров вектора M (2.3) должно происходить при движении поезда по информации системы ДЦ об изменении

состояния участков пути. Механизм для реализации динамического изменения информации в базе данных предоставляется специальным средствам - триггерами БД Oracle. Так, возможно создание триггера, активизирующего свой программный код при изменении данных в таблице сигналов ТС (BaseDC, в описанном выше примере).

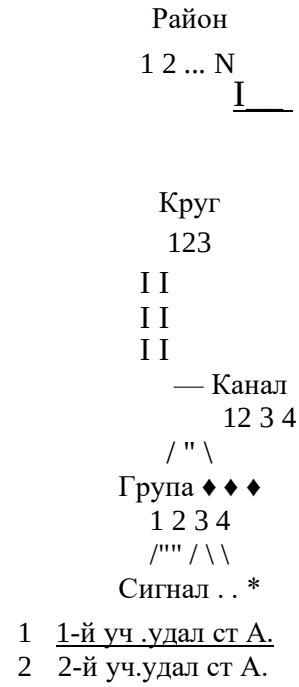


Рис. 4.2 Иерархическая структура справочной таблицы сигналов ТС

Программная реализация кода триггера производится с учетом условий (2.4, 2.5) и в зависимости от нахождения поезда на станции или перегоне. Результат работы триггера - изменение данных в таблице поездов находящихся на районе диспетчерского управления (Таблица 4.2).

Таблица 4.2

Имененные данные в таблице БД Oracle

NUMER	DATATIME	X	Y
3012	08.07.1999 17:36	893	2

Поля таблицы соответствуют содержанию вектора *M* (2.3), где NUMER - номер поезда, DATATIME - текущее время, X - расстояние от оси станции, до первой колесной пары головы поезда при движении по станции или его координата при движении по перегону, Y - номер пути, на котором находится поезд. Все записи в данной таблице и есть программно-аппаратная реализация динамической модели района диспетчерского управления *W(f)* (2.6).

По информации, получаемой программным обеспечением АРМа ДНЦ из базы данных, ведется график исполненного движения поездов. Предлагаемая модель позволяет автоматизировать этот процесс, и корректировки графика должны быть минимальны.

В свою очередь, информация ГИД необходима для обучения нейронной сети по методике, описанной в п. 3.4. Кроме того, результатом работы НС РЭ являются данные об неисправностях в устройствах автоматики, в частности ложная занятость участков пути. Эта информация также хранится в виде таблицы в базе данных и доступна по локальной сети программному обеспечению АРМа LL1H. На начальном этапе необходимо производить корректировку записей неисправностей, с целью выявления ложных, так как эта информация необходима для процесса обучения нейронной сети. По мере функционирования модели таких записей должно происходить меньше за счет адаптации НС к реальным условиям работы.

4.2. Способы реализации модели нейронной сети путевого участка для фиксации прохождения поездом стыкового пункта

В работе [84] исследованы различные методы фиксации проследования поездом межгосударственного и междорожного стыкового пункта. Показана возможность применения нейронной сети для обработки информации о состоянии датчика рельсовой цепи по данным системы диспетчерской централизации. Однако остался нераскрытыми вопросы построения такого типа устройства. В связи с этим, рассмотрим существующие способы реализации моделей нейронных

сетей. По данным, приведенным в работе Амосова Н. М. [4] выделяются 4 уровня нейрокомпьютеров.

Уровень 0. Теоретический. Работы, в которых в той или иной форме (математической, алгоритмической, словесной и т.д.) представлено описание моделей нейронных сетей.

4.2.1. ь 1. Программный. Модели нейронных сетей, программно реализованные на обычных последовательных компьютерах.

4.2.2. ь 2. Программно-аппаратный. Процессоры для ускорения моделирования нейронных сетей.

4.2.3. ь 3. Аппаратный. Физически реализованные модели нейронных сетей.

Специфичность нейросетевых операций, а также параллельность структуры и функционирования моделей нейронных сетей замедляют их реализацию на обычных последовательных компьютерах. В то же время, реализация нейронной сети на специализированном нейрочипе может быть экономически не выгодной по причине их большой стоимости. В связи с этим рассмотрим возможность реализации нейронной сети путевого участка на каждом из уровней.

4.2.4. Аппаратная реализация нейронной сети путевого участка

Аппаратные реализации нейронных сетей, где каждый нейрон моделируется отдельным обрабатывающим элементом, позволяют достичь очень высоких скоростей, однако трудность реализации большого числа связей ограничивают размер моделируемых сетей [4].

Одними из первых реализаций нейронных сетей были НС разработанная Уидроу [151] и ассоциативная матрица Стейнбуха [147], построенные на дискретных элементах.

В настоящее время интенсивно разрабатываются реализации нейронных сетей с помощью технологии СБИС. Так, однократно программируемые матрицы связей, где синапсы представляют собой тонкопленочные резисторы,

изготовляемые методом электронной литографии, разработаны в JPL и AT&T Bell Labs [149]. Однако такой тип сетей не предполагает дообучения (подстройки в процессе работы синапсов), поэтому не подходит для реализации нейронной сети путевого участка.

Группа из Калтеха в 1986 г. разработала программируемый нейросетевой чип с 22 нейронами и синапсами, принимающими значения +1, 0, -1. AT&T Bell Labs также изготовила КМОП кристалл с 54 нейронами и программируемыми синапсами, также принимающими три состояния. Такой тип также не подходит для реализации модели, так как необходимый диапазон значений выхода - действительные числа.

В конце 80-х годов на рынке появились коммерческие изделия. Характеристики некоторых коммерческих нейрокомпьютеров приведены в таблице 4.3.

Таблица 4.3

Характеристики некоторых нейрокомпьютеров

Модель, фирма	Количество нейронов, тыс. шт.	Количество связей, млн. шт.	Производительность, млн. связей/с
Mark III, TRW	65	1	0,45
Mark IV, TRW	256	5,5	5
Odyssey, TI	8	0,25	2
ANZA, HNC	ЗО	0,5	0,14 (0,025 с обучением)
Delta II, SAIC	1000	ЗД	10 (2,6 с обучением)

Последние два нейрочипа представляют определенный интерес, так как предполагают дообучение и предполагают операции с плавающей точкой и в принципе пригодны для реализации нейронной сети путевого участка. Однако, приняв во внимание их высокую стоимость, а также тот факт, что район диспетчерского управления имеет свыше 1000 путевых участков, применение нейрочипов экономически не выгодно.

4.2.5. Программно-аппаратная реализация нейронной сети путевого участка

Известно, что на рынке существуют относительно дешевые сигнальные процессоры, позволяющие оперировать с действительными числами и допускающие неоднократное перепрограммирование. Реализация нейронной сети на базе этих процессоров позволяет построить достаточно гибкую модель, как функционирующую автономно, так и в составе вычислительного комплекса.

Нейровычислители представляют собой мультипроцессорные системы с возможностью параллельной обработки, в структуре которых можно выделить две основные части:

» управляющую Host-ЭВМ, реализованную на основе обычной вычислительной системы с CISC или RISC микропроцессорами;

• виртуальное аппаратное средство, подключаемое к Host-ЭВМ посредством внутренних системных интерфейсов, выполняющее основные вычислительные операции.

Остановимся на особенностях аппаратной реализации нейровычислителя (НВ) с возможностью параллельной обработки, реализующие элементы нейросети. В основе построения НВ данного типа лежит использование сигнальных процессоров, объединенных между собой согласно определенной архитектуры, которая обеспечивает параллельность выполнения вычислительных операций. Как правило, такие НВ строятся на основе гибкой модульной архитектуры, которая обеспечивает простоту конфигурации системы и наращиваемость вычислительной мощности путем увеличения числа процессорных модулей или применения более производительных сигнальных процессоров (Рис. 4.3).

НВ данного типа реализуются в основном на базе несущих модулей стандартов ISA, PCI, VME. Основными их функциональными элементами являются: модуль матричных сигнальных процессоров (МСП), рабочая память, память программ, модуль обеспечения ввода/вывода сигналов (включающий АЦП, ЦАП и TTL линии), а также модуль управления, который может быть

реализован на основе специализированного управляющего сигнального процессора (УП), на основе ПЛИС или иметь распределенную структуру, при которой функции общего управления распределены между МСП. Для построения НВ данного типа наиболее перспективным является использование сигнальных процессоров с плавающей точкой ADSP2106x, TMS320C4x,8x, DSP96002.

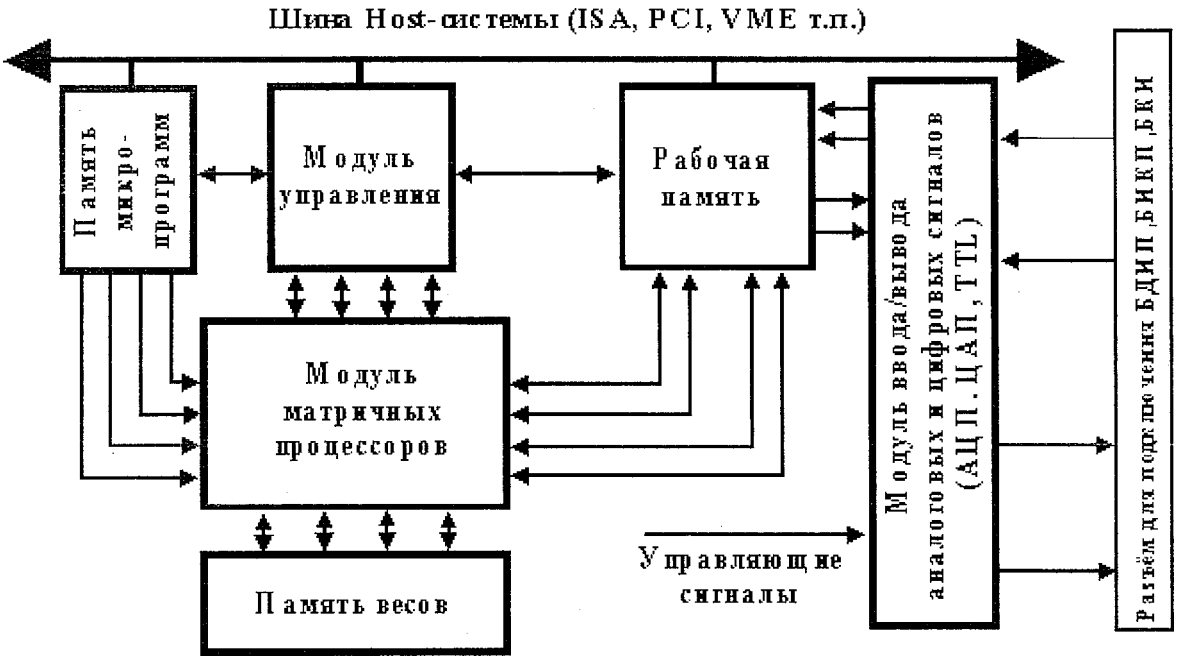


Рис. 4.3 Обобщенная функциональная схема виртуального НВ.

Структурная схема НВ на основе сигнальных процессоров TMS320C4x представлена на рис. 4.4. Несколько DSP, входящих в структуру НВ образуют распределенную вычислительную структуру из процессорных модулей, соединенных между собой высокоскоростными портами. Данный вариант реализации НВ может быть построен с использованием от двух до восьми сигнальных процессоров.

При использовании двух параллельных 32-разрядных DSP TMS320C40 обмен информацией при реализации нейросетевых алгоритмов осуществляется с помощью шести связанных портов с пропускной способностью в 30 Мб/с и каналов DMA каждого из процессоров. Поддерживая параллельную независимую работу, подсистема DMA и процессор обеспечивают параллельный обмен информацией со скоростями до 560 Мб/с.

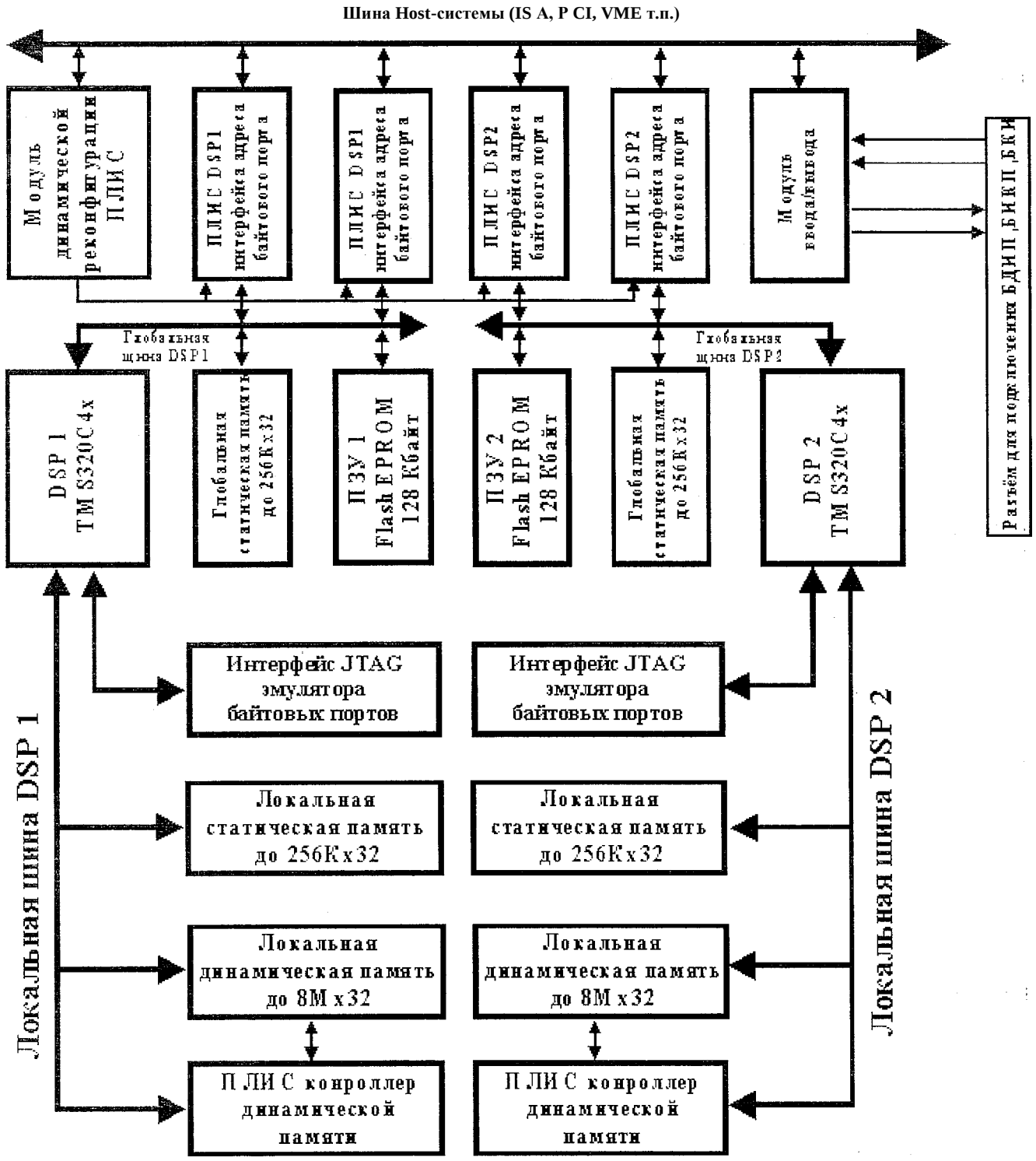


Рис. 4.4. Структура НВ на основе TMS320C4x.

При помощи высокоскоростных портов возможна реализация на основе данных DSP таких архитектур, как: кольца, иерархические деревья, гиперкуб и т.п. Каждая из локальных шин TMS320C40 обеспечивает обмен информации на скоростях до 120 Мбайт/с.

Еще больше повысить производительность НВ данного типа можно при использовании одного из самых мощных на сегодня сигнальных процессоров - TMS320C80 фирмы Texas Instruments. Данный процессор с производительностью в 2 млрд, операций в секунду представляет собой комбинацию из пяти процессоров, реализованных по MIMD (multiple-instruction, multiple-data) архитектуре. Впервые на одном кристалле реализованы одновременно две технологии - DSP и RISC, расположены один управляющий RISC процессор и четыре 32-х разрядных цифровых сигнальных процессора усовершенствованной архитектуры с фиксированной точкой (ADSP0-ADSP-3), обладающие высокой степенью конвейеризации и повышенной до 64 бит длиной слова инструкций, а это в свою очередь позволяет описывать сразу несколько параллельно выполняемых команд.

4.2.6. Программная реализация нейронной сети путевого участка

Как видно из формул (2.8, 2.9, 2.10, 2.11, 2.12, 2.13), описывающих алгоритм функционирования и обучения НС, весь этот процесс может быть записан и затем запрограммирован в терминах и с применением операций матричной алгебры. Такой подход обеспечит более быструю и компактную реализацию НС, нежели ее воплощение на базе концепций объектно-ориентированного (ОО) программирования. Однако в последнее время преобладает именно ОО подход, причем зачастую разрабатываются специальные ОО языки для программирования НС, хотя универсальные ОО языки, например C++ и Pascal, были созданы как раз для того, чтобы исключить необходимость разработки каких-либо других ОО языков, в какой бы области их не собирались применять.

Программирование НС с применением ОО подхода, позволяет создать гибкую, легко перестраиваемую иерархию моделей НС. Исходя из вышеизложенных соображений, на основе подхода предлагаемого в [21] создана библиотека классов и программ, реализующая полносвязные НС с обучением по алгоритму обратного распространения, использующая ОО подход.

Необходимо отметить, что библиотека составлена для решения задачи эмуляции нейронной сети участка пути для сигнала ТС диспетчерской централизации «Луч», однако может применяться и для работы с сигналами других систем ДЦ.

С применением пакета Microsoft Visual C++ 6.0 созданы два базовых класса: CNeuron, NetBP (Приложение Г).

Описание класса CNeuron выглядит следующим образом:

```
class CNeuron : public CObject
s
з
private:
        double F(double Par); //Функция возбуждения
        UINT n_Layer; //Номер слоя, в котором находится этот нейрон
UINT п_Неигоп!пТауег;//Порядковый номер нейрона в слое
double Randomize(); //Рандомизация весовых коэффициентов
public:
        BOOL LearnOK;   //Флаг окончания обучения
        void Impuls();      //Импульс при неудачном обучении
        double MiuPar;     //Параметр
        double NiuPar;     //Параметр
        void Update();      //Обновление весовых коэффициентов
        void Learn();       //Расчет значений весовых коэффициентов
        CNeuron** in_Neuron; //Указатель на предыдущие нейроны
        void SetCalcErr(double Par); //Обновление значения ошибки
        double CalcEri'O; // Расчет значения ошибки
        double* Synaps; //Массив весовых коэф.по входу (синапсов)
        UINT n_Synaps; //Количество синапсов
        double* Delta; //Массив ошибок по каждому синапсу
        UINT nJDelta;      //Параметр delta
        double Error;       //Значение ошибки
        CNeuron* GetPointer(); //Возвращает указатель на класс
```

но

```
void Go();          //Прямое распространение
double Axon; //Значение выхода нейрона (аксон)
BOOL lnit(UINT n_L,UINT n_N,UINT n_A); //Инициализация
CNeuronQ;           //Конструктор класса
virtual -CNeuronQ; //Деструктор класса
```

Описание класса NetBP выглядит следующим образом:

```
class NetBP : public CObject
{
 private:
        UINT* NeuronINLayers; // Количество нейронов в каждом слое
        UINT        CountLayers;       // Количество слоев в сети
        CNeuron**          N;          // Указатель на массив нейронов
        CNameList          m_FileLst;
        CdaoDatabase m_MyNetDB;
        BOOL       LoadFileListQ;
        BOOL       OpenFullDBQ;
        void MessageBox(CString Str);
        void DisplayDaoException(CDaoException *e, LPCTSTR Ipcszlnfo);
public:
        UINT CountCikl; //Максимальное количество циклов обучения
        double NiuPar; //Параметр ту
        UINT Countlter; // Максимальное количество итераций
        void ImpulsQ; //Функция «Импульс»
        void MsgWork(double* in_Array); //Рабочие сообщения
        void MsgLearn(UINT Count,double** in_Array,double** out_Array);
        void MsgToEditView(CString St);
        void Work(char* Name); //Рабочая функция
        BOOL LoadSynaps(char* №ше);//Чтение из базы синапсов
```

```
        BOOL SaveSynaps(char *Name);//Запись в базу значений синапсов
        void Upd(); //Изменение значений синапсов
        void CalcError(double* Mas); //Расчет ошибки
        void Go(double* Mas); //Прямое распространение
        BOOL    Learn(); //Обучение сети
        BOOL    Init(char* Name); //Инициализация сети
        NetBP();    //Конструктор
        Virtual     ~NetBP(); //Деструктор
};
```

Функционирование программы начинается с инициализации и выполнения стандартных функций Microsoft Visual C++. После этого производится инициализация классов из базы данных Access, в которой находятся таблицы:

- конфигурации нейронной сети (таблица Config);
- структуры сети (таблица Net);
- весовых коэффициентов (таблица Synaps);
- обучающей выборки (таблица Learn).

При вызове функции NetBP::Init(char* Name) происходит чтение таблиц базы и инициализация структуры сети (таблица Net), инициализируются нейроны каждого слоя, выполняется рандомизация (присвоение случайных значений) весовых коэффициентов. Кроме того, в процессе инициализации происходит присвоение значений параметрам 77 и д из базы данных, необходимых для проведения процесса обучения согласно формуле (3.26).

Функционирование сети начинается выполнением функции Learn() - обучения по методике, рассмотренной в п. 3.4. Обучающая выборка находится в таблице Learn базы данных Access. Функционирование сети сводится к подаче на ее входы текущего значения времени занятия участка пути и классификации этого значения.

Рассмотренный выше и реализованный в программе алгоритм является, классическим вариантом процедуры обратного распространения, однако по мере

функционирования происходит дообучение сети, а следовательно, и увеличение объема обучающей выборки. В результате такого процесса может наступить момент, когда емкости сети станет недостаточно. Априорное завышение емкости НС приводит к увеличению времени функционирования и обучения.

Для решения этой проблемы возможно применение генетических алгоритмов (ГА) функционирования нейронной сети [124]. В настоящее время теория функционирования таких сетей находится в стадии становления. Однако получаемые результаты позволяют надеяться о применении ГА для задачи классификации состояния путевого участка, в связи с этим необходимо более детально рассмотреть данный тип нейронной сети.

4.2.7. Перспективы применения нейронной сети, функционирующей по генетическому алгоритму для классификации состояния участка пути

Как отмечено выше, по мере функционирования нейронных сетей путевых участков, возрастает количество данных в обучающей выборке. Это может привести к тому, что вычислительная мощность сети, установленная априорно, перестанет удовлетворять поставленной задаче, т.е. наступит момент, когда сеть не сможет обучиться. Поэтому вполне логично применение таких алгоритмов функционирования, которые бы модифицировали, как обучающую выборку, так и внутреннее строение сети, подстраивая ее под усложнение исходной задачи.

Известно, что в настоящее время получил распространение генетический алгоритм (ГА) функционирования нейронной сети, являющийся представителем эволюционных алгоритмов. По своей сути является алгоритмом для нахождения глобального экстремума многоэкстремальной функции. ГА представляет собой модель размножения живых организмов.

Представим себе целевую функцию от многих переменных, у которой необходимо найти глобальных максимум или минимум:

$$/(x,,x_2,...,x,,) .$$

Для того чтобы заработал ГА, нам необходимо представить независимые переменные в виде хромосом. На первом этапе преобразуются независимые переменные в хромосомы, которые будут содержать всю необходимую информацию о каждой создаваемой особи. Имеется два варианта кодирования параметров:

- в двоичном формате;

® в формате с плавающей запятой.

Для задачи классификации состояния путевого участка применяется первый (рис. 3.21). Если параметр может изменяться между минимальным значением MIN и максимальным MAX, используем следующие формулы для преобразования:

$$\text{г} = \text{—} \frac{\text{c} \bullet (\text{max} - \text{min})}{2'' - 1} + \text{mm},$$

$$\& \frac{\text{г} - \text{min}}{(\text{max} - \text{min})(2'' - 1)}'$$

где g - целочисленные двоичные гены, $г$ - эквивалент генов в формате с плавающей запятой. Хромосомы в формате с плавающей запятой, создаются при помощи размещения закодированных параметров один за другим.

Если сравнивать эти два способа представления, то более хорошие результаты дает вариант представления в двоичном формате.

Генетический алгоритм работает следующим образом. В первом поколении все хромосомы генерируются случайно. Определяется их "полезность". Начиная с этой точки, ГА может начинать генерировать новую популяцию. Обычно, размер популяции постоянен.

Репродукция состоит из четырех шагов:

® селекции и трех генетических операторов (порядок применения не важен)

- кроссовер

® мутация

® инверсия

Кроссовер является наиболее важным генетическим оператором. Он генерирует новую хромосому, объединяя генетический материал двух

родительских. Существует несколько вариантов кроссовера, причем наиболее простым является одноточечный. В этом варианте просто берутся две хромосомы, и перерезаются в случайно выбранной точке. Результирующая хромосома получается из начала одной и конца другой родительских хромосом.

001100101110010111000		0011001011100107*1100*
110101101101000\11100		

Мутация представляет собой случайное изменение хромосомы (обычно простым изменением состояния одного из битов на противоположное). Данный оператор позволяет более быстро находить ГА локальные экстремумы с одной стороны, и позволяет "перескочить" на другой локальный экстремум с другой.

00110010111001011000 -------:	00110010111001111000

Инверсия инвертирует (изменяет) порядок бит в хромосоме путем циклической перестановки (случайное количество раз), однако многие модификации ГА обходятся без данного генетического оператора.

00110010111001011000 -------:	11000001100101110010

Так, генетический алгоритм на несколько порядков превосходит по быстроте случайный поиск во многих задачах [124]. Дело здесь в том, что большинство систем имеют довольно независимые подсистемы. Вследствие этого, при обмене генетическим материалом часто может встретиться ситуация, когда от каждого из родителей берутся гены, соответствующие наиболее удачному варианту определенной подсистемы (остальные "уродцы" постепенно вымирают). Другими словами, ГА позволяет накапливать удачные решения для систем, состоящих из относительно независимых подсистем (контролируемые объекты автоматики в системе ДЦ). Соответственно можно предсказать и когда

ГА скорее всего даст сбой (или, по крайней мере, не покажет особых преимуществ перед методом Монте-Карло) — системы, которые сложно разбить на подсистемы (узлы, модули), а так же в случае неудачного порядка расположения генов (рядом расположены параметры, относящиеся к различным подсистемам), при котором преимущества обмена генетическим материалом сводятся к нулю. Последнее замечание несколько ослабляется в системах с диплоидным (двойным) генетическим набором [124].

4.3. Технико - экономическое обоснование

Технико-экономическое обоснование предполагает расчет экономического эффекта, получаемого в результате внедрения основных результатов работы. Рассмотрим составляющие такого рода эффекта.

Так, реализация общей модели позволит повысить уровень автоматизации оперативного руководства перевозочным процессом, за счет обеспечения персонала информацией о поездном положении на диспетчерских участках района управления. Это позволит отказаться от бумажного графика ДНЦ и вести график исполненного движения по информации, поступающей от устройств диспетчерской централизации и от системы АСОУП с помощью корректировочных воздействий диспетчера. Вследствие чего, снизится загрузка работников диспетчерского аппарата, и возможно, сократится штат работников. Однако, расчет экономической эффективности, для этого случая, выполнить невозможно без полномасштабного внедрения, и проверки на практике результатов работы.

Кроме этого, в модели предполагается выдача информации о неисправностях в рельсовых цепях и аппаратуре диспетчерской централизации (п.3.5). Эта, по сути, диагностическая информация позволит выявлять отказы и предотказные состояния устройств автоматики, что в свою очередь, приведет к сокращению эксплуатационных расходов на их содержание и обслуживание.

Как показано выше (п.4.2), для реализации модели нейронной сети путевого участка, выбран наиболее приемлемый по стоимости способ - программный. Кроме того,, в работе предложен более универсальный подход к математическому описанию функционирования путевых участков, что должно привести к сокращению программного кода, необходимого для реализации модели, что, в свою очередь, повлечет уменьшение материальных затрат на разработку и сопровождение программы.

В общем случае, так как реализация поставленной в диссертации цели (п.1.4) предполагает модернизацию существующей системы, то расчет получаемой в результате внедрения прибыли (С) можно найти как разность между стоимостями существующей ($C_{(.}$) и предлагаемой *(С,*) разработками:

$$C = C_{(:}\text{-}C_{П}. \qquad (4.1)$$

В связи с тем, что по результатам работы внедрено программное обеспечение, то согласно ГСТУ 32.0.10.003 [91], стоимость разработки программного продукта можно рассчитать следующим образом:

$$=1{,}05(з,,Т,,,_{(}(1 + /<,,)+Я,,,,)(1 + /<,,,,+/<,,,.,), \qquad (4.2)$$

где $З_{п}$, - среднемесячный фонд заработной платы коллектива разработчиков;

$Т_{Н1}$ - общие трудозатраты одной разработки;

$К_{п}$,- обязательные отчисления в фонды;

$К_{нв}$ - накладные расходы;

$Ё_{л}$ - плановые накопления;

$П_{мв}$ - прямые затраты.

Выразим общие трудозатраты ($Т_{ад}$) через количество строк программного кода, для чего введем коэффициент *К* (количество созданных строк кода в месяц), отражающий квалификацию программиста. Будем считать, что данный коэффициент одинаков как при выполнении существующего, так и внедряемого программного обеспечения. С учетом этого общие трудозатраты составят:

$$T_{НД} = \frac{K_{СТР}}{K}$$

где $K_{сп>}$ - количество строк кода в программном продукте. С учетом этого подставим в формулу (4.1) значения выражения (4.2), с учетом коэффициента $K_{::}$

$$C = I.05I_{З'„} \quad (1 + K'_{ю)} + П'_{ш} |(1 + K'_{„„} + K'_{„„})\text{-}1{,}05| Д, ^(1 + K^{е}_{„)} + П)1 + K';_{я} + /</„,),$$

где индекс 1-су шествующая разработка, 2-внедряемая.

Таким образом, полученное выражение позволяет производить расчет прибыли, получаемой в результате сокращения количества строк кода в программном продукте, при той же функциональности.

Выводы по разделу

В разделе рассмотрены способы реализации общей модели ДМР в вычислительном центре управления железной дороги. Разработаны средства, позволяющее осуществить взаимодействие с сервером баз данных (БД Oracle). Рассмотрены способы реализации модели нейронной сети путевого участка для фиксации прохождения поездом стыкового пункта. Разработано программное обеспечение, в котором реализован эмулятор нейронной сети датчика рельсовой цепи с обучением по методу обратного распространения ошибки. Рассмотрены перспективы применения нейронной сети, функционирующей по генетическому алгоритму для классификации состояния участка пути. В технико-экономическом обосновании показана возможность расчета прибыли, получаемой в результате сокращения количества строк кода в программном продукте, при сохранении той же функциональности.

ВЫВОДЫ

В диссертации приведено теоретическое обобщение и новое решение научной задачи, которая состоит в разработке динамической поездной модели района диспетчерского управления. Основные научные результаты, выводы и практические рекомендации работы такие:

1. На основании анализа структуры, технических средств и информационного обеспечения системы оперативного управления движением поездов, сделан вывод, что методы автоматизации, математические модели, программное и аппаратное обеспечения в полном объеме, не отвечают поставленной задаче. В связи с этим возникает необходимость усовершенствования существующей модели отслеживания дислокации подвижных единиц на элементах путевого развития района диспетчерского управления.

2. Разработана общая поездная модель, которая формируется в реальном масштабе времени на базе информации о состоянии участков и станций, и в отличие от существующих, более адекватно описывает движение поездов. Разработана модель путевого развития полигона, адаптированная под функциональность общей модели. При синтезе этих моделей целесообразно применять методы векторной алгебры.

3. На основании сравнения с теорией конечных автоматов обоснована возможность применения математического аппарата нейронных сетей для моделирования работы станционных устройств автоматики. Разработана нейросетевая модель функционирования при раздельном управлении стрелками и сигналами. На основании экспериментальных исследований рассчитан порог срабатывания нейронов последнего слоя, который равняется 0,7. Сформулированы принципы построения нейронных сетей для маршрутного управления.

4. Исследован временной признак функционирования путевых участков по информации системы диспетчерской централизации. Для большинства блок-участков автоблокировки, при установленной скорости движения, время их занятия не может быть меньше 30 сек. На основании этого, показана возможность повышения достоверности информации о наличии подвижной единицы на элементе путевого развития.

5. Разработана модель функционирования путевого участка. Доказано, что при синтезе моделей возможно применение, как методов интегрального исчисления, разностных уравнений, так и нейронних сетей.

6. Разработана модель функционирования путевого участка с использованием нейронной сети обратного распространения ошибки и методика

ее обучения по информации графика исполненного движения поездов. На основании экспериментальных исследований доказанная адекватность модели состоянию перевозочного процесса (достоверность 87% для сети с 5-ю нейронами во втором слое, и 93% с 15-ю нейронами).

7. Показана перспектива применения нейронной сети, которая функционирует по генетическому алгоритму. Составлены алгоритмы и программная реализация моделей на основе современных аппаратных средств и языков программирования высокого уровня.

Результаты диссертационной работы были использованы: при разработке автоматизированной системы снятия информации о прохождении поездов по межгосударственным и междорожным стыковым пунктам, в виде технических предложений относительно разработки алгоритмов и программного обеспечения; экспериментальных данных исследования искажений сигнала ТС системы диспетчерской централизации; методик расчета и моделирования прохождения поезда по элементам путевого развития полигона диспетчерского управления.

Кроме этого, результаты диссертационной работы были использованы при разработке программно-технического комплекса «АРМ электромеханика ДЦ ЛУЧ» на ИСЦ Донецкой железной дороги, и внедрены в опытную эксплуатацию на участке Мандрыкино-Рутченково Донецкой железной дороги.

Разработанная в диссертационной работе модель функционирования устройств автоматики используется в учебном процессе и дипломном проектировании студентов кафедры “Автоматики телемеханики связи и вычислительной техники” Донецкого института железнодорожного транспорта.

Внедрение результатов работы подтверждается соответствующими актами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абу-Мустафа Я.С., Псалтис Д. Оптические нейронно-сетевые компьютеры//В мире науки, 1987. N 5. С. 42-50.
2. Автоматизированные диспетчерские центры управления эксплуатационной работой железных дорог / Под ред. П.С.Грунтова. М: Транспорт, 1990. 288 с.
3. Аксенов И. Я. Кибернетика и транспорт. В сб. Кибернетика и транспорт. - М: «Наука», 1967. с. 11-68
4. Амосов Н. М., Байдык Т. Н., Гольцев А. Д. Нейрокомпьютеры и интеллектуальные роботы. - Киев: Наук, думка, 1994. - 272 с.
5. Андреев Г. И., Витчинка В. В., Остапенко С. Н. Особенности построения методического обеспечения управления развитием сложных систем специального назначения в современных условиях // Экономика и математические методы, 1999, 35, №2.
6. Ансофф И. Стратегическое управление. М.: Экономика, 1989.
7. Б. дел Рио. Некоторые вопросы использования управляющих машин на железнодорожном транспорте. В сб. Кибернетика и транспорт. - М: «Наука», 1967. с. 69-79.
8. Базилевич Л. А., Соколов Д. В., Франева Л. К. Модели и методы рационализации и проектирования организационных структур управления. Л.: ЛФЭИ, 1991.
9. Барцев С.И. Некоторые свойства адаптивных сетей (программная реализация).-Красноярск: Институт физики СО АН СССР, - 1987.
10. Барцев С.И., Охонин В.А. Адаптивные сети обработки информации. - Красноярск: Институт физики СО АН СССР, 1986. 93 с.
11. Берталанфи фон Л. Общая теория систем - критический обзор. В кн. Исследования по общей теории систем. М.: Прогресс, 1969.
12. Васильев В. И. Распознающие системы. // К.: “Паукова думка”, 1969— 291 с.

13. Васильев В. И., Коноваленко В. В., Горелов Ю. И. Имитационное управление неопределенными объектами. // К.: “Наукова думка”, 1989—216с.
14. Волкова В. Н., Денисов А. А. Основы теории систем и системного анализа. СПб.: Изд-во СПбГТУ, 1999.
15. Глушков В. М. Синтез цифровых автоматов. М.: Физматгиз, 1962.
16. Гольдштейн Г. Я. Оценка качества сложных систем // Сб. докладов симпозиума “Методы представления и аппаратурный анализ случайных процессов и полей”. Новосибирск, 1968.
17. Гольдштейн Г. Я., Ольшевский В. В. Вопросы оценки эффективности научно-технических предложений // Сб. докладов симпозиума “Методы представления и аппаратурный анализ случайных процессов и полей”. Вильнюс: 1972.
18. Гольдштейн Г. Я. Проблематика использования математических моделей в управлении экономико-производственными системами //Сб. трудов "Системный анализ в экономике". Таганрог: Изд-во ТРТУ, 2000, с. 68-78.
19. Гольцев А.Д. Яркостная сегментация изображения при помощи нейроподобной сети.//Автоматика - 1965 - N 5 - с. 40-50.
20. Гончаров Н. Е. Алгоритмизация маневровой и местной работы на станциях. В сб. Кибернетика и транспорт. - М: «Наука», 1967. с. 134-144.
21. Горбань А.Н. Обучение нейронных сетей. М.: СП Параграф. 1990.
22. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере. Новосибирск: Наука, 1996. 256 с.
23. Гринь В. Ф. Программная увязка управляющей вычислительной машины и диспетчерской централизации. В сб. Кибернетика и транспорт. -М: «Наука», 1967. с. 101-117.

24. Грунтов П. С., Бабченко С. А., Захаров И. Е. Автоматизированные системы управления на железнодорожном транспорте. Учебное пособие. Ч. II / Под ред. Грунтова П. С.; БелИЖТ, 1987. - 69 с.
25. Грунтов П. С., Бабченко С. А., Ткачев В. М. Формирование информационной и технической базы единого центра управления дорогой. - В. кн.: Проблемы централизации диспетчерского управления на железных дорогах: Межвуз. сб. науч. Ст. - Гомель: БелИИЖТ, 1985, с 5-21.
26. Грунтов П. С., Кейзер А. П. Исследование математических методов оптимального управления движением грузовых поездов и совершенствование режимов их вождения в АСУ. - Формирование информационной и технической базы единого центра управления дорогой. - В. кн.: Проблемы централизации диспетчерского управления на железных дорогах: Межвуз. сб. науч. Ст. - Гомель: БелИИЖТ, 1985, с 5-21.
27. Грунтов П. С., Михальченко А. А., Кузнецов В. Г. Автоматизированные системы управления на железнодорожном транспорте. Учебное пособие. Ч. Ш / Под ред. Грунтова П. С.; БелИЖТ, 1988. 80 - с.
28. Дж. Дооре, А.Р. Рейнблейн, С. Вадера. Пролог - язык программирования будущего. *И* М. "Финансы и статистика" 1990- 141с.
29. Джеффри Е. Хинтон. Как обучаются нейронные сети.// В мире науки - 1992-N 11-N 12 - с. 103-107.
30. Дорохов Е. Я. О выборе оптимального периода съема информации. В сб. Кибернетика и транспорт. - М: «Наука», 1967. с. 80-100.
31. Дубов Ю. А., Травкин С. И., Якимец В. Н. Многокритериальные модели формирования и выбора вариантов систем. М.: Наука, 1986.
32. Ефремов В. С. Классические модели стратегического анализа и планирования: модель ADL/LC. // Менеджмент в России и за рубежом, 1998, №1.

33. Загарий Г. И., Федюшин Ю. М. Моделирование процесса перевозок на железных дорогах Украины с помощью расширенных сетей Петри // Информационно-управляющие системы на железнодорожном транспорте. 4/1997, Харьков, с. 52-56.
34. Жуковицкий И. В. Использование метода максимума правдоподобия для повышения достоверности идентификации собранных маршрутов. Материалы конференции. Информационно-управляющие системы на железнодорожном транспорте. 4/1998, Харьков, с. 38-42
35. Заде Л. А. Основы нового подхода к анализу сложных систем и процессов принятия решений. В кн. Математика сегодня. М.: Знание, 1974.
36. Зиндер Е. 3. Новое системное проектирование: информационные технологии и бизнес-реинжиниринг. Ч. 3. Методы нового системного проектирования. // СУБД, 1996, № 2.
37. Иванченко А.Г. Персептрон - системы распознавания образов.// К.: Наукова думка, 1972.
38. Ивахненко А. Г. Долгосрочное прогнозирование и управление сложными системами. //К.: “Техника”, 1975—312с.
39. Ивахненко А. Г. Моделирование сложных систем: информационный подход. //К.: "Наукова думка", 1987, 136 с.
40. Ивахненко А. Г. Самообучающиеся системы распознавания и автоматического регулирования. // К.: "Наукова думка", 1969—349с.
41. Искусственный интеллект: справочник в 3-х книгах. // М.: "Мир", 1990.
42. Итоги науки и техники: физические и математические модели нейронных сетей, том 1, М., изд. ВИНИТИ, 1990.
43. К. Нейлор. Как построить свою экспертную систему. *И* М. "Энергоатомиздат" 1991- 287 с.
44. Карпунин М. Г., Любинецкий Я. Г., Майданчик Б. И. Жизненный цикл и эффективность машин. М.: Машиностроение, 1989.

45. Картавцев В.В. Нейронная сеть предсказывает курс доллара?// Компьютеры + программы - 1993 - N 6(7) - с. 10-13.
46. Кейзер А. П., Бабченко С. А. Структура и ведение динамической модели поездного положения в информационно-вычислительной системе ЕЦУ. - В. кн.: Проблемы централизации диспетчерского управления на железных дорогах: Межвуз. сб. науч. Ст. - Гомель: БелИИЖТ, 1985, с 51-60.
47. Кибернетика и транспорт. Сб. научных статей. - М: «Наука», 1968. 295с.
48. Комплексная автоматизированная система управления железнодорожным транспортом (АСУЖТ). Под ред А. П. Петрова. М., «Транспорт», 1977. 600 с.
49. Концепция и программа реструктуризации на железнодорожном транспорте Украины // Міністерство транспорту України, Державна адміністрація заліз. Транспорту. - К.: 1998. - 143 с.
50. Короткий С. Нейронные сети: основные положения.// www.orc.ru/-~stasson/menu.html
51. Короткий С. Серия статей для журнала 'Монитор". // www.orc.ru/-stasson/menu.html
52. Костенко О. А., Пархоменко Н. В. Принципы построения программной цифровой модели сортировочной станции и ее исследование на ЭЦВМ. В сб. Кибернетика и транспорт. -М: «Наука», 1967. с. 118-133.
53. Кочнев Ф. П., Сотников И. Б. Управление эксплуатационной работой железной дорог: Учеб. Пособие для вузов. - М.: Транспорт, 1990. -- 424с.
54. Кузнецов В. Г. Организация централизованного диспетчерского управления подразделениями технических служб в ЕЦУ. - В. кн.: Проблемы централизации диспетчерского управления на железных дорогах: Межвуз. сб. науч. Ст. - Гомель: БелИИЖТ, 1985, с 65-78.

55. Кузнецов В. Г. Повышение надежности оперативного управления железнодорожными направлениями и узлами в АСЦДУЭР. - В. кн.: Проблемы централизации диспетчерского управления на железных дорогах: Межвуз. сб. науч. Ст. - Гомель: БелИИЖТ, 1985, с 78-82.
56. Кузнецов В. Е. Представление в ЭВМ неформальных процедур. *И* М.: "Наука" 1989,—158с.
57. Куссуль В.М., Байдык Т.Н. Разработка архитектуры нейроподобной сети для распознавания формы объектов на изображении.//Автоматика - 1990 - N 5 - с. 56-61.
58. Левин М. Ш. Комбинаторика проектирования систем. // Автоматизация проектирования, 1997, №4.
59. Лигун А. А., Малышева А. Д. Математическая обработка результатов эксперимента. // Днепродзержинск: ДИИ, 1992—47с.
60. Лорьер Ж.-Л. Системы искусственного интеллекта. // М.: "Мир", 1991—342 с.
61. Масалович А.И. От нейрона к нейрокомпьютеру.// Журнал доктора Добба- 1992 - N 1 - с. 20-23.
62. Месарович М., Мако Д., Такахара И. Теория иерархических многоуровневых систем. М.: Мир, 1973.
63. Минский М., Пейперт С. Персептроны. М.: МИР, 1971. 261 с.
64. Мирошниченко В. М. О машинном представлении схем крупных железнодорожных станций. В сб. Кибернетика и транспорт. - М: «Наука», 1967. с. 178-206.
65. Михальченко А. А. Система информационного обеспечения централизованного управления - В. кн.: Проблемы централизации диспетчерского управления на железных дорогах: Межвуз. сб. науч. Ст. -Гомель: БелИИЖТ, 1985, с 43-51.
66. Михальченко А. А. Формализация процесса диспетчерского управления и принципы оптимизации управляющих решений - В. кн.: Проблемы

централизации диспетчерского управления на железных дорогах: Межвуз. сб. науч. Ст. - Гомель: БелИИЖТ, 1985, с 34-43.

67. Мойсеенко В. И., Бутенко В. М. Компьютерная система упраления движением поездов. Залізничний транспорт України № 5-6/2000. К:, с.80-82
68. Мойсеенко В. И., Поддубняк В. И., Чепцов М. Н. Моделирование технологических процессов в системах железнодорожной автоматики. Збірник наукових праць Київського інституту залізничного транспорту. Том 3. 1999р. с. 66-73
69. Мойсеенко В. И., Чепцов М. Н. Использование нейронных сетей для моделирования в системах диспетчерской централизации. Информационно-управляющие системы на железнодорожном транспорте. 3/1999, Харьков, с. 38-42
70. Мойсеенко В. И., Чепцов М. И. Моделирование поездной ситуации на станции. Информационно-управляющие системы на железнодорожном транспорте. 6/1998, Харьков, с. 6-8
71. Мороз В. И. Теоретические основы создания нового поколения технических систем и технических средств для железнодорожного транспорта // Информационно-управляющие системы на железнодорожном транспорте. 1/1997 - с. 81-84
72. Моторыгин Б. и др. Программно-целевое управление и хозрасчет в науке. М.: Экономика, 1991.
73. Моцкус И. Б. Многоэкстремальные задачи в проектировании. М.: Наука, 1967.
74. Нагорный Е. В. и др. Методика оптимизации технических параметров пограничных передаточных станций. Информационно-управляющие системы на железнодорожном транспорте. 5/1998, Харьков, с. 32-38
75. Надежность железнодорожных систем автоматики и телемеханики. Меньшиков Н. *Я.,* Королев А. И., Ягудин Р. Ш. - М.: Транспорт, 1976. - 215 с.

76. Переборов А. С. и др. Диспетчерская централизация: Учебник для вузов ж.-д. Трансп. / А. С. Перборов, О. К. Дрейман, Л. Ф. Кондратенко; Под ред. Вал. В. Сапожникова. -М.: Транспорт, 1989. - 303 с.
77. Половинкин А. И. Основы инженерного творчества. // М.: "Машиностроение", 1988—368с.
78. Прохоров А. Ф. Системное проектирование технических средств. // Автоматизация проектирования, 1998, №1.
79. Разгонов А. П. О влиянии электрической дуги на токосъемнике при гололедообразовании на рельсовые цепи. Информационно-управляющие системы на железнодорожном транспорте. 6/1998, Харьков, с.22-29
80. Растригин Л. А. Современные принципы управления сложными объектами. - М.: Сов. Радио, 1980. - 232 с.
81. Розенблат Ф. Принципы нейродинамики.//М.: МИР, 1965.
82. Розенблат Ф. Аналитические методы изучения нейронных сетей.// Зарубежная радиоэлектроника. - 1965 - N 5 - с. 40-50.
83. Розенблатт Ф. Принципы нейродинамики (перцептрон и теория механизмов мозга).//М.: "Мир", 1965.—480с.
84. Розробка автоматизований системи знімання інформації про проходження поїздів по міждержавних і міждорожних стикових пунктах. Звіт про НДР. 904-28.99-705ю99-Цтех від 01.04.99.
85. Сапожников В.В., Кравцов Ю.А., Сапожников Вл.В. Дискретные устройства железнодорожной автоматики, телемеханики и связи. М.: Транспорт, 1988.
86. Саркисян С., Акопов П., Мельникова Г. Научно-техническое прогнозирование и программно-целевое планирование в машиностроении. М.: Машиностроение, 1987.
87. Система концентрации диспетчерского управления по направлениям и районам управления. ИВЦ Донецкой железной дороги. Техническая документация. Рег. ном. 2033910482-97. Донецк 1997.

88. Соболев Ю. В. Построение перспективных путевых преобразователей // Информационно-управляющие системы на железнодорожном транспорте. 1-2/1996 с. 45-47
89. Соболев Ю. В., Бабаев М. М., Давиденко М. Г. Распознавание двоичной кодовой комбинации в условиях априорной неопределенности // Информационно-управляющие системы на железнодорожном транспорте. 1/1997 с. 71-73
90. Соколов Е.Н., Вайтнявичус Г.Г. Нейроинтеллект: от нейрона к нейрокомпьютеру,- М.: Наука, 1989. С. 283.
91. Стандартизація та сертифікація на залізничному транспорті: Законодавчі та галузеві стандарта. Т-1. Стандартизація/Укл.: І.П.Данькевич та ін. - Харьків: ХарДАЗТ, 1999. - 353с.
92. Суворов С.В., Матихина Н.Ю. Программное моделирование нейроподобных структур.//Распределенная обработка информации.- Улан-Уде, 1989, - с. 28.
93. Теория информации и ее приложения (Сборник переводов). Под ред. Харкевича А. А. // М.: Государственное изд. Физ.-мат. Литературы. 1959 - 328 с.
94. Тимофеев А.В.. Роботы и искусственный интеллект. // М. "Наука" 1978- 192 с.
95. Трикоз Д.В. Нейронные сети: как это делается?// Компьютеры + программы - 1993 - N 4(5) - с. 14-20.
96. Тужилкин В. М. Информация о поездах и грузах. // М.: Транспорт 1972. 144 с.
97. Тэнк Д.У., Хопфилд Д.Д. Коллективные вычисления в нейроноподобных электронных схемах.//В мире науки. 1988. N2. С. 44-53.
98. У. Росс Эшби. Конструкция мозга. Происхождение адаптивного поведения. // М.: Издательство иностранной литературы, 1962—392с.

99. Управление эксплуатационной работой и качеством перевозок на железнодорожном транспорте: Учебник для вузов/ П. С. Грунтов, Ю. В. Дьяков, А. М. Макарочкин и др.; Под ред. П. С. Грунтова. - М.: Транспорт, 1994 г. - 543 с.
100. Успенский В. А. Теорема Геделя о неполноте. М.: Наука, 1982.
101. Ф.Уоссермен, Нейрокомпьютерная техника, М., Мир, 1992. 231 с.
102. Филиппенко И. Г., Глушакова А. Ю. Нейропарадигмы и перспективы их применения при реализации “НОЖпарадигмы” Информационно-управляющие системы на железнодорожном транспорте. 2/1998. Харьков, с. 51- 58
103. Флейшман Б. С. Основы системологии. М.: Радио и связь, 1982.
104. Фомиченкова Л. В. Динамическое моделирование в стратегическом анализе и планировании. // Менеджмент в России и за рубежом, 1998, №3.
105. Цуприков С. Нейронные вычисления берутся на вооружение финансистами.// Computerworld - Moscow - 1985 - N 7 - с. 57-58.
106. Чепцов М. Н. К вопросу повышения достоверности информации в системах диспетчерского управления. Інформаційно-керуючи системи на залізничному транспорті, №6, 2000, с. 56-57.
107. Чепцов М. Н. Метод оценки технического состояния объектов автоматики. Залізничний транспорт України, №5-6, 2000, с. 86.
108. Шилов И. С. Разработка вариантов декомпозиции полигона Белорусской железной дороги на районы управления. - В. кн.: Проблемы централизации диспетчерского управления на железных дорогах: Межвуз. сб. науч. Ст. - Гомель: БелИИЖТ, 1985, с 60-65.
109. Шмулевич М. И., Юшкевич Е. П. Информационное взаимодействие ж.д. транспорта и предприятий. М. Транспорт. 1984.
110. Эдельман В. И. Надежность технических систем: экономическая оценка. М.: Экономика, 1989.

111. Ягудин Р. Ш. Надежность устройств железнодорожной автоматики и телемеханики. -М.: Транспорт. 1989. - 159 с.
112. Aarts E.H.L., Korst J.H.M. Boltzmann machines for travelling salesman problem//European J. Oper. Res. 1989. V. 39. P. 79-95.
113. Aarts E.H.L., Korst J.H.M. Boltzmann machines and their applications//Lect. Notes Comput. Sci. 1987. V. 258. P. 34-50.
114. 14. Abu-Mostafa Y.S., Jaques J.N.St. Information capacity of the Hopfield model//IEEE Trans. Inform. Theory. 1985. V. 31. P. 461.
115. Ackley D.H., Hinton G.E., Sejnowski T.J. A learning algorithm for Bolzmann machines//Cognit. Sci. 1985. V. 9. N 1. P. 147-169.
116. Alain Petrowski, Gerard Dreyfus, Claude Girault, Performance Analysis of a Pipelined Backpropagation Parallel Algorithm //IEEE Transactions on Neural Networks, Vol.4, N6, 1993, pp.970-981.
117. AmariS. Field theory of self-organizing neural networks//IEEE Trans. Syst., Man, Cybern. 1983. V. 13. P. 741.
118. Artificial Intelligence. // Amsterdam: Time - Life - Books, 1986.
119. Artificial Neural Networks: Concepts and Theory, IEEE Computer Society Press, 1992.
120. Athale R., Stirk C.W. Compact architectures for adaptive neuraal nets//Tbid. 1989. V. 28. N4.
121. .Back Propagation Neural Net Engine vl.33u for C programmers by Patrick Ko Shu-pui. No. 11,14 ST., Hong Lok Yuen Tai Po, Hong Kong.
122. Bardcev S.I., Okhonin V.A. The algorithm of dual functioning (back-propagation): general approach, vesions and applications. Krasnojarsk: Inst, of biophysics SB AS USSA - 1989.
123. Bernard Widrow, Michael A. Lehr, 30 Years of Adaptive NeuralNetworks: Perceptron, Madaline, and Backpropagation //Artificial Neural Networks: Concepts and Theory, IEEE Computer Society Press, 1992, pp.327-354.

124. Carpenter G.A., Grossberg S. A massively parallel architecture for a self-organizing neural pattern recognition machine.//Comput. Vision Graphics Image Process. 1986. V. 37. p. 54-115.
125. Clemons E. C. Using scenario analysis to manage the strategic risks of reengineering. // Sloan Management Review, 1995, v.36, №4.
126. Cohen M.A., Grossberg S. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks//IEEE Trans. Syst, Man, Cybern. 1983. V. 13. N 5. P. 815-826.
127. Coutwell J., Fai F. Firms as the source of innovation and growth: the evolution of technological competence. // J. of Evolutionary Economics, 1999, №9.
128. Dayhoff J. Neural network architectures.//New-York:Van Nostrand reinhold, 1991.
129. Fogelman Soulie F. Neural networks, state of the art, neural computing.// London: IBC Technical Services, 1991.
130. Fox G.C., Koller J.G. Code generation by a generalized neural networks: general principles and elementary examples.//!. Parallel Distributed Comput. 1989. V. 6. N2. P. 388-410.
131. Gael de La Croix Vaubois, Catherine Moulinoux, Benoit Derot, The N Programming Language //Neurocomputing, NATO ASI series, vol.F68, pp.89-92.
132. H.A.Malki, A.Moghaddamjoo, Using the Karhunen-Loe've Transformation in the Back-Propagation Training Algorithm //IEEE Transactions on Neural Networks, Vol.2, N1, 1991, pp. 162-165.
133. Harris Drucker, Yann Le Cun, Improving Generalization Performance Using B ackpropagation //IEEE Transactions on Neural Networks, Vol.3, N5, 1992, pp.991-997.
134. Hecht-Nielsen R. Neurocomputing: picking the human brain.// IEEE SPECTRUM 1988 -V. 25. N3 - p. 36-41.

135. Hopfield J.J. Neural networks and physical systems with emergent collective computational abilities.//Proc. Natl. Acad. Sci. 1984. V. 9. p. 147-169.
136. Hopfield J.J., Feinstein D.I., Palmer F.G. Unlearning has a stabilizing effect in collective memoriesZZNature. 1983. V. 304. P. 141-152.
137. Hopfield J.J., Tank D.W. Neural computation of decisionin optimization problemsZZBiol. Cybernet. 1985. V. 52. P.141-152.
138. Jeffery W., Rosner R. Neural network processing as a tool for friction optimization.Z/Neuronet Comput. Conf., Snowbird, Utah, Apr. 13-16, 1986. New York, N.Y., 1986 - p. 241-246.
139. Kuzewski Robert M., Myers Michael H., Grawford William J. Exploration of fourword error propagation as self organization structure.ZZIEEE 1st. Int. Conf. Neural Networks, San Diego, Calif., June 21-24, 1987. V. 2. - San Diego, Calif., 1987.-p.89-95.
140. Lippmonn Richard P. Gold Ben Neuronet classifiers useful for speech recognition.ZZ IEEE 1st. Conf. Neural Networks, San Diego, (Calif), 1987 - p. 417-425.
141. Paul J. Werbos, B ackpropagation Through Time: What It Does and How to Do It ZZArtificial Neural Networks: Concepts and Theory, IEEE Computer Society Press, 1992, pp.309-319.
142. P. Foran. Progressive Railroading, 1997, N 6, p. 32 - 33, 36, 38, 40.
143. Rosenblatt F. Principles of neurodynamics. Spartan., Washington, D.C., 1962.
144. Rosenblatt F. The peseptron: a probabilistic model for information storage and organization in the brainZZPsychol. Rev. 1958. V. 65. P. 386.
145. Rumelhart B.E., Minton G.E., Williams R.J. Learning representations by back propagating error.ZZ Wature, 1986. V. 323. p. 1016-1028.
146. Sankar K. Pal, Sushmita Mitra, Multilayer Perceptron, Fuzzy Sets, and Classification ZZIEEE Transactions on Neural Networks, Vol.3, N5,1992, pp.683-696.
147. Steinbuch KZ Die LernmatrixZZ Kybernetik.-1961.-p.36

148. Takefuji D.Y. A new model of neural networks for error correction./ZProc. 9th Annu Conf. IEEE Eng. Med. and Biol. Soc., Boston, Mass., Nov. 13-16, 1987. V. 3, New York, N.Y., 1987 -p. 1709-1710.
149. Thakoor A. P., Moopenn A., Lambe J., Khanna S. K. Electronic hardware implementations of neural networks//Appl.Opt. 1987.-26,N23.-p5085-5092
150. Treliven P. Neurocomputers.// London: University college, 1989.
151. Widroow B., Hoff M. B. Adaptive Switching Circuits // IRE WESCOW Conv. Record. - 1960.-p.96-104

Приложение А

Гистограмы временных характеристик функционирования объектов телесигнализации типа путевой участок

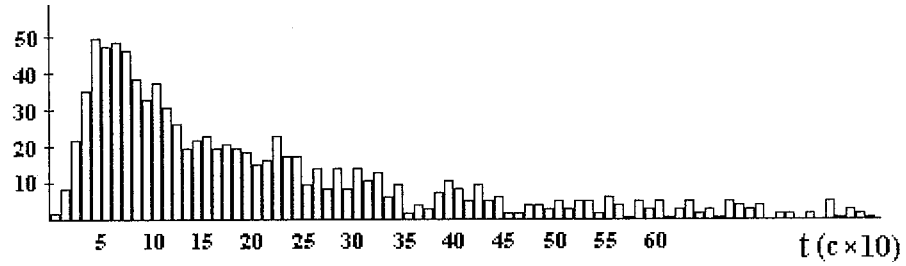


Рис. А.1 Станция Мандрыкино, 1-й участок удаления

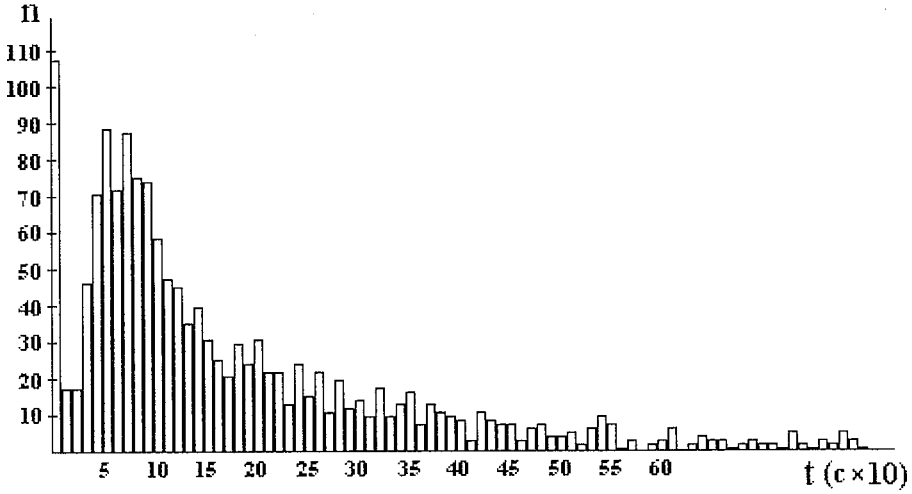


Рис. А.2 Станция Донецк 2-й участок удаления

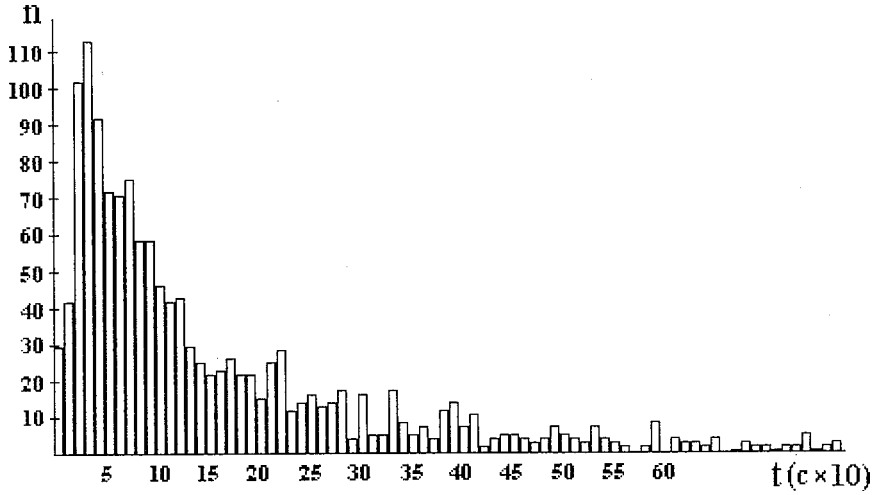


Рис. А.3 Станция Рутченково, 2-й участок удаления

Приложение А (продолжение)

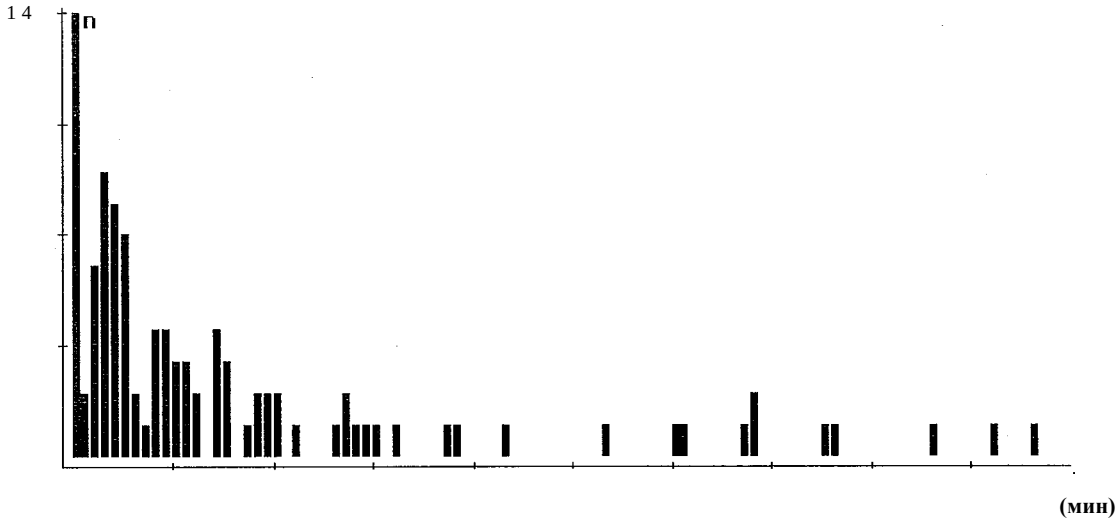


Рис А.4 Приемо-отправочный путь 9П станции Жуляны

n

(мин)

Рис. А.5 Блок-участок Б6 станции Жуляны

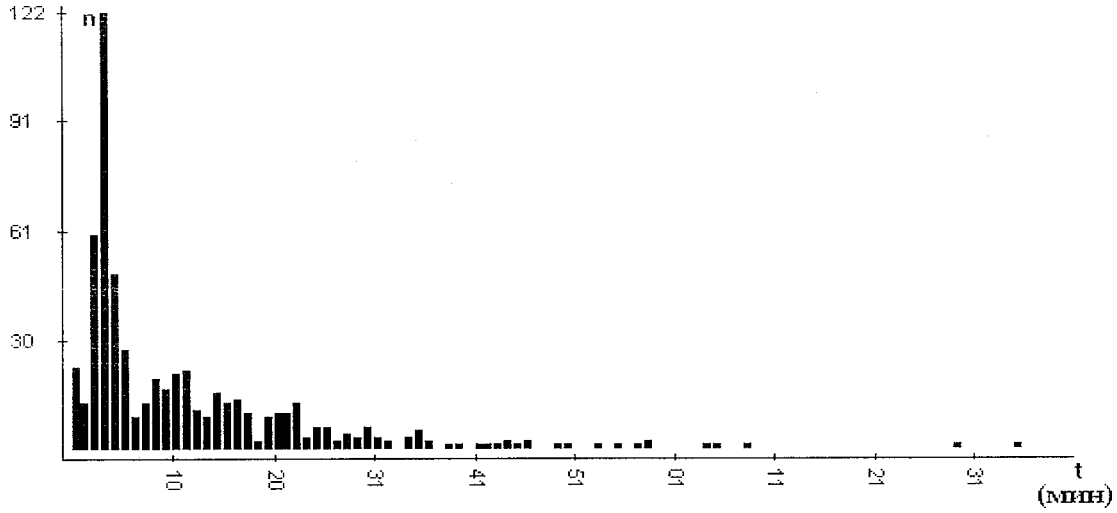


Рис. А.6 Приемо-отправочный путь 5П станции Жуляны

Приложение Б

Программное обеспечение, предназначенное для обработки статистических данных функционирования объектов телесигнализации систем ДЦ

Диспетчерская централизация «Луч», программа «МУ», создана на Microsoft Visual C++ 6.0. В программе описаны следующие классы: CAboutDlg, CC, CCodCanel, CGroup, CMainFrame, CMyApp, CMyDoc, CMyView, CRkkDlg, CTact. В связи с тем, что среда программирования предоставляет пользователю уже созданные, стандартные классы, то они не приводятся в данном приложении.

Описание класса CC - файл C.h

```
//include "CodCanel.h" // Added by ClassView
class CC : public CDocument
{
public:
    CC();    // protected constructor used by dynamic creation
    DECLAREDYNCREATE(CC)
    virtual void Serialize(CArchive& ar); // overridden for document i/o
protected:
    virtual BOOL OnNewDocument();
public:
    void Base(CString C,CDaoRecordset* V);
    BOOL Change(CString K,UINT Cikl);
    BOOL PutData(CString CodCanel,COleVariant& Data,UINT Cikl);
    BOOL InsertCanel(CString Norn,UINT t);
    virtual ~CC();
Z/ifdef _DEBUG
    virtual void AssertValid() const;
```

Приложение Б(продолжение)

```
    virtual void Dump(CDumpContext& de) const;
#endif
protected:
    DECL AREMES S AGE_MAP()
private:
    CCodCanel* ArrayCanel[12];
i ■
```

Реализация класса CC - файл C.cpp

```
// C.cpp : implementation file
#include "stdafx.h"
#include "my.h"
^include "C.h"
#ifdef JDEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE_;
#endif
IMPLEMENTED YNCREATE(CC, CDocument)
CC::CC()
i
    for(UINT k=0;k<12;k++)
    i
        ArrayCanel[k]=NULL;
    1
    /
}
```

Приложение Б(продолжение)

```
BOOL CC::OnNewDocument()
{
        if (! CDocument: :OnNewDocument())
                return FALSE;
        return TRUE;
i
i


CC::~CC()
i
i
        for(UINT k=0;k<12;k++)

                if( ArrayCanel [k])
                 (
                        delete ArrayCanel [k];



I


BEGIN_MESSAGE_MAP(CC, CDocument)
        //{{AFX_MSG_MAP(CC)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

#ifdef _DEBUG
void CC::AssertValid() const

        CDocument:: AssertValidQ;}
```

Приложение Б(продолжение)

```
void CC::Dump(CDumpContext& de) const

        CDocument: :Dump(dc);
}
#endif//_DEBUG

void                    CC::Serialize(CArchive&                    ar)
r
        if (ar.IsStoringO)
        i
                // TODO: add storing code here
        J
        else

                // TODO: add loading code here


i

BOOL CC::InsertCanel(CString Nom,UINT t)
it
        for(UINT k=0;k<12;k++)
        ii
                if( Array C anel [k]==NULL)
                                        f
                        ArrayCanel[k]= new CCodCanel(Nom);
                        return TRUE;
                J
```

Приложение Б(продолжение)

```
        return FALSE;
i
J


BOOL CC::PutData(CString CodCanel, C01eVariant& Data,UINT Ciki)
i
        for(UINT k=0;k<12;k++)

                if(ArrayCanel[k]==NULL)
                f
                        return FALSE;
                i
                J
                if(ArrayCanel[k]->GetName()==CodCanel)
                r
                t
                        char mould[58] ;
                        memcpy(mould,Data.parray->pvData,58);
                        for(UINT nGr=0;nGr<23;nGr++)
                        {
                                for(UINT nTakt=0;nTakt<20;nTakt++)
                                {
                                        UINT nbchan     =       nGr*20+nTakt;
                                        // номер бита в слепке
                                        UINT nb         =       nbchan»3;
                                        // номер байта в слепке
                                        UINT nbit = nbchan-(nb«3);
                                        // номер бита в байте
                                        unsigned char maska =           1 «nbit;
                Array Canel [k]->PutData(nGr,nTakt,(mould[nb]&(maska))! =0,Cikl);
```

Приложение Б(продолжение)

```
}

/*
        // номер бита в канале
        int nbchan=(gr->GetNumberGroup()-1 )*20+takt->GetNumberTakt()-1;
        // номер байта в слепке
        int nb=nbchan»3;
        // номер бита в байте
        int nbit=nbchan-(nb«3);
        unsigned char maska=l«nbit;
        takt->SetStatusTakt(SCB_CIKLOFPROTOKOL,    (cMould[nb]&(~maska))
!=0);
/*
        00010000    -       maska
        11101111    -       -maska
        10101010    -       cMouldfnb]
        10101010    -       результат
*/
        return TRUE;
i


BOOL CC::Change(CString K, UINT Cikl)
f
        CString NC = K;
        NC.Delete(3,6);
        for(UINT nc=0;nc<12;nc++)
```

Приложение Б(продолжение)

```
Ï
        if(ArrayCanel[nc]->GetName()==NC)
        i
            ArrayCanel [nc] ->Change(K,Cikl);
        i
    return TRUE;
i

void CC::Base(CString C, CDaoRecordset *V)

    CString NC = C;
    NC.Delete(3,6);
    for(UINT nC=0;nC<12;nC++)

        if(ArrayCanel[nC]->GetName()==NC)
        r
            ArrayCanel[nC]->Base(C,V);
```

Описание класса CCodCanel - файл CodCanel.h

```
#include "Tact.h" // Added by ClassView
#include "Group.h"
class CCodCanel
{
protected:
```

Приложение Б(продолжение)

```
CCodCanel(); // protected constructor used by dynamic creation
public:
        void Base(CString C, CDaoRecordset *V);
        BOOL Change(CString K, UINT Cikl);
        BOOL PutData(UINT NomGr,UINT NomTakt,BOOL TekSost,UINT Cikl);
        CString GetName();
        CCodCanel(CString N);
        virtual -CCodCanelQ;
private:
        C Gro up* Array Group [23];
        UINT iValue;
        CString                                                    Name;
};
```

Реализация класса CCodCanel - файл CodCanel.cpp

```
//include "stdafx.h"
//include "my.h"
//include "group.h"
//include "codcanel.h"
Z/ifdef _DEBUG
//define new DEBUG_NEW
//undefTHIS_FILE
static char THIS_FILE[] = _FILE_;
Z/endif

CCodCanel:                                                 :CCodCanel()
r
i
i
```

Приложение Б(продолжение)

```
CCodCanel: :~CCodCanel()
i
        for(UINT t=0;t<20;t++)

                i f( Array Group [t])
                {
                        delete ArrayGroup[t];




CCodCanel: :CCodCanel(CString N)
{
        Name=N;
        iValue=atoi(Name);
        for(UINT t=0;t<23 ;t++)

                ArrayGroupft] = new CGroup;



CString CCodCanel::GetName()

        return Name;
```

Приложение Б(продолжение)

```
BOOL CCodCanel ::PutData(UINT NomGr, UINT NomTakt, BOOL TekSost,UINT Cikl)
i
        ArrayGroup [NomGr] ->PutData(NomT akt,TekS ost,C ikl);
        return TRUE;


BOOL CCodCanel ::Change(CString K, UINT Cikl)
{
        CString NC = K;
        NC.Delete(0,4);
        NC.Delete(2,3);
        UINT nGr = atoi(NC);
        ArrayGroup [nGr-1 ]->Change(K,Cikl);
        return TRUE; }
void        CCodCanel::Base(CString        C,        CDaoRecordset        *V)
i
        CString NC = C;
        NC.Delete(0,4);
        NC.Delete(2,3);
        UINT nGr - atoi(NC);
        Array Group [nGr-1 ]->Base(C,V);
```

Описание класса CMyDoc - файл MyDoc.h

```
#include <afxtempl.h> //
```

Приложение Б(продолжение)

```
typedef CList <CString,CString&> CNameList;

class CMyDoc : public CDocument
r
i
        UINT i_CountmasStat;
        UINT i_MaxmasStat;
protected: // create from serialization only
        CMyDoc();
        DECLARE JTYNCREATE(CMyDoc)
public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        BOOL m_bDiagr;
        void MyBaseQ;
        CC Signal;
        virtual ~CMyDoc();
        UINT* GetMasQ {return m_masStat;};
        UINT GetCountMas() {return i_CountmasStat;};
        UINT GetMaxMas() {return i_MaxmasStat;};
#ifdef _DEBUG
        virtual void AssertValidQ const;
        virtual void Dump(CDumpContext& de) const;
#endif
protected:
        //{{AFX_MSG(CMyDoc)
        afx_msg void OnRunQ;
        afx__msg void OnGrafikQ;
        afximsg void OnDiagrQ;
```

Приложение Б(продолжение)

```
        //}}AFX_MSG
        DECLAREJMESSAGE_MAP()
private:
        BOOL CopyChangeTakts();
        BOOL CopyMouldChannels();
        void DisplayDaoException(CDaoException* e, LPCTSTR lpcszInfo=NULL);
        BOOL CopyTimeCSQ;
        void CloseNextDBQ;
        BOOL OpenNextDB(CString& FileName);
        BOOL OpenFullDB();
        BOOL LoadFileListQ;
        CNameList       mJFileLst;
        CDaoDatabase    mProtocolDB;
        CDaoRecordset*  m_pRstTimeCikl;
        CDaoRecordset*  m_pRstMouldCannels ;
        CDaoRecordset*  m_pRstChangeTakts ;
        CDaoRecordset*  m_pRstMyBase ;
        CDaoRecordset*  mjpRstMySignal;
        CDaoDatabase    m_WorkDB;
        UINT*           m_masStat;
i
```

Реализация класса CMyDoc - файл MyDoc.cpp

```
#include "stdafx.h"
//include "my.h"
//include "myDoc.h"
//include "RkkDialog.h"
```

Приложение Б(продолжение)

```
#ifdef DEBUG
#defme new DEBUG_NEW
#undef THIS_FILE
static char THTS_FTLE[] = _FILE_;
#endif
const UINT COUNTJEL                =        100;

IMPLEMENT_DYNCREATE(CMyDoc, CDocument)
BEGIN_MESSAGE_MAP(CMyDoc, CDocument)
        ON_COMMAND(ID_RUN, OnRun)
        ON_COMMAND(ID_GRAFIK, OnGrafik)
        ON_COMMAND(ID_DIAGR, OnDiagr)
END_MESS AGE_MAP()

CMyDoc::CMyDoc()
)
I
        m_pRstTimeCikl              NULL;
        mpRstMouldCannels           NULL;
        m_pRstChangeTakts           NULL;
        m_masStat                   NULL;
        m_bDiagr                    FALSE;



CMyDoc::~CMyDoc()
i
        if (m_masStat)

                delete[] m_masStat;
```

Приложение Б(продолжение)

```
        m_masStat = NULL ;
i
i f( m_pRstT i meC i kl)
i
        if(m_pRstTimeCikl->IsOpen())
        {
                m_pRstTimeCikl->Close();

        delete m_pRstTimeCikl ;
        m_pRstTimeCikl = NULL ;

if(m_pRstMouldCannels)
f
        if(m_pRstMouldCannels->IsOpen())
        {
                mjpRstMouldCannels->Close();

        delete m_pRstMouldCannels ;
        m_pRstMouldCannels = NULL;
}
if(m_pRstChangeTakts)
s
i
        if(m_pRstChangeTakts->lsOpen())
        {
                mjpRstChangeTakts->Close();

        delete m_pRstChangeTakts ;
        m_pRstChangeTakts = NULL;
```

Приложение Б(продолжение)

```
BOOL CMyDoc::OnNewDocument()
{
    if (! CDocument:: OnNewDocument())
        return FALSE;
    if(LoadFileList()&&OpenFullDB())
    {
    }
    return TRUE ;
}

void CMyDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoringO)

        // TODO: add storing code here

    else
    {
        // TODO: add loading code here

}

#ifdef DEBUG
void CMyDoc::AssertValid() const
```

Приложение Б(продолжение)

```
        CDocument: :AssertValid();
i


void CMyDoc::Dump(CDumpContext& de) const
{
        CDocument: :Dump(dc);
i
j
#endif//JDEBUG


BOOL CMyDoc::LoadFileList()
i
i
        CFileFind fpr;
        BOOL bFind=fpr.FindFile("\\bdscb\\PROTOCOL\\*.mdb");
        while(bFind)
        i
                bFind=fpr.FindNextFile();
                CString name=fpr.GetFileTitle();
                if(name=="LostGroup")
                        continue;
                POSITION pos;
        for(pos=m_FileLst.GetHeadPosition();pos!=NULL;m_FileLst.GetNext(pos))
                {
                         CString curname=m_FileLst.GetAt(pos);
                        if(name < curname)
                                break;
                }
                if(pos)
                        m_FileLst.InsertBefore( pos, name );
```

Приложение Б(продолжение)

```
        else

                m_FileLst.AddTail(name);
    i
    J
    return TRUE ;
/

BOOL CMyDoc::OpenFullDB()
i
t
try
f
        CString     FileName("fullprot.mdb");
        CString     FileExt(".mdb");
        CString     FileFiltrf'Basa данных Access
(fullprot.mdb)|fullprot.mdb|") ;
        CFileFind fpr;
        if(! fpr .F indF ile(F ileName))
        r
            CFileDialogdlg(TRUE,FileExt,FileName,OFN_HIDEREADONLY |
OFNO VERWRITEPROMPT,FileFiltr);
            if (dlg.DoModal()!=IDOK)
             f
             i
                    return FALSE ;

            FileName = dlg.GetPathName();
    }
    m_ProtocolDB.Open(FileName);
    i
    \
```

Приложение Б(продолжение)

```
        {
                DisplayDaoException(e);
                return FALSE;
        }
        return TRUE ;
f

void            CMyDoc::DisplayDaoException(CDaoException       *e,LPCTSTR
Ipcszlnfo/*=NULL*/)
{
        CString strName;
        if(e->m_pError!nfo)
                strName.Format("Ошибка ДАО : %d",e->m_pErrorInfo->m_lErrorCode);
        else
                strName=_T("Ошибка ДАО");
        if(lpcszlnfo)
        {
                strName=_T(" ")+strName;
                strName=_T(lpcszInfo)+strName;
        i
        (CMyApp*)AfxGetApp()->m_pMainWnd->MessageBox(strName);
        e->Delete();
j

void CMyDoc::OnRun()
{
        //Убрать, если необходимо слить базы
        return;
```

Приложение Б(продолжение)

```
BOOL ok = FALSE ;
POSITION pos;
pos=m_FileLst.GetHeadPosition();
while (pos)
{
    CString curname=m_FileLst.GetAt(pos);
    OpenNextDB(curname);
    if (CopyTimeCS()&&CopyMouldChannels()&&CopyChangeTakts())
    {
        ok = TRUE;
    }
    CloseNextDBQ ;
    m_F i leL st. GetN ext(po s);
}
J


BOOL CMyDoc::OpenNextDB(CString &FileName)

{
try
{
    CString name = "Wbdscb\\protocol\\"+FileName+".mdb";
    m_ W orkDB. Open(name);

catch (CDaoException* e)
{
    DisplayDaoException(e);
    return FALSE;
```

Приложение Б(продолжение)

```
        return TRUE ;
i
}

void CMyDoc::CloseNextDB()

        if (m_WorkDB.IsOpen())
        {
                m_WorkDB.Close();
        }



BOOL CMyDoc::CopyTimeCS()
{
        m_pRstTimeCikl = new CDaoRecordset(&m_ProtocolDB);
        CString strSQL-'SELECT * FROM TimeCS";
        trv
        {
                m_pRstTimeCikl->Open(dbOpenDynaset,strSQL,dbAppendOnly);
        }
        catch (CDaoException* e)
        {
                DisplayDaoException(e);
                return FALSE;
        }
        m_pRstMouldCannels = new CDaoRecordset(&m_ProtocolDB);
        strSQL-"SELECT * FROM Mouldchannels";
        try
```

Приложение Б(продолжение)

```
        m_pRstMouldCannels->Open(dbOpenDynaset,strSQL,dbAppendOnly);

catch (CDaoException* e)

        DisplayDaoException(e);
        return FALSE;
}
m_pRstChangeTakts = new CDaoRecordset(&m_ProtocolDB) ;
strSQL="SELECT * FROM ChangeTakts";
try

        m_pRstChangeTakts->Open(dbOpenDynaset,strSQL,dbAppendOnly);

catch (CDaoException* e)

        DisplayDaoException(e);
        return FALSE;

CDaoRecordset WorkTimeCikl(&m_WorkDB);
CString My StrSQL-'SELECT * FROM TimeCS ORDER BY TimeCikl";
try

        WorkTimeCikl.Open(dbOpenSnapshot,MyStrSQL,dbRcadOnly);
        WorkTimeCikl.MoveFirstQ ;
        UINT count = WorkTimeCikl.GetRecordCount();
        for(UINT i = 0;i<count;i++)

                COle Variant        varl ,var2,var3 ;
```

Приложение Б(продолжение)

```
                WorkTimeCikl.GetFieldValue("TimeCikl",varl) ;
                WorkTimeCikl.GetFieldValue("Restart",var2);
                WorkTimeCikl.GetFieldValue("Changed",var3);

                m_pRstTimeCikI->AddNew();
                m_pRstTimeCikl->SetFieldValue("TimeCikl",varl);
                m_pRstTimeCikl->SetFieldValue("Restart",var2);
                m_pRstTimeCikl->SetFieldValue("Changed",var3);
                m__pRstTimeCikl->Update();

                WorkTimeCikl.MoveNext();
        i
        j
        i
        catch (CDaoException* e)
        f
        i
                DisplayDaoException(e);
                return FALSE;
        }
        return TRUE ;
i
j

BOOL CMyDoc::CopyMouldChannels()
i
        CDaoRecordset rstMould(&m_WorkDB);
        CString strSQL="SELECT * FROM MouldChannels ORDER BY TimeCikl";
        try
        i
                rstMould.Open(dbOpenSnapshot, strSQL, dbReadOnly);
```

Приложение Б(продолжение)

```
            rstMould.MoveFirstQ ;
            UINT count = rstMould.GetRecordCount();
            for(UINT i = 0;i<count;i++)
            {
                COleVariant varl,var2,var3 ;
                rstMould.GetFieldValue("TimeCikl",varl);
                rstMould.GetFieldValue("CodeChannel",var2);
                rstMould.GetFieldValue("MouldChannel",var3);
                m_pRstMouldCannels->AddNew();
                m_pRstMouldCannels->SetFie!dValue("TimeCikl",varl);
                m_pRstMouldCannels->SetFieldValue("CodeChannel",var2);
                mjpRstMouldCannels->SetFieldValue("MouldChannel",var3);
                m_pRstMouldCannels->Update();
                rstMould.MoveNext();
            i
            j
        i
        catch (CDaoException* e)
        f
        i
            DisplayDaoException(e);
            return FALSE;
        i
        i
        return TRUE ;
)

BOOL CMyDoc::CopyChangeTakts()

        CDaoRecordset rstChanged(&m_WorkDB);
        CString strSQL="SELECT * FROM ChangeTakts ORDER BY TimeCikl";
```

Приложение Б(продолжение)

```
        try
        s
        I
                rstChanged.Open(dbOpenSnapshot, strSQL, dbReadOnly);
                rstChanged.MoveFirst();
                UINT count = rstChanged.GetRecordCount();
                for(UINT i = 0;i<count;i++)
                f
                i
                        COleVariant varl,var2;
                        rstChanged.GetFieldValue("TimeCikl",varl);
                        rstChanged.GetFieldValue("CodeTakt",var2);
                        m_pRstChangeT akts-> AddNe w();
                        m_pRstChangeTakts->SetFieldValue("TimeCikl",varl);
                        m_pRstChangeTakts->SetFieldValue("CodeTakt",var2);
                        m_pRstChangeTakts->Update();
                        rstChanged.MoveNext();
                }

        catch (CDaoException* e)
        {
                DisplayDaoException(e);
                return FALSE;
        i
        return TRUE ;
j

void CMyDoc::OnGrafik()

        //убрать, если необходимо пересчитать
```

Приложение Б(продолжение)

```
//return;
CString strSQL-'SELECT * FROM TimeCS ORDER BY TimeCikl";
try
г
i
        CDaoRecordset MyTimeCS(&m_ProtocolDB);
        MyTimeCS.Open(dbOpenSnapshot,strSQL,dbReadOnly);
        MyTimeCS.MoveFirst();
        UINT count = MyTimeCS.GetRecordCountQ ;
        CDaoRecordset MyTimeMould (&m_ProtocolDB);
        CDaoRecordset MyTimeChangeSignal(&m_ProtocolDB);
        for(UINT i = 0;i<count;i++)
         i
         i
                COleVariant varl,var2,var3;
                MyTimeCS.GetFieldValue("TimeCikl",varl) ;
                MyTimeCS.GetFieldValue("Restart",var2);
                MyTimeCS.GetFieldValue("Changed",var3);
                CString MySQL;
                MySQL.Format("%d ORDER BY TimeCikl",varl.lVal);
                if(var2.boolVal)                                    //рестарт
                г
                x
                        My SQL-' SELECT * FROM MouldChannels        WHERE
TimeCikl = "+MySQL;
                        if(MyTimeMould.IsOpen())
                        {
                          MyTimeMould.Close();

                        try
                        {
```

Приложение Б(продолжение)

```
MyTimeMould.Open(dbOpenSnapshot, MySQL, dbReadOnly);
                    MyTimeMould.MoveFirst();
                    UINTMyCount =
MyTimeMould.GetRecordCount();
                    for(UINT y=0;y<MyCount;y++)
                    {
                        COleVariant vl,v2,v3;
                        MyTimeMould.GetFieldValue("TimeCikl",vl);
MyTimeMould.GetFieldValue("CodeChannel",v2);
MyTimeMould.GetFieldValue("MouldChannel",v3);
                        // Получены данные по слепку или рестарту
                        CString code=V_BSTRT(&v2);
                        Signal.InsertCanel(code,i);
                        Signal.PutData(code,v3,i) ;
                        MyTimeMould.MoveNext();
                    }
                i
                catch (CDaoException* e)
                i
                    DisplayDaoException(e);
                    return ;
                i
            s
            else //Установлен флаг Changed
```

Приложение Б(продолжение)

```
                    MySQL-'SELECT * FROM ChangeTakts WHERE
Cikl - "+MySQL;
                    if(MyTimeChangeSignaI.IsOpen())

                        MyTimeChangeSignal.CloseQ ;
                    }
MyTimeChangeSignal.Open(dbOpenSnapshot, MySQL, dbReadOnly);
                    UINTMyCount =
MyTimeChangeSignal.GetRecordCount();
                    if(MyCount)

                        COleVariant v2;
                        MyTimeChangeSignal.MoveFirst();
                        for(UINT y=0;y<MyCount;y++)

MyTimeChangeSignal.GetFieldValue("CodeTakt",v2);
                                CString code = V_BSTRT(&v2);
                                Signal.Change(code,i) ;
                                MyTimeChangeSignal.MoveNextQ;



                catch(CDaoException* e)
                {
                    DisplayDaoException(e);
                    return;
```

Приложение Б(продолжение)

```
                MyT imeC S .MoveN ext();
            }
            MyTimeMould.Close();
            MyTimeChangeSignal.CloseQ ;
            MyTimeCS.Close() ;
        }
        catch (CDaoException* e)

            DisplayDaoException(e);
        /
        MyBasef);


void  CMyDoc::MyBase()
(
i
        m_pRstMyBase = new CDaoRecordset(&m_ProtocolDB) ;
        CString strSQL-'SELECT * FROM MyBase";
        try
        f
            m_pRstMyBase->Open(dbOpenDynaset,  strSQL,  dbAppendOnly);
        i
        j
        catch                          (CDaoException*                          e)
        f
        i
            DisplayDaoException(e);
            return;
        }

        m_pRstMySignal = new CDaoRecordset(&m_ProtocolDB) ;
```

Приложение Б(продолжение)

```
CString MyStrSQL-'SELECT * FROM MySignal ORDER BY MyTakt";
COleVariant MV1;
try
f
    m_pRstMySignal->Open(dbOpenSnapshot,MyStrSQL,dbReadOnly);
    m_pRstMySignal->MoveFirst();
    UINT count = m_pRstMySignal->GetRecordCount();
    for(UTNT i = 0;i<count;i++)
    t
        m_pRstMySignal->GetFieldValue("MyTakt",MV 1);
        CString code=V_BSTRT(&MVl);
        Signal.Base(code,m_pRstMyBase);
        m_pRstMySignal->MoveNext();
    i
    j
}
catch (CDaoException* e)
f
    DisplayDaoException(e);
    return;
j
m_pRstMySignal->Close();
m_pRstMyBase->Close();
}

f
    CRkkDialogdlg ;
    if(dlg.DoModal()==IDOK)
```

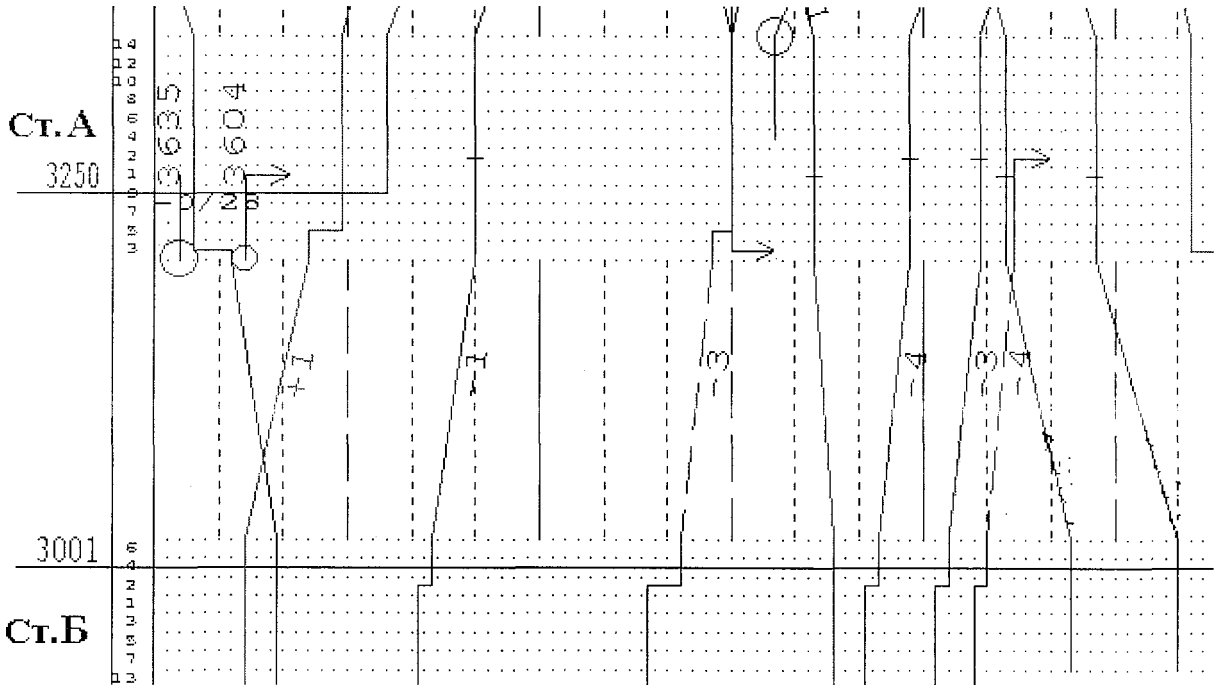
Приложение Б(продолжение)

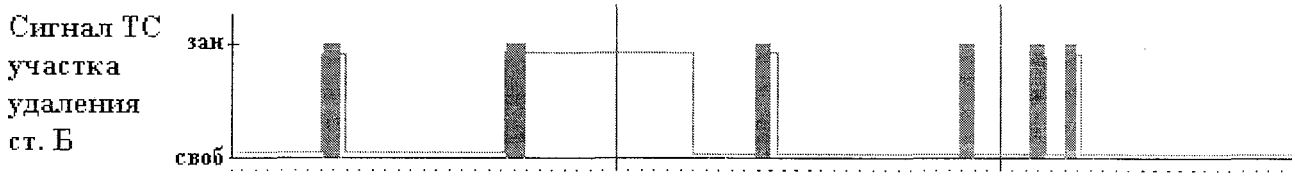
```
// получить имя такта
CString TaktName = dlg.m_rkk;
CDaoQueryDef qryReg(&m_ProtocolDB);
try
{
qryReg. Open(" GetStatistic");
qryReg.SetParamValue("nTakt",C01eVariant(TaktName,VT_BSTRT));
        CDaoRecordset rstStat(&m_ProtocolDB);
        rstStat.Open(&qryReg);
        UINT countRec = rstStat.GetRecordCount();
        if(!countRec)
        i
        i
            rstStat.CloseQ;
            qryReg.Close();
            return;
        }
        // получить данные
        if (m_masStat)
        t
        i
            delete[] m_masStat;
        mjmasStat = NULL ;
        }
        m_masStat = new UINT[COUNT_EL] ;
        i_C ountmas S tat=C OUNT_EL;
        for(UINT i=0;i<COUNT_EL;i++)
        f
            m_masStat[i]=O;
```

Приложение Б(продолжение)

```
        }
        for(i=0; i<countRec; i++)
        c
                COle Variant        var;
                rstStat.GetFieldValue("NInterval",var);
                UINT index =        var.1 Vai ;
                rstStat.GetFieldValue("Count_MyTakt",var);
                if ((index>=0)&&(index<100))
                 f
                     m_masStat[index-l] = var. 1 Vai;
                 i
                 J
                rstStat.MoveNext() ;
        i
        S
        m_bDi agr=TRUE;
        i_MaxmasStat=O;
        for(i=0;i<COUNT_EL;i++)
        r
        i
                if(m_masStat[i]>i_MaxmasStat) i_MaxmasStat=m_masStat[i];
        s
        UpdateAllViews(NULL);
)
catch (CDaoException* e)
i
        DisplayDaoException(e);
        return ;
```

Приложение В
Исполненный график движения, сигнал ТС и таблица расчетных значений к методике обучения нейронной сети

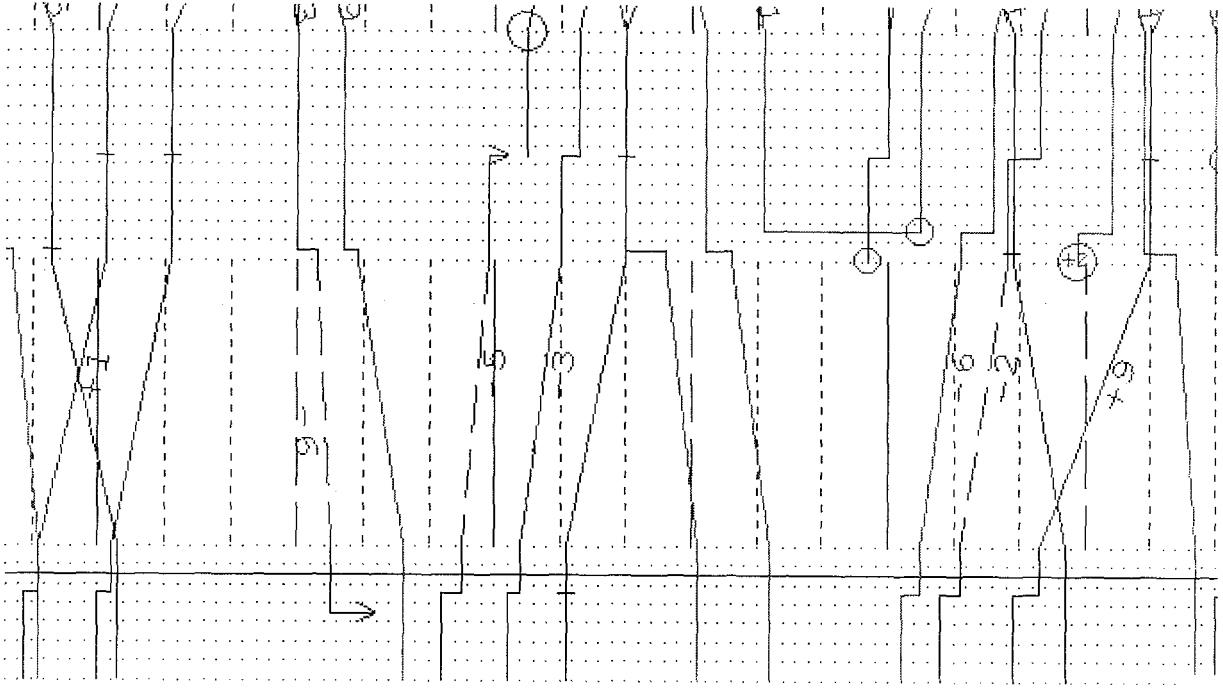


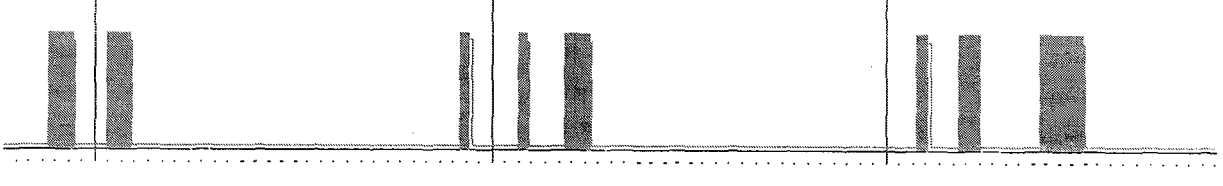


Номер поезда	6372 3635	3012	3606	5201 3002	2006	3506	3301	3003
Время хода по перегон}' (мин)	о Г-	г-	U3	СП из	из		00	сп
Ходовая скорость (м/с)	сп. су 00 гН	Су	U0	00 ио г- ио Є4	ио ио ио ио		‘TI* со	ио
Количество вагонов (шт)	О О СП	00	о тг	локо «J	из г- гч		сч	оз из
Длина поезда (м)	U0 U3 С П uo>	о р	о о ч>	из со сП ио	из г?	405	о	из ОО 00
Длит, занятия уч. удаления ст.Б по данным ГИД (с)	U0	U0	00 о	Гн	00	г- Сз		
Длит, занятия уч. удаления ст.Б по сигнал}’ТС (с)	О tq	1740	из	из	о гч	tri о		

150	148	ЗО	лок	8,3	10	5208
		272	16	20,8	4	187
210	206	720	48	9,3	9	3604
		795	53	8,3	10	2051
		405	27	41,7	o	3501
		136	8	11,9	■"I	6331
90	89	645	43	20,8	4	3608
90	88	ЗО	лок	13,9	6	5222
190	190	570	38	9,3	9	3506
		170	10	16,6	5	641
		136	8	13,9	6	6311
100	96	136	8	13,9	6	6320
145	145	525	35	11,9		3506
405	404	780	52	4,9	17	3012
		645	43	13,9	6	3061
		136	8	27,8	3	6341

Приложение В (продолжение)





Приложение Г

Программное обеспечение эмулятора нейронной сети путевого

Файл Neuron.h

```
class CNeuron : public CObject
{
private:
  double F(double Par);//Функция возбуждения
  UINT п_Ьауег;//Номер слоя, в котором находится этот нейрон
  UINT п_№игоп1пТауег;//Порядковый номер нейрона в слое
  double Randomize();
public:
  double Kof;
  BOOL LearnOK;
  void ImpuIsQ;
  double MiuPar;
  double NiuPar;
  void Update();
  void Learn();
  CNeuron** іп_Леигоп;//предьідущие нейроны
  void SetCalcErr(double Par);
  double CalcEri'O;
  double* Synaps;//МассНВ весовых коэф.по входу (синапсов)
  UINT п_8упарз;//Количество синапсов
  double* Delta;//МассНВ ошибок по каждому синапсу
  UINT n_Delta;//Kon-BO, равно количеству синапсов - 1


  double Error;
```

Приложение Г (продолжение)

```
  CNeuron* GetPointerQ;
  void Go();
  double Axon;//Значение выхода нейрона (аксон)
  BOOL Init(UINT n_L,UINT n_N,UINT n A);
  CNeuron();
  virtual ~CNeuron();
};
```

Файл Neuron.cpp

```
CNeuron: :CNeuron()
{
  Synaps = NULL;
  Delta = NULL;
  in_Neuron = NULL;
  Error =        0.;
  LearnOK = FALSE;
»
J
```

```
CNeuron: :~CNeuron()

  if(in_Neuron) delete[] in_Neuron;
  if(Synaps) delete[] Synaps;
  if(Delta) delete[] Delta;
>
```

Приложение Г (продолжение)

```
  n_Layer = п_Ь;//Номер слоя, в котором находится этот нейрон
n_NeuronInLayer = п_Ы;//Порядковый номер нейрона в слое
n_Synaps = п_А;//Количество синапсов
  Synaps = new double[n_Synaps]//Массив весовых коэф, по входу
nJDelta = n_Synaps;
Delta = new double[nJDelta];
//Массив указателей на нейроны пред слоя
in_Neuron = new CNeuron*[n_Synaps];

for(UINT i=O;i<n_Synaps;i++)
{
        in_Neuron[i] = NULL;
        Delta[i]=0.;
}

if(n_Layer!=O)

        for( i=0; i<n_Sy nap s; i++)
        {
                Synaps [i]=Randomize();
                Delta[i]=0.;
        i

}
return TRUE;



double CNeuron::Randomize()
```

Приложение Г (продолжение)

```
return (double)rand()/(double)RAND_MAX-0.5;
// return (double)RAND_MAX/(double)rand()-0.5;
i


double CNeuron: :F(double Par)
r
i
 return -0.5+1./(l+exp(-2.*Par));



void CNeuron: :Go()
r
i
 double S = 0.0;
 if(n_Synaps==l)
 {
        S+=Synaps[0];
 j
 else

        for(UINT i=0;i<n_Synaps;i++)
        i
                ASSERT(in_Neuron[i]);
                S+=in_Neuron[i]->Axon*Synaps[ij;
        i
        //и прибавляем значение последнего синапса
 i
 i
 Axon = F(S);
```

Приложение Г (продолжение)

```
CNeuron* CNeuron::GetPointer()

  return this;



double CNeuron::CalcErr()
f
  return 2. * (Axon+0.5) * (1.5 - Axon);
i
)

void CNeuron: :SetCalcErr(double Par)

  Error^ Axon-Par)* Cal cErr();



void CNeuron: :Learn()
r
i
  for(UINT i=O;i<n_Synaps;i++)
  {
        Deltafi] =N iuPar* (Deltafi]+(1. -MiuPar) *Error* Axon);

1
J .

void CNeuron::Update()
i
I
  for(UINT i=O;i<n_Synaps;i++)
```

Приложение Г (продолжение)

```
        Synaps[i]-=Delta[i];



void CNeuron::Impuls()

  if(Axon==-0.5)
  i
  i
        Axon=0.5;
  }
   -
  t
        Axon=-0.5;

// Axon=Randomize();
i

Файл NetBP.h
#include "afxdao.h"
#include <afxtempl.h>
^include "Neuron.h"
include "MY_NETView.h"

typedef CList <CString,CString&> CNameList;

class NetBP : public CObject
i
i
private:
```

Приложение Г (продолжение)

```
 UINT* NeuronINLayers;        // Количество нейронов в каждом слое
 UINT CountLayers;            // Количество слоев в сети
 CNeuron** N;                                  // Нейроны

CNameList m_FileLst;
CDaoDatabase m_MyNetDB;

BOOL      LoadFileList();
BOOL      OpenFullDB();
void      MessageBox(CString Str);
void      DisplayDaoException(CDaoException *e, LPCTSTR Ipcszlnfo);

public:
UINT CountCiId;
double NiuPar;
HINT Countlter;
void ImpulsQ;
double Kof;
void MsgWork(double* in_Array);
void MsgLearn(UINT Count,double** in_Array,double** out_Array);
void MsgToEditView(CString St);
void Work(char* Name);
BOOL LoadSynaps(char* Name);
BOOL SaveSynaps(char *Name);
void Upd();
void CalcError(double* Mas);
void Go(double* Mas);
BOOL Learn();
```

Приложение Г (продолжение)

```
BOOL            Init(char* Name);
NetBP();
virtual          -NetBPQ;
Ъ
J ■>
```

Файл NetBP.cpp

```
//include "stdafx.h"
//include "MY_NET.h"
//include "NetBP.h"
//include "Math.h"

#ifdef _DEBUG
//undefTHIS_FILE
static char THIS_FILE[]=_FILE__;
//define new DEBUG_NEW
//endif

NetBP ::NetBP()
(
 N = NULL;
NeuronINLayers =      NULL;
CountLayers           =     0;
j

NetBP ::~NetBP()
r
t
```

Приложение Г (продолжение)

```
 if(N&&CountLayers)
 {
        for(UTNT i = 0;i<CountLayers;i++)

            delete[] N[i] ;
        }
        delete[] N;
 i
 if(NeuronINLayers) delete[] NeuronINLayers;

i

BOOL NetBP::OpenFullDB()
f
 if(m_MyNetDB.IsOpen()) m_MyNetDB.Close();


 i
        CString     FileName("my_net.mdb");
        CString     FileExt(".mdb");
        CString     FileFiltr("Ba3a данных Access (my_net.mdb)|my_net.mdb|") ;
        CFileFind fpr;
        if(! fpr.FindFile(FileName))
        {
            CFileDialogdlg(TRUE,FileExt,FileName,OFN_HIDEREADONLY | OFNO VER WRITEPROMPT,FileFiltr);
            if (dlg.DoModal()!=IDOK)
```

Приложение Г (продолжение)

```
            {
                return FALSE ;
            i
            I
            FileName = dlg.GetPathName();
        i
        I
        m_MyNetDB.Open(FileName);
}
catch                    (CDaoException*                    e)
r
        DisplayDaoException(e,NULL);
        return FALSE;

return TRUE;
i
c

BOOL NetBP::LoadFileList()
if
CFileFind fpr;
BOOL                        bFind=fpr.FindFile("\\misha\\my_net\\*.mdb");
while(bFind)
i
i
        bF ind=fpr.F indNextFile();
        CString name=fpr.GetFileTitIe();
        if(name=-'LostGroup")
            continue;
        POSITION pos;

for(pos=m_FileLst.GetHeadPosition();pos!=NULL;m_FileLst.GetNext(pos))
```

Приложение Г (продолжение)

```
        {
                CString curname=m_FileLst.GetAt(pos);
                if(name < curname)
                        break;
        }
        if(pos)
                m_FileLst.InsertBefore( pos, name);
        else
                m_FileLst.AddTail(name);
 }
 return TRUE ;
i
j

 void NetBP ::DisplayDaoException(CDaoException *e, LPCTSTR Ipcszlnfo)
i
 CString strName;
 i f(e-> m_p Erro r I n fb)
        strName.Format("Ошибка ДАО : %d",e->m_pErrorInfo->m_lErrorCode);
 else
        strName=_T("Ошибка ДАО");
 if(lpcszlnfo)
 r
 i
        strName=_T(" ")+strName;
        strName=_T(lpcszInfo)+strName;



 MessageBox(strName);
 e->Delete();
```

Приложение Г (продолжение)

```
void NetBP::MessageBox(CString Str)
i

 (CMY_NETApp*)AfxGetApp()->GetMainWnd()-
>MessageBox(Str,NULL,MB_OK);
}

BOOL NetBP::lnit(char *Name)
f
i
 if(Name=="MDB")
 if(LoadFileList()&&OpenFullDB())

        CString Str;
        double Niu;
        double Miu;
        Str.Format("HeHpoHHan сеть из базы %s",Name);
        MsgToEditView(Str);
        try
        I
            CDaoRecordset m_Rl (&m_MyNetDB);
            COle Variant      var;
            CString strSQL;

            strSQL-'SELECT * FROM Config";
            m_RLOpen(dbOpenSnapshot, strSQL, dbReadOnly);
            if(m_RLGetRecordCount()!=0)

                COleVariant        var;
```

Приложение Г (продолжение)

```
        m_Rl .GetFieldValue("Niu",var) ;
        Niu=var.dblVal;
        NiuPar=Niu;
        m_Rl .GetFieldValue("Miu",var);
        Miu=var.dblVal;
        m_Rl.GetFieldValue("Par",var);
        Kof=var.dblVal;
        m_Rl .GetFieldValue("Iter",var);
        Countltei-var. 1 Vai;
        m_Rl.GetFieldValue("Cikl",var) ;
        CountCikl=var.lVal;
i
i
else
I
I
        MessageBoxf'ERROR open config table!");

m_Rl.Close();


strSQL-"SELECT * FROM NET ORDER BY Layers";
m_R 1 .Open(dbOpenSnapshot, strSQL. dbReadOnly);
//Количество слоев в сети
CountLayers = m_Rl .GetRecordCount();
if(CountLayers==0) //Кол-во слоев = O
{
        m_Rl.Close();
        return FALSE;
```

Приложение Г (продолжение)

```
Str.Format("KonH4ecTBo слоев %d",CountLayers);
MsgToEditView(Str);
//Количесчво нейронов в каждом слое
NeuronINLayers = new UINTfCountLayers];
Str-'Количество нейронов в слоях:";
MsgToEditView(Str);
UINT i;
for(i=0;i<CountLayers;i++)
i
        m_Rl.GetFieldValue("Neurons",var);
        NeuronINLayers [i]        = var. 1 Vai;
        m_Rl ,MoveNext();
        Str.Format(" %d - %d",i,NeuronINLayers[i]);
        MsgToEditView(Str);
I
m_Rl.Close();

//Делаем сеть
//По слоям
N = new CNeuron*[CountLayers];
for(i=0;i<CountLayers;i++)
{
        N[i] = new CNeuron[NeuronINLayers[i]];
S

//Инициализируем нейроны сети
//N[i][j], i - номер слоя,] - номер нейрона в слое
for(i=0; i<CountLay ers; i++)
```

Приложение Г (продолжение)

```
t
        for(UINT j=0 ;j <N euronINLay ers [i] ;j++)
                        r
                        i
        //В первом слое количество аксонов = 1,
        //а в остальных по количеству нейронов
            if(i==0)
            i
                N[i][j].Init(i,j,l);
            }
            else
            \(
                N[i][j].lnit(i ,j,Neuron I NLayers[i-l ]);
                //делаем массив входных значений нейрона

                for(UINT k=0;k<Neuron!NLayers[i- 1 ];k++)
                {
                    N[i][j].in_Neuron[k]=N[i-
1 ] [k] .GetPointerQ;
                }
            s
            N[i][j].MiuPar=Miu;
            N[i][j].NiuPar=Niu;
            N[i][j].Kof=Kof;

    catch (CDaoException* e)
```

Приложение Г (продолжение)

```
                DisplayDaoException(e,NULL);
    return                                                    FALSE;
    i
    i
    }
    return                                                    TRUE;
    i
    j

    //include "MainFrm.h"
    BOOL NetBP::Learn()
    {
      CString Str;

      //Массивы обучающей выборки
      double** in_Array;                                                ;
      double** out_Array;

      UINT CountIN = 0;
      UINT CountOUT = 0;
      UINT CountStrIN;
      UINT CountStrOUT;
      UINT i;
      try
      { ■
            CDaoRecordset m_Rl (&m_MyNetDB);
            CString strSQL-'SELECT * FROM Learn WHERE Layers-0 ORDER
BY Layers";
            m_Rl.Open(dbOpenSnapshot,strSQL,dbReadOnly);
            CountIN = m_Rl.GetRecordCount();
```

Приложение Г (продолжение)

```
CountStrIN = CountIN/NeuronINLayersfO];
in_Array = new double*[CountStrIN];
for(i=0;i<CountStrIN;i++)
{
        in_Array[i] = new double[NeuronINLayers[0]];
}
m_Rl .MoveFirst();
for( i=0; i<CountIN; i++)
i
        COle Variant var;
      . m_Rl.GetFieldValue("NumNeuron",var);
        UINT nn=var.lVal;
        m_Rl ,GetFieldValue("NumStr",var);
        UINT ns=var.lVal;
        m_Rl .GetFieldValue("Value",var);

        double nv=var.dblVal;

        for(UINT j=0;j<CountStrIN;j++)

                for(UINT k=0;k<NeuronINLayers[0] ;k++)
                {
                        if(j==ns&&k==nn)
                                i
                                1
                           in_ Array [j ] [k] =n v;
                           //TRACE("\nin_Array[%d][%d] = %f',j,k,nv);
```

6

Приложение Г (продолжение)

```
        j
        m_Rl .MoveNextQ;
    }
    m_Rl.Close();

    strSQL.Format("SELECT * FROM Learn WHERE
Layers=%d" ,CountLayers-1);
    strSQL+-" ORDER BY Layers";
    m_Rl.Open(dbOpenSnapshot,strSQL,dbReadOnly);
    CountOUT = m_Rl .GetRecordCount();
    CountStrOUT = CountOUT/NeuronINLayersfCountLayers-l];
    out_Array = new double*[CountStrOUT];
    for(i=0; i<CountStrOUT;i++)
    {
        out_Array[i] = new double[NeuronINLayers[CountLayers-l]];
    }
    for(i=0;                    i<CountOUT;                    i++)
    i
    i
    COle Vari ant          var;
        m_Rl .GetFieldValue("NumNeuron",var) ;
        UINT nn=var.lVal;
        m_Rl .GetFieldValue("NumStr",var);
        UINT ns=var.lVal;
        m_Rl .GetFieldValue("Value",var) ;
        double nv=var.dblVal;
        for(UINT j=0 ;j <CountStrOUT ;j ++)
            for(UINT k=0;k<NeuronTNLayers[0];k++)
```

Приложение Г (продолжение)

```
                {
                    if(j=ns&&k==nn)
                    ï
                      out_Array [j ] [к] =nv;
                    }

        }
        m_Rl .MoveNext();
}
m_Rl.Close();
MsgLearn(CountStrIN,in_Array,out_Array);

//Рандомизация порядка подачи
//строк обучающей последовательности
        UINT* RndMas;
RndMas - new UINT[CountStrIN*4];
srand( (unsigned)time( NULL ));
for( i=0; i <C ount S trIN* 4; i++)
г
i
        RndMas [i] = rand()*CountStrIN/RAND_MAX;
        ASSERT(RndMas[i]>=0&&RndMas[i]<CountStrIN);
        TRACE("\nRnd - %d",RndMas[i]);
i
f


for(UINT CCC=0;CCC<=CountCikl;CCC++)

        CString Str;
```

Приложение Г (продолжение)

```
for(i=0;i<CountStrIN;i++)
                                        {
//                          TRACE("\nN Выборка - %d",i);
                    Str.Format("НоМер выіборки=%ci",i);
                    MsgToEditView("");
                    MsgToEditView(Str);
                    double AllError=100;
                    double OldError=AllError;
                    UINT Iter=0;
                    while((AHError>Kof)&&(Iter<CountIter))
                    {
                        Co(iп_Аггау[i]);//Прямое распространение

                        OldError=AllError;
                        AllError = 0.;
                        for(UINT j=0;j <NeuronINLay ers [Co untLay er s-1 ]
                        ;j ++)
                        {
                            AllError+=fabs(N[CountLayers-1 ] [j]. Axon-
out_Array[i][j]);
//                                  TRACE("\nOut=%f,
Axon=%f ',out_Array [i] [j] ,N[CountLayers-1 ] [j] .Axon);
                        i
```

Приложение Г (продолжение)

```
                    for(UINT ii=0;ii<CountLayers;ii++)
                    г
                        for(UINT yy=0;yy<NeuronINLayers[ii];yy++)

//                                  AllError+=fabs(N[ii][yy] .Error);
                                for(UINT
kk=0;kk<N[ii] [yy] .n_Delta;kk++)

                                    AllError+=fabs(N[ii][yy].Delta[kk]);
                                }

                        }
                    }

*/

                    i f(fabs(OldError)<=fabs( Al lError))
                     r
                    i




                    //Условие выхода из процесса обучения
                    //при невозможности его дальнейшего продолжения
/*                      if(fabs(AHError)>fabs(OldError))

                        MessageBoxf'Не могу обучиться!");
```

Приложение Г (продолжение)

```
                                    MsgToEditView("OcTaHOBKa процесса
обучения");
A11Errог=0.;
else
{
OldError=AllError;
}
*/

    //                              Str.Format("OniH6Ka=%f', AllError);
    //                              MsgToEditView(Str);

    И                               TRACE("\nAllError=%f', AllError);

                                Iter++;
                                if(Iter==CountIter-l)  MsgToEditView("He  могу
обучиться!");
                        }
                        8E.EогтаI("Суммарная  ошибка=%E',AllError);
                        MsgToEditVie w( Str);

                }
    //                  deletefl RndMas;

      catch (CDaoException* e)
```

Приложение Г (продолжение)

```
        DisplayDaoException(e,NULL);
        for(i=0;i<CountStrIN;i++)
        ¡
        ¡
                delete [] in_Array;
        ¡
        deletef] in_Array;

        for (1=0; i<CountStrOUT ;i++)

                deletef] out_Array[i];
        ¡
        ¡
        deletef] out_Array[i];

        return FALSE;


for(i=0;i<CountStrIN;i++)

        deletef] in_Arrayfi];
¡
¡
deletef] inArray;


for (i=0; i<CountStrOUT; i++)

        deletef] out_Arrayfi];

deletef] out_Array;
```

Приложение Г (продолжение)

```
 return TRUE;
i
/


void NetBP::Go(double* Mas)
t i
 //Первый слой
 UINT i;
 UINTj;
 for(i=0;i<NeuronINLayers[0];i++)
  i
        N[O][i].Synaps[O]=Mas[i];
        N[0][i].Go();
 }
 //и дальше по слоям
 for(i= 1; i<CountLayers;i++)
  i
  i
        for(j=0 ;j <NeuronINLay er s [i] ;j ++)



         1
         j


i


 void                    NetBP::CalcError(double*                    Mas)
c
i
 //Ошибка в последнем слое
 UINT i;
 double sum;
```

Приложение Г (продолжение)

```
for(i=0; i<NeuronINLayers [CountLay ers-1 ]; i++)
r
i
        N[CountLayers-l][i].SetCalcErr(Mas[i]);
}
//Расчет ошибки по остальным слоям
for(int j=CountLayers-2;j>=0;j—)
(
        //По нейронам в слое
        fo r( i=0; i <Neu ron INLay ers [ i ]; i ++)
        {
                sum = 0.;
                for(UINT k=0;k<NeuronINLayers[j+1 ] ;k++)
                i
                sum+=N[j+1 ] [k] .Error *N [j+1 ] [k]. Synaps [ i];
                }
                N[j] [i] .Error=sum*N[j] [i] .CalcErr();
        }
}


void NetBP::Upd()
i
 for(int i=CountLayers-l ;i>=0;i—)
 {
        for(UINT j=0; j <NeuronINLay ers [i] ;j ++)
         r
                N[i][j].Learn();
                N[i]|j].Update();
```

Приложение Г (продолжение)

```
BOOL NetBP ::SaveSynaps(char *Name)
{
 if(Name!="MDB") return FALSE;
 try
 {
        CDaoQueryDef qryReg(&m_MyNetDB);
        qryReg.Open("DeleteFromSynaps");
        qryReg.Execute();
        qryReg.Close();

        CDaoRecordset m_Rl (&m_MyNetDB);
        CString strSQL-'SELECT * FROM Synaps";
        m_Rl.Open(dbOpenDynaset,strSQL,dbAppendOnly);

        for(UINT i=l;i<CountLayers;i++)
        {
              for(UINT j=0 ;j <N euronINLay er s [ i] ;j ++)
                                    {
                    for(UINT k=0;k<N[i] |j] .n_Synaps;k++)
                                    {
                                m_Rl .AddNewQ;
```

Приложение Г (продолжение)

```
m_R 1. SetF ieldV alue("n_Layer" ,CO1e V ariant(long(i), VT_I4));

m_Rl.SetFieldValue("n_Neuron",COleVariant(long(j),VT_I4));

m_Rl ,SetFieldValue("n_Synaps",COleVariant(long(k),VT_I4));

m_Rl.SetFieldValue(" Value", C01eVariant(N[i][j].Synaps[k]));
                        m_Rl.Update();

                i
        j
        i
        /
        m_Rl.Close();
i
J
 catch (CDaoException* e)
i
        DisplayDaoException(e,NULL);
        return FALSE;
i
)
```

```
BOOL NetBP::LoadSynaps(char *Name)
{
 if(Name!="MDB") return FALSE;
 try
  i
        CDaoRecordset m_Rl(&m_MyNetDB);
        CString strSQL="SELECT * FROM Synaps ORDER BY nJLayer";
```

Приложение Г (продолжение)

```
m_Rl.Open(dbOpenSnapshot, strSQL, dbReadOnly);
UINT Count = m_Rl.GetRecordCount();
UINT RealCount = 0;
for(UINT b=0;b<Count;b++)

    COleVariant var_i;
    COleVariant var_J;
    COleVariant var_k;
    COleVariant var_v;
    m_Rl.GetFieldValue("n_Layer",var_i);
    m_RLGetFieldValue("n_Neuron",varJ);
    m RI .GetFieldValue("n_Synaps",var_k);
    m_Rl.GetFieldValue(" Value",var_v);
    for(UINT i=l;i<CountLayers;i++)
    {
        for(UINT j=0;j<NeuronINLayers(i];j++)
                    if
                for(UINT k=0;k<N[i] [j] ,n_Synaps;k++)


if((long(i)==var_i.lVal)&&(long(j)==varJ.lVal)&&(long(k)==var_k.lVal))
                    {
                        N[i][j].Synaps[k]=var_v.dblVal;
                        RealCount++;
                    }
                }
            }
        i
        f
```

Приложение Г (продолжение)

```
            m_Rl .MoveNextQ;
        r
        m_RLClose();
        if(Count!=RealCount) return FALSE;
i
/
 catch (CDaoException* e)
 i
 r
        DisplayDaoException(e,NULL);
        return FALSE;
-

 return TRUE;


void NetBP::Work(char *Name)

 if(Name=="MDB")

        UINT Count;
        UINT CountStr;
        double** in_Array;
        UINT i;
        try
        I
            //Читаем из базы значения входов (таблица in_Work)
            //в массив in_Array
            CDaoRecordset m_Rl (&m_MyNetDB);
```

```
CString strSQL-' SELECT * FROM in_Work ORDER BY
NumStr";
m_Rl.Open(dbOpenSnapshot,strSQL,dbReadOnly);
Count = m_Rl.GetRecordCount();
CountStr = Count/NeuronINLayersfO];
in_Array = new double*[CountStr];
for(i=0;i<CountStr;i++)

    in_Array[i] = new double[NeuronlNLaycrs[OJ];

m_Rl .MoveFirstQ;
for(i=0;i<Count;i++)

    COleVariant var;
    m_Rl .GetFieldValue("NumStr",var) ;
    UINT ns^var.lVal;
    m_Rl .GetFieldValue("NumNeuron",var);
    UINT nn=var.lVal;
    m_RI .GetFieldValue("Value",var);

    double nv=var.dblVal;
    for(UINT j=0;j<CountStr;j++)
                        f
        for(UINT k=0;k<NeuronINLayers[0];k++)

            if(j==ns&&k==rin)

                i n_ Arr ay [j ] [k]=n v;
```

Приложение Г (продолжение)

```
                                        //TRACE("\nin_Array [%d] [%d]
%f",j,k,nv);
                                    }
                                }
                            }
                            m_Rl .MoveNextQ;
                        s
                        m_Rl.Close();

                        //удаляем старые записи в табл (out_Work)
                        CDaoQueryDef qryReg(&m_MyNetDB);
                        qry Reg. Open( "DeleteFromOUT_ Work");
                        qryReg.ExecuteQ;
                        qryReg.CloseQ;
                        //
                        strSQL-' SELECT * FROM out_Work";
                        m_Rl.Open(dbOpenDynaset,strSQL,dbAppendOnly);

                        for(i=0; i<CountStr; i++)
                        {
                            Co(iп_Аггау[i]);//Прямое распространение
                            Msg W ork(in_Array [i]);
                            for(UINT j=0 ;j <NeuronINLayers [CountLay ers-1 ] ;j++)
                            {
                                m_Rl .AddNewQ;
```

Приложение Г (продолжение)

```
m_R 1. SetF ieldV alue( "NumNeuron" ,CO1e V ariant(long(j), VT_I4));

m_Rl. SetFieldValue(" Value", COleVariant(N[CountLayers-l]fj], Axon));
                        m_Rl.Update();


                m_Rl.Close();

                for(i=0;i<CountStr;i++)

                        deletef] in_Array[i];

                deletef] in_Array;

        catch (CDaoException* e)

                DisplayDaoException(e,NULL);




void NetBP::MsgToEditView(CString St)
c
i
  CMainFrame* pFrame =
  (CMainFrame*)((CMY_NETApp*)AfxGetApp()->GetMainWnd(.));
  ASSERT(pFrame);
  CMY_NETDoc* pDoc = (CMY_NETDoc*)(pFrame-
>GetActiveDocument());
```

Приложение Г (продолжение)

```
if (pDoc)
{
        POSITION pos = pDoc->GetFirstViewPosition();
        CMY_NETView* pView =                    (CMY_NETView*)(pDoc-
>GetNextView( pos ));
        if (pView)
        Г                                                                    :
        X
                pView->Message(St);
        i
        j
```

```
void NetBP::MsgLearn(UINT Count,double** in_Array,double** out__Array)
{
 CString Str;
 Str="Обучающая выборка:";
 MsgToEditView(Str);
 UINT i;
 for(i=0;i<Count;i++)
 Г
 i
        Str="";
        UlNTj;
        for(j=0 ;j <NeuronINLay ers [0] ;j++)
        f
                CString Stl;
                Stl.Format(" %f :",in_Array[i][j]);
                Str+=Stl;
```

Приложение Г (продолжение)

```
/*              for(j=0;j<NeuronINLayers[CountLayers-1 ] ;j++)
        i
i
                CString Stl;
                Stl.Format(" %f :",out_Array[i]|j]);
                Str+=Stl;
        }
        MsgToEditView(Str);
*/
  )
i
i


void NetBP::MsgWork(double* in_Array)
{
  CString Str;
  Str="Результат работы:";
  MsgToEditView(Str);
  UINT i;
  Str-'";
/*      for( i =0; i <N euronINLay ers [0]; i++)
  f
         CString Stl;
         Stl.Format(" °/°f :",in_Array[i]);
         Str+=Stl;
  3
  i
*/
  for(i=0;i<NeuronINLayers[CountLayers-1 ] ;i++)
  (
i
```

Приложение Г (продолжение)

```
        CString Stl;
        Stl.Format(" %f :",N[CountLayers-l][i].Axon);
        Str+=Stl;
    }
 MsgToEditView(Str);
}

void NetBP ::Impuls()
{
 UINT к = rand()*NeuronINLayers[0]/RAND_MAX;
 N[O][k].Impuls();
}
```

Міністерство транспорту України
ДОНЕЦЬКИЙ ІНСТИТУТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ
Харківської державної академії залізничного
340018, м. Донецьк, вул. Горна, 6
тел. (0622) 51-28-31 факс (0622) 91-52-63, 51-23-54
р/р 35308308801 в ОПЕРУ НБУ м.Донецька

ЗАТВЕРДЖУЮ :
Ректор Донецького інституту
залізничного транспорту

В.И. Поддубняк

АКТ

про впровадження результатів кандидатської дисертаційної роботи
Чепцова Михайла Миколайовича

Акт складено про те, що результати дисертаційної роботи були використані при розробці навчального лабораторного макету “Електрична централізація проміжної станції з мікропроцесорним маршрутним набором” на кафедрі “Автоматика телемеханіка зв’язок та обчислювальна техніка” Донецького інституту залізничного транспорту у наступному вигляді:

1) експериментальних даних по дослідженню моделі функціонування електричної централізації проміжної станції;
2) технічних пропозицій по розробці апаратних засобів, алгоритмів функціонування;
3) програмного забезпечення.

Використання вказаних результатів дозволяє:

підвищити методичний рівень викладання курсу “Автоматизовані комп’ютерні системи на станціях”;

проводити лабораторні та практичні роботи, дипломне проектування, із застосуванням сучасних апаратних і програмних засобів;

проводити експериментальні дослідження на макеті, з метою подальшого впровадження на виробництві.

Проректор з навчально-виховної роботи Ю. В.Тимохін

Декан факультету “Інфраструктура залізничного трансіда-II С. Н. Харитонова

Міністерство транспорту України
Укрзалізниця
Донецька залізниця
ІНФОРМАЦІЙНО-
СТАТИСТИЧНИЙ ЦЕНТР
Код ЄДРПОУ 01071934
83000, м. Донецьк, бул. Пушкіна, 11
телефон 91-64-92
Р/р 26000000003 МФО 335838
Донецька філія АБ "Експрес-Банк"
12 10 2001 р.
№ 1320

Ш ЗАЖРДЖУЮ
^/ІНФОРМАШИМр-^Ж тг-тт тт ....
•S СТАтистичі$§рідаик ІСЦ Донецької залізниці
*\ ЗАЛІЗНИЦІ ^«навиЙ/ХЙ, .
■УОк

АКТ

про впровадження результатів кандидатської дисертаційної роботи
Чепцова Михайла Миколайовича у дослідну експлуатацію.

Акт складено про те, що результати дисертаційної роботи були використані при розробці програмно-технічного комплексу «АРМ електромеханіка ДЦ ЛУЧ» на ІСЦ Донецької залізниці у наступному вигляді:

1) експериментальних даних по дослідженню викривлень сигналу ТС системи диспетчерської централізації "ЛУЧ";
2) рекомендацій по розробці програмного забезпечення комплексу «АРМ электромеханика ДЦ ЛУЧ».

Результати дисертаційної роботи впроваджено у дослідну експлуатацію на ділянці Мандрикіно-Рутченково Донецької залізниці.

Використання вказаних результатів дозволяє:

1) підвищити рівень вірогідності інформації у сигналі ТС диспетчерської централізації "ЛУЧ";
2) зменшити витрати на розробку програмного забеспечення.

Заступник керівника ІСЦ
Донецької залізниці Жевжик Є. Г.

Керівник відділу ІСЦТП
Донецької залізниці Голков А. В.

З оригіналом згідно.
Вчений секретар спеціалізованої вченої ради Д 64.820.04

МІНІСТЕРСТВО ТРАНСПОРТУ УКРАЇНА
ХАРКІВСЬКА ДЕРЖАВНА АКАДЕМІЯ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ
№01116472

МІНІСТЕРСТВО ТРАНСПОРТУ УКРАЇНИ
ДЕРЖАВНЕ ПІДПРИЄМСТВО
ЦЕНТР СТАНДАРТИЗАЦІЇ, МЕТРОЛОГІЧНОГО ТА НОРМАТИВНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ ТА АВТОМАТИЧНИХ СИСТЕМ УПРАВЛІННЯ

ЗАТВЕРДЖУЮ

Керівник ЦСМ АСУ УПП ЗТ

_____ **В.М. Бутенко**

I АКТ № *01-2001*

про впровадження результатів кандидатської дисертаційної роботи
Чепцова Михайла Миколайовича

Науково-технічна рада у складі:
голова ради -j- Максименко О. М., головний інженер ЦСМ АСУ УПП ЗТ;

члени ради — 4- Кривошей Б. О., ст. науковий співробітник ЦСМ АСУ УПП ЗТ;
— Шевченко Р. В., в.о. ст. наукового співробітника ЦСМ АСУ УПП ЗТ;
— Азаров М. В., в.о. ст. наукового співробітника ЦСМ АСУ УПП ЗТ;
відп. секретар — Іванова І. М., ст. інженер ЦСМ АСУ УПП ЗТ;

скликана наказам по ЦСМ АСУ УПП ЗТ № 15-2001 від 11.04.2001 року в період з 12.04.2001 р. по 13.04.2001 р. розглянула кандидатську дисертаційну роботу Чепцова М.М. на науково-технічній раді фахівців ЦСМ АСУ УПП ЗТ та склала цей акт про те, що результати ^дисертаційної роботи б іули використані в науково-дослідній роботі № 904-28/99-705.99-Цтех від 01.04.99 р. “Розробка автоматизованої системи знімання інформації про проходження поїздів по міждержавних і міждорожних стикових пунктах” при розробці алгоритмів та підпрограм виявлення рухомої одиниці та часупроходження стикового пункту в наступному вигляді:

1) технічних пропозицій по розробці алгоритмів роботи програмного забезпечення автоматизованої системи знімання інформації про проходження поїздів по міждержавних і міждорожних стикових пунктах (АСІП СП);

2) експериментальних даних по дослідженню викривлень сигналів ТС систем диспетчерської централізації;

3) методик розрахунку та моделювання проходження поїзду по елементам шляхового розвитку полігону диспетчерського управління;

4) рекомендацій ро розробці програмного забезпечення АСІП СП.

Використання вказаних результатів дозволяє: підвищити якість проектування та ефективність роботи систем такого Ьівня; скоротити витрати на проведення науково-дослідних та дослідно-конструкторських робіт і натурних випробувань; підвищити продуктивність праці оперативного ■персоналу.

Голова комісії _____ ***О. М. Максименко***

Члени комісії _____ ***Б. О. Кривошей***

_____ ***Р. В. Шевченко***

_____ ***М. В. Азаров***

_____ ***І. М. Іванова***

Відповідальний секретар _____ ***Т. В. Мелкумян***

З оригіналом згідно.
Вчений секретар спеціалізованої вченої ради Д 64.052.01 В.М. Зотова

13 ____ 2001 р.