Московский государственный технический университет
имени Н.Э. Баумана

СОВРЕМЕННЫЕ

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Сборник трудов кафедры ИУ-6
МГТУ имени Н.Э. Баумана

Москва
НИИ РЛ МГТУ им. Н.Э. Баумана
2011

УДК 004.2/9(048.2)
ББК 32.973
С56

Современные информационные технологии: Сб. трудов каф.,
С56 МГТУ им. Н.Э. Баумана / Московский гос. технический ун-т им. Н.Э. Баумана, фак. «Информатика и системы упр.», каф. ИУ-6 «Компьютерные системы и сети» - М.: НИИ РЛ МГТУ им. Н.Э. Баумана, 2011. - 184 с.

ISBN 978-5-98669-029-2

Настоящий сборник трудов кафедры «Компьютерные системы и сети» факультета «Информатика и Системы управления» МГТУ им. Н.Э. Баумана составлен по материалам ежегодной межвузовской научно-технической конференции аспирантов, студентов и молодых ученых «Современные информационные системы и технологии», состоявшейся в ноябре 2010 года, а также по материалам научных трудов преподавателей кафедры.

В сборник включены статьи по широкому спектру направлений информатики, отвечающих научным направлениям работы аспирантов, студентов, молодых ученых и преподавателей кафедры. Основными из них являются: проблемы оценки информационных параметров сигналов, компьютерные технологии моделирования, проектирование языков представления знаний, управление корпоративной сетью, параллельные вычисления и системное программирование и др.

Авторы надеются, что публикуемые материалы будут полезны заинтересованным читателям.

Статьи печатаются в авторской редакции.

УДК 004.2/9(048.2)
ББК 32.973

ISBN 978-5-98669-029-2

СОДЕРЖАНИЕ

УДК 004

СИСТЕМЫ ОБНАРУЖЕНИЯ УЯЗВИМОСТЕЙ И ВТОРЖЕНИЙ

Анохин С.Д., Ващенко Б.И.

Для успешной защиты беспроводной сети, необходимо контролировать и анализировать трафик этой сети.

Основные задачи, решаемые с помощью современных средств контроля:

1) современные средства контроля измеряют уровни сигнала в любой точке зоны обслуживания сети;

2) фиксировать служебную информацию и данные, фиксировать повторные передачи и определять активные узлы сети (порождающие наибольший трафик);

3) с помощью средств контроля можно определить, какое устройство было переведено на меньшую скорость и принять соответствующие меры;

4) с помощью средств контроля можно обеспечить анализ трафика сетей на предмет определения используемых механизмов защиты;

5) поддержка протоколов как низкого, так и высокого уровней. Большинство современных средств контроля поддерживают такой режим работы.

На сегодняшний день разработано большое количество программных и аппаратных средств, позволяющих осуществлять мониторинг, обработку и анализ трафика сети, построенной на оборудовании семейства стандартов IEEE 802.11 (a, b). Эти средства могут контролировать работающие радиоустройства в диапазонах 2,4-2,5 ГГц, 5,15-5,25 ГГц, 5,25-5,35 ГГц и 5,725-5,825 ГГц. [1]

Система распознавания атак — программное или аппаратное средство, предназначенное для выявления фактов неавторизованного доступа в компьютерную систему или сеть либо несанкционированного управления ими в основном через Интернет. Соответствующий английский термин – *Intrusion Detection System (IDS)*. Системы обнаружения вторжений обеспечивают дополнительный уровень защиты компьютерных систем.

IDS автоматизируют процесс просмотра событий, возникающих в компьютерной системе или сети, и анализируют их с точки зрения безопасности.

Обнаружение проникновения является процессом мониторинга событий, происходящих в компьютерной системе или сети, и анализа их. **Проникновения** определяются как попытки компрометации конфиденциальности, целостности, доступности или обхода механизмов безопасности компьютера или сети. Проникновения могут осуществляться как атакующими, получающими доступ к системам из Интернета, так и авторизованными пользователями систем, пытающимися получить дополнительные привилегии, которых у них нет.

IDS состоят из трех функциональных компонентов: информационных источников, анализа и ответа. Система получает информацию о событии из одного или более источников информации, выполняет определяемый конфигурацией анализ данных события и затем создает специальные ответы – от простейших отчетов до активного вмешательства при определении проникновений.

Каждое средство защиты адресовано конкретной угрозе безопасности в системе. Более того, каждое средство защиты имеет слабые и сильные стороны. Только комбинируя их, можно защититься от максимально большого спектра атак.

Firewall'ы являются механизмами создания барьера, преграждая вход некоторым типам сетевого трафика и разрешая другие типы трафика. Создание такого барьера происходит на основе политики firewall'а. IDS служат механизмами мониторинга, наблюдения активности и принятия решений о том, являются ли наблюдаемые события подозрительными. Они могут обнаружить атакующих, которые обошли firewall, и выдать отчет об этом администратору, который, в свою очередь, предпримет шаги по предотвращению атаки.

IDS становятся необходимым дополнением инфраструктуры безопасности в каждой организации. Технологии обнаружения проникновений не делают систему абсолютно безопасной. Использование IDS помогает достичь нескольких целей:

1) возможность иметь реакцию на атаку позволяет заставить атакующего нести ответственность за собственную деятельность. Это определяется следующим образом: **«Я могу прореагировать на атаку, которая произведена на мою систему, так как я знаю, кто это сделал или где его найти»**;

2) возможность блокирования означает возможность распознать некоторую активность или событие как атаку и затем выполнить действие по блокированию источника. Данная цель определяется следующим образом: **«Я не забочусь о том, кто атакует мою систему, потому что я могу распознать, что атака имеет место, и блокировать ее»**;

3) возможность определения преамбул атак, обычно имеющих вид сетевого зондирования или некоторого другого тестирования для обнаружения уязвимостей, и предотвращения их дальнейшего развития;

4) выполнение документирования существующих угроз для сети и систем;

5) обеспечение контроля качества разработки и администрирования безопасности, особенно в больших и сложных сетях и системах;

6) получение полезной информации о проникновениях, которые имели место, с предоставлением улучшенной диагностики для восстановления и корректирования вызвавших проникновение факторов;

7) IDS помогает определить расположение источника атак по отношению к локальной сети (внешние или внутренние атаки), что важно при принятии решений о расположении ресурсов в сети.

Типы IDS. Существует несколько способов классификации IDS, каждый из которых основан на различных характеристиках IDS. Тип IDS следует определять, исходя из следующих характеристик:

- **Способ мониторинга системы.** По способам мониторинга системы делятся на network-based, host-based и application-based;
- **Способ анализа.** Это часть системы определения проникновения, которая анализирует события, полученные из источника информации, и принимает решения, что происходит проникновение. Способами анализа являются обнаружение злоупотреблений (misuse detection) и обнаружение аномалий (anomaly detection);
- **Задержка во времени** между получением информации из источника и ее анализом и принятием решения. В зависимости от задержки во времени, IDS делятся на interval-based (или пакетный режим) и real-time.

Большинство коммерческих IDS являются real-time network-based системами.

К характеристикам IDS также относятся:

- **источник информации.** IDS может использовать различные источники информации о событии для определения того, что проникновение произошло. Эти источники могут быть получены из различных уровней системы, из сети, хоста и приложения;
- **ответ.** Набор действий, которые выполняет система после определения проникновений. Они обычно разделяются на активные и пассивные меры, при этом под активными мерами понимается автоматическое вмешательство в некоторую другую систему, под пассивными мерами – отчет IDS, сделанный для людей, которые затем выполнят некоторое действие на основе данного отчета.

Системы анализа и оценки уязвимостей. Инструментальные средства анализа уязвимостей (известные также как оценка уязвимостей) тестируют сеть или хост для определения наличия уязвимостей для известных атак. Анализ уязвимостей предоставляет дополнительную информацию для систем обнаружения проникновения. Используемыми источниками информации являются атрибуты состояния системы и выходные данные осуществленных атак. Источники информации являются частью механизма оценки. Интервал времени анализа либо является фиксированным, либо определяется параметром в пакетном режиме, типом анализа является определение злоупотреблений. Это означает, что системы оценки уязвимостей являются пакетным режимом детекторов злоупотреблений, которые оперируют с информацией о состоянии системы, и результатом становятся специальные тестовые подпрограммы. [2]

Анализ уязвимостей – очень сильная технология управления безопасностью, но эта технология является лишь дополнением к использованию IDS, а отнюдь не ее заменой.

Литература:

1. http://www.nestor.minsk.by/sr/2006/01/sr60123.html - сетевые решения A-Z
2. http://citforum.ru/security/internet/firewalls_ids/ - CIT FORUM

УДК 681.3.07

ВЫЯВЛЕНИЕ И ФОРМИРОВАНИЕ СВЯЗЕЙ МНОГИЕ - КО МНОГИМ В ЗАПОЛНЕННЫХ РЕЛЯЦИОННЫХ ТАБЛИЦАХ

А.В. Брешенков

В статье рассмотрим пример двух реляционных таблиц, между которыми возможна связь типа многие - ко многим. Неформально, а затем формально с использованием примера описан алгоритм выявления множественных зависимостей. Предложен алгоритм избавления от лишних атрибутов, алгоритм формирования 3-й таблицы для реализации многозначных связей. Рассмотрены особенности использования существующих инструментальных средств проектирования баз данных для выявления и формирование связей типа многие - ко многим.

В работах [1-3] обоснована актуальность проблемы преобразования заполненных нереляционных таблиц в реляционные таблицы, сформулированы задачи преобразования, намечены пути решения отдельных задач. Здесь рассматривается одна из этих задач – выявление и формирование связей между таблиц. Типы связей между таблицами рассмотрены в теоретических работах по базам данных, в частности в [4].

Рассмотрим пример отношений «Авторы» (А) и «Книги» (В), которые приведены соответственно в табл. 1 и в табл. 2.

Таблица 1

NA	ФАМИЛИЯ	ПУБЛИКАЦИИ	ГОРОД
1	Иванов	Книга 1	Балашиха
2	Иванов	Книга 2	Балашиха
3	Иванов	Книга 3	Балашиха
4	Петров	Книга 1	Ногинск
5	Петров	Книга 3	Ногинск
6	Сидоров	Книга 2	Козельск
7	Сидоров	Книга 3	Козельск

Таблица 2

NK	КНИГА	АВТОР	ИЗДАТЕЛЬСТВО	ТИРАЖ
1	Книга 1	Иванов	Эльдорадо	10000
2	Книга 1	Петров	Эльдорадо	10000
3	Книга 2	Иванов	Континент	7000
4	Книга 2	Сидоров	Континент	7000
5	Книга 3	Иванов	Пламя	15000
6	Книга 3	Петров	Пламя	15000
7	Книга 3	Сидоров	Пламя	15000

Нетрудно заметить, что соавтором Иванова является Петров; соавтором Сидорова является Иванов; соавторами Петрова являются Иванов и Сидоров. В связи с этим и в отношении А, и в отношении В для отражения этого факта потребовалось по 7 записей. В обоих отношениях могут быть задействовано множество столбцов (атрибутов) и значительное число строк (записей), а для их хранения требуются соответствующие объемы памяти.

Представленную ситуацию можно существенно улучшить, если отдельно хранить в одной таблице только необходимые данные об авторах (без дублирования фамилий), в другой таблице хранить только необходимые данные о книгах (без дублирования названий). А для отражения фактов авторства ввести третью таблицу, в которой представить номера авторов и соответствующие номера книг, которые они написали.

Табл. 1 после указанных преобразований примет вид таблицы 3.

Таблица 3

NA	Ф.И.О.	ГОРОД
1	Иванов	Балашиха
2	Петров	Ногинск
2	Сидоров	Козельск

Табл. 2 после указанных преобразований примет вид таблицы 4.

Таблица 4

NK	КНИГА	ИЗДАТЕЛЬСТВО	ТИРАЖ
1	Книга 1	Эльдорадо	10000
2	Книга 2	Континент	7000
3	Книга 3	Пламя	15000

Третья таблица, в которой представлены номера авторов и соответствующие номера книг, которые они написали, примет вид табл. 5.

Таблица 5

NA	NK
1	1
1	2
1	3
2	1
2	3
3	2
3	3

Посредством этой третьей таблицы и организуются связи «∞ : ∞» между таблицы с авторами и таблицы с книгами.

Выгода такого преобразования заполненных таблиц очевидна даже для приведенного небольшого примера. Она будет значительно больше, если для проектирования БД используются реальные таблицы. В связи с этим представляет теоретический и практический интерес метод выявления и формирования связей «∞ : ∞» между заполненными реляционными таблицами, о котором пойдет речь далее.

Из приведенного примера видно, что в табл. 1 и 2 Z(Фамилии) = Z(Автор), Z(Публикации) = Z(Книги), где Z(A) – значения атрибута «A». Таким образом, для выявления множественных зависимостей необходимо сравнить во всех парах таблиц множества значений неключевых атрибутов. Если для каких – либо двух таблиц найдется пара одинаковых множеств, то

это означает, что между таблицами существует многозначная зависимость и ее нужно устранить.

Пусть A_i – неключевой атрибут отношения R1;

A_m – другой неключевой атрибут отношения R1;

B_j – неключевой атрибут отношения R2;

B_s – другой неключевой атрибут отношения R2;

Тогда между отношениями R1 и R2 имеется множественная зависимость, если выполняется следующее условие:

$(E\ A_i)\ (R1 \in A_i)\ (\exists A_m)\ (R1 \in A_m)\ (\exists B_j)\ (R2 \in B_j)\ (\exists B_s)\ (R2 \in B_s)$

$(Z(A_{it}) = Z(B_{jp}))\ (Z(A_{mt}) = Z(B_{sp}))$, где

$i = 1, n-1;\ m = 1,n-1;\ j = 1, k-1;\ s = 1, k-1.$

Здесь n – степень отношения R1;

k – степень отношения R2;

t – номер записи в отношении R1;

p – номер записи в отношении R2;

$R1 = (A_1, \ldots, A_i, \ldots, A_n)$;

$R2 = (B_1, \ldots, B_j, \ldots, B_k)$.

Следует обратить внимание, что $i = 1, n-1;\ m = 1,n-1;\ j = 1, k-1;\ s = 1, k-1$ в связи с тем, что ключевые атрибуты не рассматриваются (в каждом отношении предполагается, что для формирования первичного ключа используется по одному атрибуту).

Алгоритм выявления множественных зависимостей между отношениями R1 и R2.

```
FOR i = 1 то n
    FOR t = 1 то tm
        FOR j = 1 то k
            FOR p = 1 то pm
                IF ((Z(Ait) = Z (Bjp))  and (i <> ik) and (j <> jk) THEN
                    ia1 = i
                    is = t
                    ja1 = j
                    js = p
                END IF
            NEXT p
```

```
            NEXT j
        NEXT t
    NEXT i
    FOR i = 1 то n
        FOR t = 1 то tm
            FOR j = 1 то k
                FOR p = 1 то pm
                    IF ((Z(Ait) = Z (Bjp))   and (i  <> ik) and (j  <> jk) and
                       (i <> ia1) and (j <> ja1) and (t = is) and (p = js)
                    THEN
                        ia2 = i
                        ja2 = j
                        PRINT ("Возможны множественные связи !")
                        PRINT  ("Атрибут R1",  i)
                        PRINT  ("Атрибут R2",  j)
                        PRINT  ("Значение атрибута R1",  Z(Ait)j )
                        PRINT  ("Значение атрибута R2", Z (Bjp) )
                        PRINT  ("Связь по атрибуту R1", ia1,  ia2)
                        PRINT  ("Связь по атрибуту R2", ja1,  ja2)
                        PRINT  ("Запись в R1", is)
                        PRINT  ("Запись в R2", js)
                    END IF
                NEXT p
            NEXT j
        NEXT t
    NEXT i
```

Здесь tm – мощность множества R1;

pm – мощность множества R2;

ik – столбец отношения R1, в котором расположен ключевой атрибут;

jk – столбец отношения R2, в котором расположен ключевой атрибут;

Остальные обозначения прокомментированы в операторах PRINT.

Поясним работу алгоритма.

В первом фрагменте алгоритма перебираются все столбцы и строки отношения R1, а также все столбцы и строки отношения R2. Если обнаружено совпадение значений атрибутов и они – неключевые, то

запоминаются номера атрибутов и номера записей в обоих отношениях. Во втором фрагменте алгоритма также перебираются все столбцы и строки отношения R1, а также все столбцы и строки отношения R2. Если обнаружено совпадение значений атрибутов и они неключевые и не те, которые уже обнаружены, и они находятся в тех же строках отношений, в которых уже обнаружены одинаковые значения атрибутов, то, вероятно, обнаружились множественные связи. В этом случае запоминаются номера атрибутов обеих отношений, значения которых совпадают. Номера двух пар атрибутов отношений R1 и R2 «подозрительных» на множественные связи выводятся на печать. Кроме того, на печать выводятся строки отношений R1 и R2, в которых найдены «подозрительные» значения.

В связи с тем, что рассмотренный алгоритм не дает гарантии того, что с помощью его выявятся множественные зависимости, то разработчику БД необходимо визуально оценить предложенные атрибуты и принять решение о необходимости выполнения следующих шагов алгоритма.

Избавление от лишних атрибутов.

Если наличие множественных связей подтвердилось, то необходимо в отношении R1 избавиться от атрибута A_{ia2} , а в отношении R2 избавиться от атрибута B_{ja1} .

Для этого формируется новое отношение $R1' = R1 – R(A_{ia2}) – R(A_{ik})$.

Кроме того, формируется также отношение $R2' = R2 – R(B_{ja1}) – R(A_{jk})$.

Другими словами, в R2 остается атрибут A_{ia1} – характерный для R1 и удаляется атрибут A_{ia2} – характерный для R2.

И, наоборот, в отношении R2 остается атрибут B_{ja2} – характерный для R2 и удаляется B_{ja1} – характерный для R2.

Под характерными атрибутами подразумеваются атрибуты, посредством которых ликвидируются множественные связи. В частности для отношения «Авторы» таким атрибутом является «Фамилия», а для отношения «Книги» таким атрибутом является «Наименование».

$R(A_{ik})$ и $R(A_{jk})$ – столбцы с ключевыми атрибутами.

Результаты преобразования такого рода проиллюстрированы соответственно в табл. 3 и 4. В таблицах присутствует ключевой атрибут, но этот атрибут является результатом выполнения следующего шага алгоритма.

При выполнении рассмотренного фрагмента алгоритма исключения и назначения ключевых атрибутов оправданно и полезно участие разработчика БД. Таким образом, следует в очередной раз обратить внимание на то, что предлагается концепция автоматизированного проектирования БД, а не автоматического проектирования. Эта концепция здесь и далее предлагает использование человеко-машинных процедур.

Как видно из последних выражений, в отношениях R1 и R2 исключаются и ключевые атрибуты. Эти атрибуты на данном шаге алгоритма становятся неактуальными, и впоследствии предполагается сформировать новые ключи.

Следующим шагом метода является исключение повторяющихся записей в отношениях R1’ и R2’.

Ниже представлен алгоритм исключения повторяющихся записей в отношении R1’.

FOR i= 1 то m

sa = sa_i

FOR i1 = i + 1 то m

IF sa_{i1} = sa THEN DELETE(sa_{i1})

NEXT i1

NEXT i

Если отношения представлены в формате БД, то на SQL соответствующий запрос исключения повторяющихся записей выглядит следующим образом.

SELECT DISTINCT Авторы.Фамилия, Авторы.Публикация, Авторы.[E-mail]

INTO Авторы1

FROM Авторы;

Здесь из таблицы «Авторы» формируется таблица «Авторы1», в нее включаются три поля исходной таблицы. При этом конструкция «DISTINCT» позволяет исключить повторяющиеся записи.

Ниже представлен алгоритм исключения повторяющихся записей в отношении R2’.

FOR i= 1 то m

sb = $sв_i$

FOR i1 = i + 1 то m

IF $sb_{i1} = sb$ THEN DELETE(sb_{i1})

NEXT i1

NEXT i

Если отношения представлены в формате БД, то на SQL соответствующий запрос исключения повторяющихся записей выглядит следующим образом.

SELECT DISTINCT Книги.Книга, Книги.Издательство, Книги.Тираж INTO Книги1

FROM Книги;

Далее для отношений R1’ и R2’ необходимо сформировать новые первичные ключи. Наиболее просто и эффективно это осуществляется посредством добавления в описанные таблицы нового ключевого поля типа «СЧЕТЧИК». В результате получаются отношения вида табл. 3 и 4.

Формирование 3-й таблицы для реализации многозначных связей.

Для формирования таблицы связей необходимо сканировать отношение R1'.

Для каждого значения поля этого отношения необходимо найти соответствующее значение в R1 (в исходном отношении).

Для этого значения необходимо выбрать соответствующее характерное значение отношения R2 (оно принадлежит той же записи отношения R1).

В отношении R2’ необходимо найти это характерное значение.

Первому атрибуту отношения R3 необходимо присвоить значение ключевого атрибута текущей записи отношения R1’, а второму атрибуту R3 – присвоить значение ключевого атрибута найденной записи отношения R2’.

Неформальный алгоритм формирования 3-й таблицы осуществляется следующим образом:

П1: Выбирается очередное значение характерного атрибута R1’ - $a'_{к1}$ до тех пор, пока они не исчерпаются, при этом запоминается значение соответствующего ключевого атрибута $ka'_к$.

П2: В отношении R1 ищется значение соответствующего характерного атрибута, равного a_{i1}. Найденные значения в предыдущих итерациях не рассматриваются. Если такого значения не находится, то выполняется переход к П6.

П3: В записи с найденным значением характерного атрибута выбирается значение атрибута, характерного для отношения R2 - a_n.

П4: В отношении R2’ ищется значение характерного соответствующего атрибута a_n ($b'_m = a_n$). Запоминается значение ключевого атрибута соответствующее b'_m (kb'_m).

П5: В отношение R3 формируется запись. В качестве значения 1-го атрибута отношения R3 используется ka'_i. В качестве значения 2-го атрибута отношения R3 используется kb'_m. Переход к П2.

П6: Переход к П1.

Формальный алгоритм формирования 3-го отношения выглядит следующим образом.

```
c = 0
FOR k = 1 то kv
   k1 = ka_к’
   aк = a'_к1
    M1: FOR n = 1 то nv
       IFa_n1 = aк THEN
           an = a_n2
           a_n1 = NULL
           F = TRUE
           EXIT FOR
       ELSE
           F = FALSE
       END IF
   NEXT n
   IF (F =  FALSE) THEN EXIT FOR
   FOR m = 1 то mv
       IF b’_m2 = an THEN
          k2 = kb’_m
          EXIT FOR
       END IF
   NEXT m
   CREATE S FOR R3
   c = c + 1
   d_c1 = k1
   d_c2 = k2
   IF F THEN GOTO M1
```

NEXT k

Здесь kv – число записей в отношении R1';

nv – число записей в отношении R1;

mv – число записей в отношении R2';

$ka_к$' – ключевой атрибут в k-й записи отношения R1';

$a'_{к1}$ – 1-й характерный атрибут в k-й записи отношения R1';

a_{n2} – 2-й характерный атрибут в n-й записи отношения R1;

b'_{m2} – 2-й характерный атрибут в m-й записи отношения R2';

kb'_m – ключевой атрибут в m-й записи отношения R2';

d_{c1} – 1-й характерный атрибут в c-й записи отношения R3;

d_{c2} – 2-й характерный атрибут в c-й записи отношения R3;

Оператор «CREATE S FOR R3» создает новую запись в отношении R3

Рассмотрим, что можно сделать по поводу выявления связей многие - ко многим с помощью визуального анализа содержимого таблиц и стандартных средств СУБД.

На рис. 1 представлена таблица «Авторы» в формате Microsoft Access, которая претендует на роль таблицы, принадлежащей связи многие - ко многим.

	NA	Фамилия	Публикации	Город
▶	1	Иванов	Книга 1	Балашиха
	2	Иванов	Книга 2	Балашиха
	3	Иванов	Книга 3	Балашиха
	4	Петров	Книга 1	Ногинск
	5	Петров	Книга 3	Ногинск
	6	Сидоров	Книга 2	Козельск
	7	Сидоров	Книга 3	Козельск
*	(Счетчик)			

Рис. 1. Таблица «Авторы», которая претендует на роль таблицы, принадлежащей связи многие - ко многим

На рис. 2 представлена таблица «Издательства» в формате Microsoft Access, которая претендует на роль таблицы, принадлежащей связи многие - ко многим.

	NK	Книга	Автор	Издательство	Тираж
▶	1	Книга 1	Иванов	Эльдорадо	10000
	2	Книга 1	Петров	Эльдорадо	10000
	3	Книга 2	Иванов	Континент	7000
	4	Книга 2	Сидоров	Континент	7000
	5	Книга 3	Иванов	Пламя	15000
	6	Книга 3	Петров	Пламя	15000
	7	Книга 3	Сидоров	Пламя	15000
*	(Счетчик)				0

Рис. 2. Таблица «Издательства», которая претендует на роль таблицы, принадлежащей связи многие - ко многим

Визуальный анализ приведенных на рисунках таблиц позволяет предположить, что они связаны между собой. С одной стороны связи реализуются посредством полей «Фамилия» и «Публикации» (таблица «Авторы»). С другой стороны связи реализуются посредством и полей «Книга» и «Автор» (таблица «Издательство»).

В дальнейшем будем действовать в соответствии с вышеизложенными алгоритмами.

На основе таблицы «Авторы» сформируем новую таблицу «Авторы1», в которой исключено ключевое поле, поле «Публикации» и дублирование записей. Соответствующий запрос на создание таблицы выглядит следующим образом:

```
SELECT DISTINCT Авторы.Фамилия, Авторы.Город INTO Авторы1
FROM Авторы;
```

Конструкция DISTINCT позволяет исключить повторяющиеся записи. В результате выполнения данного запроса сформируется таблица, представленная на рис. 3.

	Фамилия	Город
▶	Иванов	Балашиха
	Петров	Ногинск
	Сидоров	Козельск

Рис. 3. Результат выполнения запроса на создание таблицы

На основе таблицы «Авторы» сформируем новую таблицу «Издательства1», в которой исключено ключевое поле, поле «Автор» и дублирование записей. Соответствующий запрос на создание таблицы выглядит следующим образом:

SELECT DISTINCT Издательства.Книга, Издательства.Издательство, Издательства.Тираж INTO Издательства1
FROM Издательства;

В результате выполнения данного запроса сформируется таблица, представленная на рис. 4.

	Книга	Издательство	Тираж
▶	Книга 1	Эльдорадо	10000
	Книга 2	Континент	7000
	Книга 3	Пламя	15000

Рис. 4. Результат выполнения запроса на создание таблицы

Теперь назначим ключевые поля типа «Счетчик» таблицам «Авторы1» и «Издательства1». После назначения ключевого поля таблица «Авторы1» примет вид (рис. 5).

	NA	Фамилия	Город
▶	1	Иванов	Балашиха
	2	Петров	Ногинск
	3	Сидоров	Козельск
*	(Счетчик)		

Рис. 5. Вид таблицы «Авторы1» после назначения ключевого поля

После назначения ключевого поля таблица «Издательства1» примет вид (рис. 6).

	NK	Книга	Издательство	Тираж
▶	1	Книга 1	Эльдорадо	10000
	2	Книга 2	Континент	7000
	3	Книга 3	Пламя	15000
*	(Счетчик)			

Рис. 6. Вид таблицы «Издательства1» после назначения ключевого поля

На следующем шаге необходимо спроектировать таблицу, используемую для организации связей между таблицами «Авторы1» и «Издательства1». Эта таблица состоит из двух числовых полей. Для наглядности полям приписаны имена NA и NK, а самой таблице присвоено имя «Связи».

Далее нужно открыть одну из исходных таблиц, таблицу «Авторы1», таблицу «Издательства1», таблицу «Связи» и расположить их на экране,

чтобы были видны их поля. Такое расположение названных таблиц представлено на рис. 7.

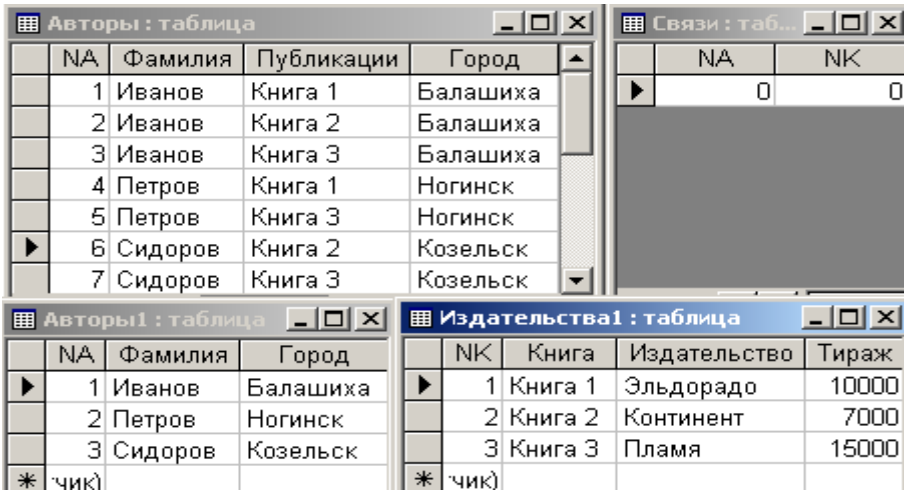


Рис. 7. Расположение таблиц на экране для заполнения таблицы «Связи»

Теперь нужно просматривать записи таблицы «Авторы». Для каждого сочетания полей «Фамилия» и «Публикации» следует выбирать соответствующие значения ключевых полей в таблицах «Авторы1» и «Издательства1» и заносить их в таблицу «Связи». В результате этих действий таблица «Связи» примет вид рис. 8.

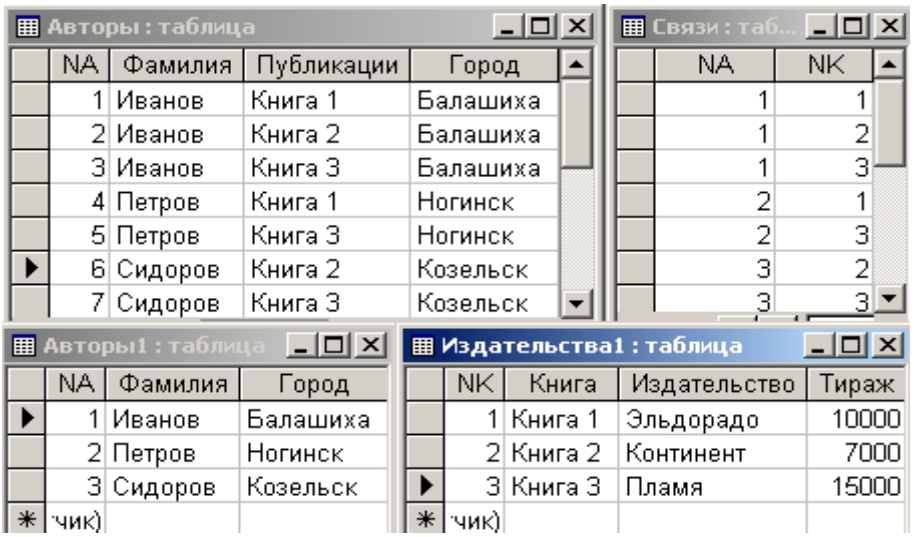


Рис. 8. Заполненная таблица «Связи»

После выполнения перечисленных подготовительных действий можно сформировать связи типа многие - ко многим между таблицами «Авторы1» и «Издательства1» с помощью таблицы «Связи». Соответствующая схема данных представлена на рис. 9.

Рис. 9. Схема данных, реализующая связь многие - ко многим

Теперь убедимся в том, что в случае необходимости мы сможем получить данные в таком же виде, что и в исходных таблицах.

Для формирования данных, содержащихся в таблице «Авторы», сформирован запрос, бланк которого приведен на рис. 10.

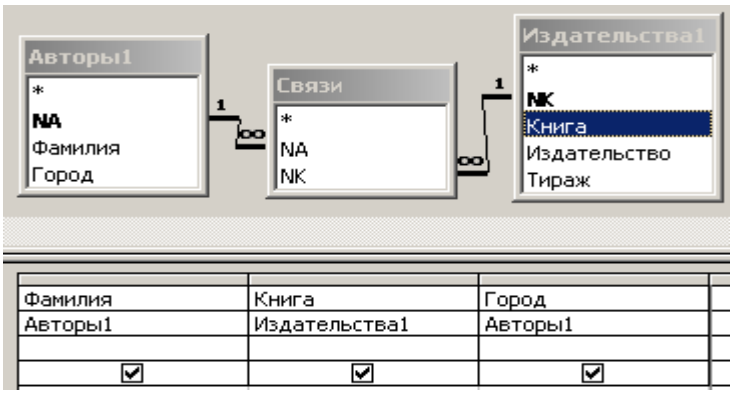


Рис. 10. Бланк запроса для формирования данных, содержащихся в таблице «Авторы»

Результат выполнения данного запроса представлен на рис. 11.

Фамилия	Книга	Город
Иванов	Книга 1	Балашиха
Иванов	Книга 2	Балашиха
Иванов	Книга 3	Балашиха
Петров	Книга 1	Ногинск
Петров	Книга 3	Ногинск
Сидоров	Книга 2	Козельск
Сидоров	Книга 3	Козельск

Рис. 11. Результат выполнения запроса на выборку данных об авторах

Нетрудно заметить, что данные, приведенные на этом рисунке, полностью совпадают с данными исходной таблицы, приведенными на рис.1.

Для формирования, данных содержащихся в таблице «Издательства», сформирован запрос, бланк которого приведен на рис. 12.

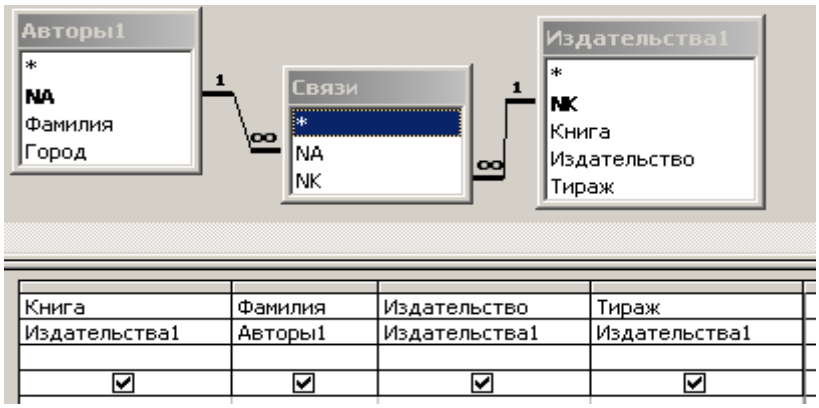


Рис. 12. Бланк запроса для формирования данных содержащихся в таблице «Издательства»

Результат выполнения данного запроса представлен на рис. 13.

Запрос Издательства : запрос на выбор...

Книга	Фамилия	Издательство	Тираж
Книга 1	Петров	Эльдорадо	10000
Книга 1	Иванов	Эльдорадо	10000
Книга 2	Сидоров	Континент	7000
Книга 2	Иванов	Континент	7000
Книга 3	Сидоров	Пламя	15000
Книга 3	Петров	Пламя	15000
Книга 3	Иванов	Пламя	15000

Запись: 1 из 7

Рис. 13. Результат выполнения запроса на выборку данных об издательствах

Нетрудно заметить, что данные, приведенные на этом рисунке, полностью совпадают с данными исходной таблицы, приведенными на рис.2.

Выгода выполненного преобразования таблиц очевидна. После несложных подсчетов можно сделать заключение, что в двух исходных таблицах было заполнено 63 поля, а в трех полученных – 35 полей. Гораздо существенней будет ощущаться выгода преобразования реальных таблиц. Правда, далеко не всегда их удастся преобразовать посредством визуального анализа и использования стандартных средств СУБД.

Литература:

1. Брешенков А.В. Неформальная постановка проблемы преобразования информации табличного вида в файлы баз данных. Сб. трудов АУ МВД России "Актуальные вопросы технологий в деятельности органов внутренних дел". - М.: 2004. – 20 с.

2. Брешенков А.В., Бараков Д.Д. Вопросы преобразования электронных

таблиц в таблицы реляционных баз данных. Современные информационные технологии. Сб. трудов каф. ИУ-6, посвященный 175-летию МГТУ им. Н.Э. Баумана. - М.: Эликс +, 2004.-5 с.

3. Брешенков А.В., Бараков Д.Д. Методика назначения ключевых полей в заполненных реляционных таблицах. Современные информационные технологии. Сб. трудов каф. ИУ-6, посвященный 175-летию МГТУ им. Н.Э. Баумана. - М.: Эликс +, 2005. – 16 с.

4. Гарсиа-Молина Г., Ульман Д., Уидом Д. Системы баз данных. Полный курс: Пер. с англ. - М.: Вильямс, 2004.- 1088 с.

УДК 681.3.07

ПРОБЛЕМЫ ОБЪЕДИНЕНИЯ ТАБЛИЦ В БАЗАХ ДАННЫХ

А.В. Брешенков

В статье проиллюстрировано объединение двух таблиц с данными. Приведены конструкции языка запросов, которые обеспечивают объединение таблиц. Выполнен анализ различных ситуаций возникающих при объединении таблиц.

В работах [1-3] обоснована актуальность проблемы преобразования заполненных нереляционных таблиц в реляционные таблицы, сформулированы задачи преобразования, намечены пути решения отдельных задач. Здесь рассматривается одна из этих задач – задача объединения импортированных таблиц в базах данных.

Если не рассматривать множество специфических особенностей конкретных БД, необходимость объединения содержимого таблиц может возникнуть в двух случаях. В первом случае данные поступают из нескольких источников в центр и там они сводятся в одну таблицу для дальнейшего анализа и обработки. Во втором случае, когда создается БД на основе существующей информации табличного вида, необходимо выявить сходные по структуре и смысловому содержанию таблицы и объединить их в одну.

В соответствии с положениями реляционной алгебры объединение двух отношений есть множество всех кортежей, принадлежащих каждому из

исходных отношений [4]. Другими словами в результате объединения двух таблиц создается третья таблица, которая включает в себя все записи 1-й таблицы и недостающие записи 2-й таблицы. Исходные отношения должны быть совместимы по объединению. Отношения называются совместимыми по объединению, если они базируются на одном и том же числе одних и тех же доменов (столбцов).

В качестве иллюстрации операции объединения используем отношения, представленные таблицами табл. 1, табл. 2 и табл. 3. В табл. 1 и табл. 2 приведены операнды операции отношение, в табл. 3 приведен результат выполнения этой операции.

Таблица 1

ФАМИЛИЯ	ГОД РОЖДЕНИЯ	ГОРОД
Чугунов	**1955**	**Ногинск**
Конев	**1958**	**Козельск**
Деребизова	**1959**	**Моршанск**
Караваев	**1957**	**Семикино**
Попова	**1951**	**Ледово**

Таблица 2

ФАМИЛИЯ	ГОД РОЖДЕНИЯ	ГОРОД
Харченко	**1954**	**Киев**
Умуралиев	**1954**	**Астана**
Комлев	**1958**	**Москва**
Мялицына	**1959**	**Москва**
Попова	**1951**	**Ледово**
Чугунов	**1955**	**Ногинск**

Из анализа табл. 1 и табл. 2 нетрудно заметить, что операнды операции объединения совместимы. Действительно, они состоят из одних и тех же столбцов – заголовки столбцов одинаковые, содержимое одноименных столбцов совпадают по типу. Результаты объединения, как видно из табл. 3., представляют собой все записи 1-й таблицы и недостающие записи 2-й таблицы. Действительно, т.к. первая и последняя запись 1-й таблицы

присутствуют и в 1-й и во 2-й таблице, то в 3-й таблице (результирующей) они встречаются единожды.

Таблица 3

ФАМИЛИЯ	ГОД РОЖДЕНИЯ	ГОРОД
Чугунов	**1955**	**Ногинск**
Конев	**1958**	**Козельск**
Деребизова	**1959**	**Моршанск**
Караваев	**1957**	**Семикино**
Попова	**1951**	**Ледово**
Харченко	**1954**	**Киев**
Умуралиев	**1954**	**Астана**
Комлев	**1958**	**Москва**
Мялицына	**1959**	**Москва**

Смысловое содержание приведенного примера может быть таким. В таблицах – источниках приведены списки участников конференции. Некоторые участники по какой-либо причине зарегистрировались у двух регистраторов. В базе данных необходимо сохранить данные обо всех участниках конференции без дублирования записей. Применение оператора объединения для таблиц двух регистраторов и позволяет получить нужную таблицу. Если участников конференции регистрировало N регистраторов, то оператор объединения отношений необходимо выполнить N -1 раз. Причем в качестве 1-го операнда при первой итерации объединения выступает 1-я таблица, а в качестве 2-го операнда - 2-я таблица. При второй итерации объединения в качестве 1-го операнда выступает результат выполнения предыдущего объединения, а в качестве 2-го операнда - 3-я таблица и т.д.

Запрос, который необходимо выполнить для объединения 2-х таблиц, выглядит следующим образом:

INSERT INTO Список1 (Фамилия, [Год рождения], Город)

SELECT Список2.Фамилия, Список2.[Год рождения], Список2.Город FROM Список2;

Здесь с помощью конструкции «SELECT Список2.Фамилия, Список2.[Год рождения], Список2.Город FROM Список2» из таблицы

«Список2» выбираются значения трех полей. Посредством конструкции «INSERT INTO Список1 (Фамилия, [Год рождения], Город)» значения выбранных полей добавляются в соответствующие столбцы таблицы «Список1». На рис. 1 приведено содержимое таблицы «Список1» после выполнения запроса.

Список1 : таблица

Фамилия	Год рождения	Город
Чугунов	1955	Ногинск
Конев	1958	Козельск
Деребизова	1959	Моршанск
Караваев	1957	Семикино
Попова	1951	Ледово
Харченко	1954	Киев
Умуралиев	1954	Астана
Комлев	1958	Москва
Мялицына	1959	Москва
Попова	1951	Ледово
Чугунов	1955	Ногинск

Рис. 1. Результат выполнения запроса на добавление

Как видно из рисунка, в таблице имеет место дублирование записей. Поэтому для исключения дублирования необходимо выполнить еще один запрос вида:

SELECT DISTINCT Список1.* INTO Список_общий

FROM Список1;

Здесь посредством конструкций «SELECT DISTINCT Список1.*» и «FROM Список1» выбираются все значения полей таблицы «Список1». Посредством режима «DISTINCT» из выбранного списка исключаются повторяющиеся записи, а посредством конструкции «INTO Список_общий» выбранные записи помещаются в новую таблицу «Список_общий». В результате выполнения этого запроса сформируется таблица «Список_общий», которая имеет вид рис. 2.

Как видно из рисунка, нужный результат достигнут – сформирован общий список, а дублирование записей исключено.

В рассмотренном примере, который полностью удовлетворяет условию совместимости по объединению, проблем в процессе формирования сводной таблицы практически не возникает. Это видно из сказанного выше.

Список_общий : таблица

Фамилия	Год рождения	Город
Деребизова	1959	Моршанск
Караваев	1957	Семикино
Комлев	1958	Москва
Конев	1958	Козельск
Мялицына	1959	Москва
Попова	1951	Ледово
Умуралиев	1954	Астана
Харченко	1954	Киев
Чугунов	1955	Ногинск

Рис. 2. Таблица без дублирования записей

В реальных ситуациях дело нередко обстоит несколько сложнее, и возникают проблемы, которые необходимо решать. Рассмотрим эти ситуации в порядке возрастания сложности.

Исходные таблицы по своей природе удовлетворяют требованиям совместимости, а по форме – нет.

Такого рода ситуации возникают когда:

- заголовки одинаковых по смыслу столбцов у объединяемых таблиц отличаются;
- порядок столбцов первой таблицы – операнда не совпадает с порядком столбцов второй таблицы – операнда.

В формате Microsoft Access таблица приведена на рис. 3.

Список0 : таблица

Фамилия	Год рождения	Город
Чугунов	1955	Ногинск
Конев	1958	Козельск
Деребизова	1959	Моршанск
Караваев	1957	Семикино
Попова	1951	Ледово

Рис. 3. Первый операнд операции объединения в формате Microsoft Access

В качестве 2-го операнда используем таблицу, приведенную в формате Microsoft Access на рис. 4.

Список3 : таблица

Откуда прибыл	Участник конференции	Дата рождения
Киев	Харченко	1954
Астана	Умуралиев	1954
Москва	Комлев	1958
Москва	Мялицына	1959
Ледово	Попова	1951
Ногинск	Чугунов	1955

Рис. 4. Второй операнд операции объединения в формате Microsoft Access

Как видно из рис. 4, заголовки 2-го операнда не совпадают с заголовками 1-го операнда. Кроме того, порядок столбцов 2-го операнда не совпадает с порядком столбцов 1-го операнда. Однако визуальный анализ содержимого 1-го и 2-го операндов (обеих таблиц) позволяет сделать вывод о том, что структуры этих таблиц совпадают, и каждому столбцу 1-й таблицы находится соответствующий столбец 2-й таблицы. В связи с этим администратор БД может принять решение о том, в каком порядке содержимое столбцов 2-й таблицы добавлять к содержимому 1-й таблицы. Свое решение администратор может отразить в бланке запроса на добавление. Соответствующий бланк запроса в системе Microsoft Access приведен на рис. 5.

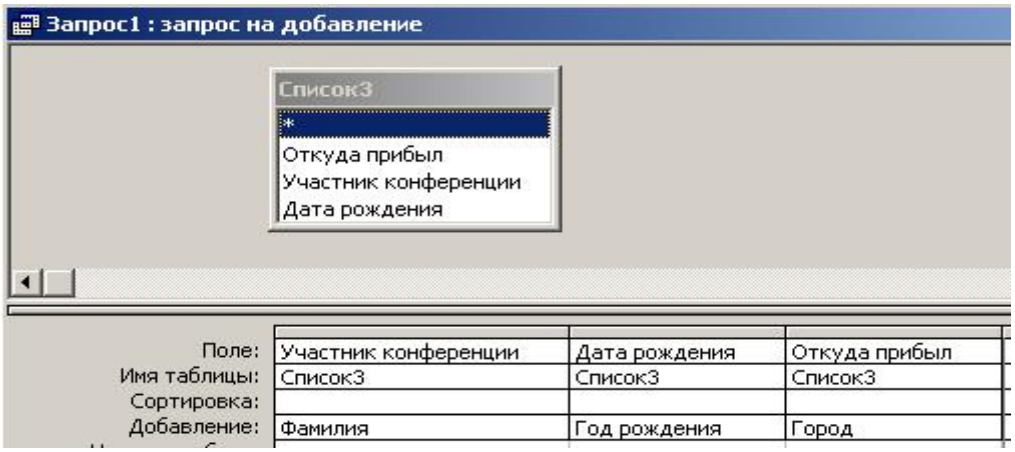


Рис. 5. Бланк запроса на объединение двух таблиц

Таблица, в которую предполагается добавить записи, указывается в меню Запрос/Добавление. В нашем случае указана таблица с именем «Список0». Из рисунка видно, каким образом установлено соответствие между полями таблиц «Список3» и «Список0». Соответствующий запрос в режиме SQL можно просмотреть с помощью меню Вид/Режим SQL. Этот запрос, который может быть использован с незначительными модификациями в любой СУБД, выглядит следующим образом:

INSERT INTO Список0 (Фамилия, [Год рождения], Город)

SELECT Список3.[Участник конференции], Список3.[Дата рождения], Список3.[Откуда прибыл]

FROM Список3;

Из анализа запроса можно сделать заключение о том, каким образом установлено соответствие между полями двух таблиц.

Результат выполнения запроса – это таблица «Список0» с добавленными записями. Она приведена на рис. 6.

Список0 : таблица

Фамилия	Год рождения	Город
Чугунов	1955	Ногинск
Конев	1958	Козельск
Деребизова	1959	Моршанск
Караваев	1957	Семикино
Попова	1951	Ледово
Харченко	1954	Киев
Умуралиев	1954	Астана
Комлев	1958	Москва
Мялицына	1959	Москва
Попова	1951	Ледово
Чугунов	1955	Ногинск

Рис. 6. Результат выполнения запроса на объединение

Как видно из рисунка, результат выполнения запроса ничем не отличается от результата выполнения запроса, приведенного на рис. 1. Таким образом, в результате формирования описанного запроса получен требуемый результат.

Как следует из сказанного выше, рассмотренная проблема несложно решается, если администратор БД способен принять решение о том, каким образом установить соответствие между столбцами объединяемых таблиц. Очень часто такое решение не представляет существенных проблем, т.к. изначально, как правило, предполагается объединение таких таблиц, для которых это имеет смысл и суть столбцов очевидна.

Теоретически можно выявить соответствующие столбцы и автоматически. Но для этого нужно с помощью соответствующих программных средств анализировать семантику содержимого столбцов, что сложно и практически неприемлемо.

Исходные таблицы удовлетворяют требованиям совместимости, результирующую таблицу необходимо обновлять.

Ситуации такого рода возникают в том случае, если в центральной БД аккумулируются данные, поступающие из различных источников. При этом формат БД центра и форматы БД источников заранее оговорены и совпадают. В этом случае проблема совместимости не стоит. Однако возникают проблемы двух типов.

Проблема первого типа связана с тем, что наряду с новыми записями в центр передаются и те записи, которые уже ранее были переданы и добавлены в таблицы центральной БД.

В этом случае, если не принять специальных мер, в центральной БД возможно дублирование записей, что в конечном итоге приводит к искажению информации, противоречивости БД. Передавать же только те данные, которые не передавались ранее затруднительно, а часто и невозможно. Во-первых, механизм отслеживания хронологии импортированных данных из БД регионов нетривиален, а во-вторых, нередко имеется необходимость передачи уже импортированных данных, т.к. эти записи с данными могут быть обновлены, например, изменена стадия выполнения проекта.

Проблема второго типа возникает в связи с тем, что нередко в центр передаются обновленные записи таблиц, которые ранее в центр уже передавались. В связи с этим возникает необходимость обновления всех записей в центре, которые повторно экспортированы из регионов и которые в регионах были изменены. Проблема несколько упрощается в связи с тем, что, как показывает опыт работы с такого рода механизмом передачи и обработки данных, число обновляемых полей невелико и их состав регламентирован.

Исходные таблицы частично удовлетворяют требованиям совместимости.

Ситуация такого рода чаще всего возникает, когда осуществляется проектирование баз данных на основе использования существующей информации табличного вида. Например, возникает необходимость проектирования БД на основе использования набора заполненных электронных таблиц.

Нередко в наборе электронных таблиц можно выявить группы таблиц, в которых значительная часть атрибутов совпадает. Такое положение вещей может быть обусловлено различными причинами. В частности, столбцы могут быть продублированы по ошибке, одинаковые столбцы в разных таблицах используются из соображений удобства визуального анализа данных в рамках одной таблицы, могут быть и другие причины. В этом случае имеет место дублирование данных. Если это иногда приемлемо и даже оправданно в электронных таблицах, то в БД дублирование информации недопустимо. Дублирование БД приводит к противоречивости

БД, ее избыточности. В связи с этим при проектировании БД на основе использования существующей информации табличного вида необходимо решить две проблемы.

Первая проблема связана с выявлением таблиц, в которых значительная часть атрибутов совпадает. В рамках этой проблемы необходимо убедиться не только в том, что имеются совпадающие заголовки таблиц в разных таблицах, но и в том, что эти совпадения неслучайны и суть этих совпадающих заголовков одинакова.

Вторая проблема решается тогда, когда выявлены таблицы с одинаковыми по смыслу атрибутами. Тогда необходимо выявленные таблицы объединить.

В общем случае состав атрибутов даже в таблицах, в которых имеются совпадения атрибутов, может быть различный и тогда задача объединения таблиц усложняется по сравнению с рассмотренными выше примерами.

Литература:

1. Брешенков А.В. Неформальная постановка проблемы преобразования информации табличного вида в файлы баз данных. Сб. трудов АУ МВД России "Актуальные вопросы технологий в деятельности органов внутренних дел". - М.: 2004. – 20 с.

2. Брешенков А.В., Бараков Д.Д. Вопросы преобразования электронных таблиц в таблицы реляционных баз данных. Современные информационные технологии. Сб. трудов каф. ИУ-6, посвященный 175-летию МГТУ им. Н.Э. Баумана. - М.: Эликс +, 2004.-5 с.

3. Брешенков А.В., Бараков Д.Д. Методика назначения ключевых полей в заполненных реляционных таблицах. Современные информационные технологии. Сб. трудов каф. ИУ-6, посвященный 175-летию МГТУ им. Н.Э. Баумана. - М.: Эликс +, 2005. – 16 с.

4. Дейт К., Дж. Введение в системы баз данных. 8-е изд.: Пер. с англ.- М.: Вильямс, 2005. - 1328 с.

УДК 004.056.57

ПРОБЛЕМЫ БЕЗОПАСНОСТИ В БЕСПРОВОДНЫХ СЕТЯХ СТАНДАРТА 802.11

С.Н. Букарева, Б.И. Ващенко

Беспроводные технологии все больше и больше входят в нашу жизнь: с их помощью организуются точки доступа в Интернет, строятся сети, делается множество других вещей. И проблемы защиты и организации безопасности компьютерной информации становятся актуальны, как никогда.

Помимо собственных недостатков беспроводные сети наследуют все уязвимости проводных сетей. Главным их недостатком, как и их преимуществом, является доступность физической среды передачи данных. Если беспроводная сеть не защищена проверенными средствами, у нее нет четко установленных стандартов и политик, которые определяют необходимые методологии развертывания сети и обеспечения безопасности, тогда целостность беспроводной сети находится под угрозой. Хакеры и другие злоумышленники подвергают угрозам сетевые информационные ресурсы, пытаясь получить к ним доступ с помощью специальных атак. Хотя нередки случаи, когда утечка конфиденциальной информации происходит без злого умысла, по нелепой случайности или в результате человеческой ошибки. В данной статье рассмотрим основные виды атак.

Атаки на беспроводные компьютерные сети можно разбить условно на 2 группы:

- Атаки, направленные на получение доступа к беспроводной сети;
- Атаки, направленные на выведение сети из работоспособного состояния [1] .

Существует три основных вида атак направленных на беспроводные сети: атаки отказа в обслуживании (DOS-атаки), атаки «человек посредине» и атаки подмены ARP-записей. Так же взлом WEP-ключей часто рассматривается как атака.

Атаки отказа в обслуживании (DOS-атаки).

Цель любой атаки отказа в обслуживании состоит в создании помехи при доступе пользователя к сетевым ресурсам. Стандартные методы инициирования Dos-атаки заключаются в посылке огромного количества фиктивных пакетов, заполняющих легальный трафик и приводящих к зависанию системы.

Злоумышленник может использовать DOS-атаку, просто чтобы отомстить за безуспешные попытки взломать сеть. Кроме того, DOS-атаки на беспроводные сети могут быть проведены конкурентами, по политическим, экономическим и другим мотивам. К сожалению, из-за природы радиоволн, как носителя информации, и в силу структуры базовых протоколов стандарта 802.11 беспроводные сети невозможно защитить от DOS-атак на первом уровне и от некоторых DOS-атак на втором уровне. Поэтому, каналы 802.11 не стоит использовать для критически важных приложений.

Существует три варианта DoS-атаки на беспроводную сеть:

1. атаки на физическом уровне модели ISO/OSI (требуется специальное оборудование);
2. атаки на канальном уровне и выше (требуется обычный беспроводной адаптер);
3. атаки использующие особенности конкретного оборудования.

В первом случае глушится диапазон Wi-Fi (2,4 ГГц), помимо Wi-Fi сетей ещё глушатся все Bluetooth-устройства в радиусе действия.

Во втором случае, в зависимости от типа шифрования и аутентификации, будут проводиться специальные действия: деаутентификация клиентов или посылка от их MAC адресов ложных пакетов.

В третьем случае используются аппаратные и/или программные уязвимости беспроводных клиентов и/или точки доступа. [2]

Другой проблемой на канальном уровне беспроводных сетей является спуфинг точек доступа. Клиентская часть обычно конфигурируется таким образом, чтобы связываться с точкой доступа с наиболее сильным сигналом. Злоумышленник может просто подделывать SSID точки доступа и клиенты автоматически будут с ней связываться. Таким образом, злоумышленник может захватывать весь трафик и со временем определять WEP-ключи используемые для аутентификации и кодирования трафика в беспроводной сети.

Атака «Человек посредине». Также как и DOS-атаки, атаки «человек посредине» выполняются на беспроводных сетях гораздо проще, чем на проводных. Обычно атаки «человек посредине» имеют две разновидности: подслушивание и манипуляция.

При прослушивании, злоумышленник просто прослушивает набор передач между различными хостами, при этом компьютер злоумышленника не должен быть одной из сторон в соединении. Атаки манипуляции используют возможность прослушивания и нелегального захвата потока данных с целью изменения его содержимого, необходимого для удовлетворения некоторых целей злоумышленника, например для спуфинга IP адресов, изменения MAC адреса для имитирования другого хоста и т.д.

Злоумышленник на своей рабочей станции имитирует узел доступа с более мощным сигналом, чем реальный узел доступа. Клиент беспроводной сети автоматически переключается на новый узел доступа, передавая на него весь свой трафик. В свою очередь, злоумышленник передает этот трафик реальному узлу доступа под видом клиентской рабочей станции. Таким образом, система злоумышленника включается в обмен данными между клиентом и узлом доступа как посредник, что и дало название данному виду. Эта атака опасна тем, что позволяет взламывать защищенные соединения (VPN), устанавливаемые по беспроводной сети, вызывая принудительную реавторизацию VPN-клиента. В результате злоумышленник получает авторизационные данные скомпрометированного им клиента.

Для предотвращения атак прослушивания, необходимо провести шифрование данных на различных уровнях, желательно используя SSH, SSL, или IPSEC. В противном случае большие объемы передаваемой конфиденциальной информации будут попадать к злоумышленникам для дальнейшего анализа.

Атаки подмены ARP-записей. Для понимания механизма действия атак подмены ARP-записей сперва необходимо разобрать в самом ARP. Протокол разрешения адресов позволяет Ethernet объектам, использующим TCP/IP как протокол для передачи данных, различать другие, имеющие IP адреса, объекты в сети. Во многом этот протокол похож на NetBios – он передает по радио трафик на все хосты, в то время как конкретный пакет предназначен только для одного хоста из этой сети. ARP передает запрос для опознания требуемого хоста, имеющего определенный IP адрес. Этот хост принимает сообщение и подтверждает его, а компьютер породивший сигнал сохраняет его MAC адрес в кеше, и при дальнейшей передаче на требуемый хост не будет нужным определение подлинности его IP адреса.

Проблема появляется с приходом новых операционных систем, которые не слишком хорошо придерживаются основной тенденции APR передачи. Если компьютер, управляемый новой версией Windows или даже Linux, обнаруживает пакет, посылаемый конкретной машиной в сети, то он принимает, что MAC адрес этого компьютера правильно сопоставлен с IP адресом компьютера, пославшего этот пакет. Все последующие передачи на этот компьютер будут происходить с использованием действенного, но плохо изученного IP адреса, сохраненного в кеше компьютера для будущих обращений.

Но что, если злоумышленник создаст пакеты с подделанным IP адресом, в которых будет утверждаться, что IP принадлежит MAC адресу его собственного компьютера? Тогда, все данные передаваемые с хостов, использующих сокращенный метод изучения комбинаций MAC/IP адресов, будут приходить на компьютер злоумышленника, а не на предназначенных хост. Это является достаточно серьезной проблемой. Злоумышленник может получить пакеты, просто заменяя в данном локальном кеше комбинации MAC/IP адресов для любых двух хостов, связанных с физической сетью, на которой запущена точка доступа.

Еще один метод получения доступа к закрытой информации – использование специальных программ: вирусы, троянские кони, почтовые черви, снифферы, Rootkit-ы и др. Такие программы предназначены для поиска и передачи своему владельцу секретной информации, либо просто для нанесения вреда системе безопасности и работоспособности атакуемого компьютера. Принципы действия этих программ различны.

Также довольно распространённый вид атаки – сниффинг пакетов, основанный на работе сетевой карты в режиме promiscuous mode, а также monitor mode для сетей Wi-Fi. В таком режиме все пакеты, полученные сетевой картой, пересылаются на обработку специальному приложению, называемым сниффером, для обработки. В результате злоумышленник может получить большое количество служебной информации: кто откуда куда передавал пакеты, через какие адреса эти пакеты проходили. Самой большой опасностью такой атаки является получение самой информации, например логинов и паролей сотрудников, которые можно использовать для незаконного проникновения в систему под видом обычного сотрудника компании.

Инъекция – атака, которая подразумевает внедрение сторонних команд или данных в работающую систему с целью изменения хода работы системы, а в результате – получение доступа к закрытым функциям и информации, либо дестабилизации работы системы в целом. Наиболее популярна такая атака в сети Интернет, но также может быть проведена через командную строку системы.

SQL-инъекция – атака, в ходе которой изменяются параметры SQL-запросов к базе данных. В результате запрос приобретает совершенно иной смысл, и в случае недостаточной фильтрации входных данных способен не только произвести вывод конфиденциальной информации, но и изменить/удалить данные. Очень часто такой вид атаки можно наблюдать на примере сайтов, которые используют параметры командной строки (в данном случае – переменные URL) для построения SQL-запросов к базам данных без соответствующей проверки.

PHP-инъекция – один из способов взлома веб-сайтов, работающих на PHP. Он заключается в том, чтобы внедрить специально сформированный злонамеренный сценарий в код веб-приложения на серверной стороне сайта, что приводит к выполнению произвольных команд.

Межсайтовый скриптинг или XSS (аббр. от англ. Cross Site Scripting) – тип уязвимостей, обычно обнаруживаемых в веб-приложениях, которые позволяют внедрять код злоумышленнику в веб-страницы, просматриваемые другими пользователями. Примерами такого кода являются HTML-код и скрипты, выполняющиеся на стороне клиента, чаще всего JavaScript. Другие названия: CSS, реже – скрипт-инъекция.

XPath-инъекция – вид уязвимостей, который заключается во внедрении XPath-выражений в оригинальный запрос к базе данных XML. Как и при остальных видах инъекций, уязвимость возможна ввиду недостаточной проверки входных данных. [5]

До сих пор считалось, что алгоритм защиты беспроводных сетей WPA не может быть взломан с помощью «грубой силы». Это сумел опровергнуть Томас Рот (Thomas Roth), германский исследователь в области безопасности.

Для взлома паролей сетей, базирующихся на протоколе WPA-PSK, он использовал облачные вычислительные ресурсы, а именно систему Amazon Elastic Cloud Computing (EC2). Целью его исследований было не создание инструмента взлома, а изучение возможностей нового сервиса EC2,

включающего доступ к ресурсам кластера GPU NVIDIA Tesla «Fermi» M2050. Дело в том, что алгоритм подбора паролей очень хорошо распараллеливается, а платформа Tesla прекрасно справляется с подобного рода задачами.

В результате, удалось решить тестовую задачу по подбору целого набора паролей длиной до 6 символов всего за 49 минут. И это при цене одного часа вычислений в $2.10.

Пароли, которые взломал Рот, были очень короткими, и пока нет точной информации сколько времени занял бы взлом более длинных паролей. В своем последнем посте, датированным 12 января 2011 года, Томас Рот пообещал, что представит доклад на грядущей конференции Black Hat, где продемонстрирует взлом WPA-PSK handshake со скоростью ~400.000 PMKs/s.

«Это лишь в очередной раз доказывает, насколько устаревшей является система SHA-1», – пишет Рот.

Национальный институт стандартов и технологий в 2006 году рекомендовал федеральным агентствам прекратить использование SHA-1 и перейти на SHA-2, однако возникло опасение что слишком короткие пароли, шифрованные в SHA-2, будут не менее уязвимы к подбору, чем в SHA-1 [3] .

Описанные выше атаки, ни в коем случае не являются единственными атаками, используемыми хакерами для взлома беспроводных сетей. Постоянно совершенствуются и разрабатываются новые средства защиты информации, но вместе с ними растет и изощренность атак, направленных на их преодоление.

Подводя итог, хочется подчеркнуть, что никакие аппаратные, программные и другие решения не могут обеспечить полную надежность и безопасность информации в компьютерных сетях. В то же время можно свести риск потерь к минимуму. Это возможно лишь при комплексном подходе к вопросам безопасности. Необходимо так же учитывать, оправдывает ли ценность защищаемой информации, средства потраченные на организацию этой защиты.

Литература:

1. Владимиров А.А. Wi-фу: «боевые» приёмы взлома и защиты беспроводных сетей. М.: НТ Пресс, 2005. – 462с.
2. Джонатан Хансен “Атаки на беспроводные сети” http://dewil.ru/security/article/wireless_atack/
3. http://www.xakep.ru/post/53911/default.asp
4. http://www.fssr.ru/hz.php?name=News&file=article&sid=7273
5. http://ru.wikipedia.org/wiki/Хакерская_атака

УДК 004.312

ПРОБЛЕМЫ МЕТАСТАБИЛЬНОСТИ В ЦИФРОВЫХ УСТРОЙСТВАХ

В.С. Буренков, С.Р. Иванов

Метастабильность – явление, которое может привести к нарушению нормальной работы цифровых устройств, возникающее при передаче сигнала между частями устройства, управляемыми несвязанными друг с другом синхросигналами. Если момент изменения входных данных триггера не удовлетворяет требованиям, устанавливаемым временами предустановки и выдержки, то сигнал на выходе триггера может перейти в метастабильное состояние. В данном состоянии значение сигнала колеблется между уровнями логического нуля и единицы в течение некоторого периода времени, что означает, что переход сигнала в единицу либо нуль задерживается на время, большее времени задержки распространения триггера.

В синхронных системах входные сигналы всегда удовлетворяют требованиям, предъявляемым триггерами к временным соотношениям сигналов, следовательно, метастабильность отсутствует. Проблемы, связанные с метастабильностью, обычно возникают в асинхронных системах, где разработчик не может гарантировать соблюдения времен предустановки и выдержки, так как сигнал из передатчика может прийти в любое время относительно синхросигнала приемника.

Для того чтобы минимизировать сбои при асинхронной передаче сигналов, вызванные метастабильностью, возможно использование

последовательности триггеров (синхронизирующей цепочки триггеров, или синхронизатора) в приемнике с тем, чтобы все изменения сигналов в нем приосходили по активному фронту управляющего им синхросигнала.

Синхронизирующая цепочка триггеров, или синхронизатор – это последовательность триггеров, удовлетворяющая следующим требованиям:

- все триггеры в цепи тактируются одним синхросигналом;
- на вход первого триггера цепи поступают данные из передатчика, тактируемого другим синхросигналом (либо эти данные поступают асинхронно);
- выход каждого триггера является входом только одного триггера, за исключением последнего триггера в цепи.

Наиболее часто используются синхронизаторы, состоящие из двух триггеров. Первый триггер такого синхронизатора принимает на вход асинхронный сигнал, за один такт метастабильность сигнала на его выходе будет угасать, затем сигнал с выхода первой ступени поступает на вход второй ступени. Сигнал на выходе второй ступени является стабильным, и его изменение происходит по активному фронту синхросигнала, тактирующего триггеры синхронизатора и в то же время все триггеры приемника.

Основная проблема, связанная с синхронизаторами, заключается в том, что сигнал от передатчика может изменить свое значение дважды до того, как он будет передан в более медленно работающий приемник. В результате приемник не получит информацию, которую он должен был получить.

Передача многоразрядных данных между частями устройства, тактируемыми разными синхросигналами, является примером передачи множества случайным образом изменяющихся сигналов. Использование синхронизаторов при передаче данных, как правило, неприемлемо: слишком много ситуаций, при которых изменение данных приведет к их неправильной передаче.

Существует два распространенных метода синхронизации данных:

- использование сигналов подтверждения для передачи данных между частями устройства;
- использование FIFO-буферов, в которых запись данных в память осуществляется в передатчике, а их чтение – в приемнике.

Основным недостатком первого способа является наличие задержки, требуемой для передачи и распознавания всех сигналов подтверждения для каждого слова передаваемых данных, причем, чем больше сигналов подтверждения используется, тем больше задержка при передаче данных.

Второй способ является наиболее распространенным. В данном случае двухпортовая память используется для хранения данных. Один порт контролируется передатчиком, записывающим данные со скоростью в одно слово за такт синхросигнала записи. Другой порт контролируется приемником, считывающим данные из памяти со скоростью в одно слово за такт синхросигнала чтения. Два управляющих сигнала используются для индикации заполненности или опустошенности буфера.

FIFO-буфер может быть построен различными способами.

Определение того, является ли буфер полным или пустым, требует некоторых математических манипуляций и/или операций сравнения над указателями чтения и записи. Проблема заключается в том, что эти два указателя генерируются в частях устройства, тактируемых разными синхросигналами. Одним из возможных решений этой проблемы является передача одного или обоих указателей в противоположную часть устройства через синхронизатор до того, как можно будет корректно исполнять математические операции и операции сравнения. Лучшим подходом к передаче указателей является использование для обоих указателей счетчиков, работающих в коде Грея. Такие счетчики изменяют свое значение только на один бит за раз, следовательно, если фронт синхросигнала появится в середине перехода состояния счетчика Грея, то будет принято либо старое значение, либо новое, и никаких потерь не возникнет.

Операцию сравнения указателей записи и чтения FIFO можно выполнять асинхронно. В этом случае уменьшается количество используемых синхронизирующих триггеров.

Структура счетчиков Грея также может быть различной. Возможна структура, в которой регистр используется для хранения значений в коде Грея. Сигнал с выхода этого регистра подается на вход преобразователя кода Грея в двоичный код, двоичное значение увеличивается на единицу и передается на преобразователь двоичного кода в код Грея, управляющий входами регистра. Однако данная структура не является оптимальной. Оптимизировать ее можно следующим образом: использовать два регистра –

один в качестве двоичного счетчика, а другой – для хранения значения в коде Грея. При этом двоичный счетчик увеличивает значение, представляемое двоичным кодом, которое передается на вход данного счетчика в качестве следующего двоичного значения и на преобразователь двоичного кода в код Грея, сигнал с выхода которого поступает на вход регистра, хранящего значения в коде Грея. Такие модификации позволяют упростить преобразование из кода Грея в двоичный код, уменьшить количество комбинационной логики и увеличить максимальную частоту работы счетчика.

При проектировании FIFO-буфера необходимо решить еще одну сложную проблему: нужно корректно сгенерировать сигналы, свидетельствующие о заполненности либо опустошенности буфера. Оба этих сигнала генерируются в момент, когда указатели чтения и записи сравниваются. Следовательно, необходимо еще какое-то условие для того, чтобы отличить эти сигналы друг от друга. Получить такое условие можно путем разделения адресного пространства FIFO на четыре квадранта и декодирования двух старших битов обоих счетчиков. Если указатель записи находится на один квадрант позади указателя чтения, то это говорит о том, что буфер, возможно, заполнится. Если указатель записи находится на один квадрант впереди указателя чтения, то это говорит о возможности опустошения буфера.

Проблемы, вызываемые метастабильностью, необходимо учитывать при проектировании современных цифровых устройств на программируемых логических интегральных схемах (ПЛИС) или сверхбольших интегральных схемах с использованием языков описания аппаратуры, таких как Verilog. Если устройство состоит из частей, тактируемых различными несвязанными синхросигналами, то разработчик должен обеспечить правильную передачу данных между этими частями, то есть не допустить потери данных либо приема неверных данных. Для этого, в зависимости от ситуации, ему, как правило, нужно выбирать между использованием синхронизаторов или FIFO-буферов.

FIFO-буфер с асинхронным сравнением указателей записи и чтения и выполнением этих указателей в виде модернизированных счетчиков, работающих в коде Грея, был применен автором во впервые разрабатывающемся блоке сбора и обмена цифровыми данными. В данном

блоке данные с аналого-цифрового преобразователя (АЦП) поступают с одной частотой и затем считываются контроллером прямого доступа к памяти процессора (DMA-контроллером) с другой рабочей частотой. Для обеспечения корректного обмена данными между АЦП и DMA-контроллером в блок входит ПЛИС, в которой был реализован указанный FIFO-буфер.

Литература:
1. Altera. Understanding Metastability in FPGAs, 2009.
2. Clifford E. Cummings. Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs, 2002.
3. Clifford E. Cummings, Peter Alfke. Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons, 2002.
4. Угрюмов Е.П. Цифровая схемотехника. – СПб.: БХВ – Санкт-Петербург, 2000. – 528 с.

УДК 615.47:616-072.7

АЛГОРИТМЫ ОЦЕНКИ СОСТОЯНИЯ ЧЕЛОВЕКА

Е.В. Смирнова

Проблема диагностики утомления (усталости) является актуальной в физиологии труда и спорта [1, 2]. Знание механизма утомления и стадий его развития позволяет правильно оценить функциональное состояние и работоспособность спортсмена, и поэтому утомление должно учитываться при разработке средств и мероприятий, направленных на сохранение здоровья и достижение высоких спортивных результатов.

Утомление – это функциональное состояние организма, вызванное умственной и (или) физической работой, при котором могут наблюдаться временное снижение работоспособности, изменение функций организма и появление субъективного ощущения усталости [3]. На утомление влияет умственная и физическая деятельность. В системе физической культуры и спорта в первую очередь учитывается физическая составляющая утомления,

поэтому величина физической нагрузки непосредственно влияет на физиологическое состояние организма [1].

Дополнительные внешние и внутренние факторы, влияющие на утомление, такие как факторы внешней среды (температура, влажность, давление, газовый состав воздуха), факторы, связанные с нарушением режимов труда и отдыха и с изменением суточных биоритмов, социальные факторы, мотивация и другие при разработке относительно простых и дешевых технических тренировочных средств обычно не учитывают, их считают находящимися в нормальных допустимых пределах. Хотя в более сложных тренировочных системах и методиках эти факторы пытаются различными способами учитывать. Проблема состоит в формализации процесса их учета. Перспективным видится путь взвешенного учета дополнительных факторов на основе теории факторного анализа. Реализация этого направления развития тренировочного процесса требует дополнительной теоретической и прикладной проработки и невозможна без использования компьютеров и компьютерных технологий.

Во время тренировки в зависимости от нагрузки по-разному работают различные системы человеческого организма, идут различные физические и химические процессы. Тренировка под действием нагрузки требует напряженной работы сердечно-сосудистой и дыхательной систем для обеспечения интенсивно работающего организма необходимым количеством кислорода. При такой работе запрос кислорода начинает превышать его потребление и накапливается кислородный долг. Недоокисленные продукты обмена веществ, всасываясь в кровь, ухудшают деятельность нервных клеток. Напряженная деятельность нервных центров осуществляется на фоне кислородной недостаточности, что и приводит к быстрому развитию утомления. Из-за большого расхода энергии в крови уменьшается уровень глюкозы, гормонов некоторых желез внутренней секреции, эритроцитов, гемоглобина. В результате появляются тахикардия, лабильность (неустойчивость) артериального давления, нарушение межцентральных связей в коре головного мозга, ослабление условно-рефлекторных реакций.

Все это может привести к переутомлению – суммарному явлению утомления изо дня в день, когда усталость от предыдущей работы смешивается с усталостью от повторной работы. Переутомление, возникающее в связи с физической тренировкой как результат суммирования

явлений утомления от повторных тренировочных занятий или соревнований, называют еще перетренировкой. Перетренировка не является неизбежным следствием тренировки. Перетренировка наступает только при нарушениях режима тренировки. Соответствующая дозировка нагрузки при тренировочных занятиях и соблюдение необходимых временных интервалов между тренировочными занятиями или соревнованиями, обеспечивающих положенный отдых и ликвидацию явлений острого утомления, исключают возможность перетренировки. Признаками перетренировки в большинстве случаев являются: нежелание заниматься данным видом спорта, потеря своеобразного чувства «мышечной радости», которым обычно сопровождаются занятия физическими упражнениями. Часто отмечаются также общая вялость, уменьшение аппетита, сонливость днем, бессонница ночью, повышенная раздражительность, быстрое наступление усталости при работе и т.д. Эти явления связаны с известным истощением центральной нервной системы и характерны для состояния утомления.

Очевидно, что при тренировке спортсмена необходимо вести контроль за утомлением спортсмена и недопущением утомления. Современная спортивная медицина располагает достаточным количеством способов определения и оценки работоспособности различных органов и систем организма (нервной, дыхательной, сердечно-сосудистой, пищеварительной). Способы оценки включают различные пробы и анализы (например, легочные пробы и анализы крови), измерение пульса и артериального давления, кардиографию, рентгенографию, электромиографию [4, 5]. Все это используется перед тренировкой и после тренировки. Однако мониторинг непосредственно в режиме тренировки осложняет задачу определения состояния спортсмена, поскольку значительную часть медицинских процедур невозможно выполнить во время работы спортсмена в реальном масштабе времени, а сам мониторинг может вызвать у спортсмена чувство дискомфорта из-за наличия средств контроля. Требования к числу средств контроля противоречивы: их число должно быть достаточным для точной диагностики и в то же время минимальным для удобства работы спортсмена.

Возможны, как минимум, три варианта организации систем определения степени утомления:

- с прямой оценкой состояния организма с помощью специальных датчиков, работающих в режиме реального времени;

- с косвенной оценкой состояния, при которой критерием усталости служит не состояние организма человека, а, например, тот факт, что он не может выполнить тренировочное упражнение;

- комбинированная организация, использующая элементы первых двух вариантов.

При прямой оценке состояния обычно измеряются частота пульса и артериальное давление, в частности, может измеряться электрокардиограмма (ЭКГ) и ее параметры. Например, фирмой ООО «Конструкторское бюро информационно-управляющих систем» под руководством профессора Горячева А.В. разработана портативная система измерения кардиологических параметров с автоматической передачей по мобильной связи на сервер и мобильный телефон лечащего врача.

Для комплексной оценки степени утомления может оказаться весьма полезным метод оценки функционального состояния центральной нервной системы (ЦНС) с помощью измерения микроколебания конечностей (МКК) [4]. Для регистрации микротремора применялся сейсмодатчик типа СКГ с пьезоэлектрическим элементом, а запись тремометрии осуществлялась в измерительном комплексе. В исследовании приняло участие 55 спортсменов различной квалификации.

В процессе записи спортсмен сидел в кресле, глаза были закрыты, сейсмодатчик укреплялся на фаланге пальца левой руки, лежащей на мягкой прокладке, кисть свободно свисала. Параллельно с микроколебаниями регистрировалась ЭКГ, по которой определялась длительность кардиоинтервалов. Амплитуда (А) микроколебаний определялась по формуле

$$A = \frac{1}{k}\sum_{i=1}^{n} a_i ,$$

где n – число измерений, a_i – величина i-й амплитуды, а k – коэффициент усиления.

Все полученные значения амплитуд микроколебаний были разделены на семь градаций, начиная с 0,3 мВ, соответствующих состоянию сильного утомления, и заканчивая 7 мВ, соответствующих состоянию сильного возбуждения. В таблице 1 приведены значения амплитуды МКК для наиболее часто встречающихся в спорте состояний: возбуждение, утомление и нормальное состояние. Исследования проводились до тренировки и сразу

после тренировки в течение всего тренировочного периода. Для уточнения степени активации функционального состояния ЦНС проводилось измерение частоты пульса и величины артериального давления, а также проводился опрос спортсменов об их самочувствии.

Приведенные в таблице 1 средние значения МКК можно использовать для оценки состояния спортсмена, определяя степень его утомления. В случае необходимости по предложенной в [4] методике можно снять индивидуальную МКК-грамму и сравнивать ее с реальными показаниями микроколебаний мышц в процессе тренировки, дозируя на основании полученных результатов нагрузку на спортсмена.

Значения микроколебаний конечностей Таблица 1

Степень изменения МКК	**Функциональное состояние ЦНС**	**Диапазон МКК в мВ**	**Среднее значение МКК в мВ**
1	Сильное утомление	0,30 - 0,90	0,60
2	Умеренное утомление	0,90 - 1,40	1,15
3	Легкое утомление	1,40 - 2,00	1,70
4	Нормальное состояние	2,00 - 3,10	2,55
5	Легкое возбуждение	3,10 - 4,10	3,60
6	Умеренное возбуждение	4,10 - 5,20	4,65
7	Сильное возбуждение	5,20 - 9,00	7,10

Результаты измерения колебаний мышц можно использовать не только для определения степени усталости спортсмена, но и для анализа эффективности процесса тренировки. Например, можно исследовать влияние величины нагрузки, длительности тренировки, количества упражнений на такие параметры колебания мышцы, как декремент затухания и условный период затухания [6].

В системах с косвенной оценкой состояния утомления могут быть использованы датчики времени, перемещения, скорости и ускорения, счетчики числа упражнений и длительности занятий. В этом случае вывод о состоянии организма принимается по результатам сравнения показаний этих устройств с эталонными значениями. Так, например, о степени усталости

спортсмена можно судить по времени выполнения упражнения или по числу выполненных упражнений.

Следует отметить, что ни один из рассмотренных подходов не позволяет сформулировать однозначный критерий усталости, пригодный для всех людей. Целесообразно использование комбинированного подхода с применением различных комбинаций элементов перечисленных подходов.

Наличие математических и технических средств оценки степени усталости позволяет изменить подход к построению силовых тренажерных устройств и создать новый класс интеллектуальных тренажерных систем (ИТС) с автоматическим дозированием нагрузки в зависимости от степени усталости. Для повышения точности дозирования в таких системах целесообразно использовать комбинированный подход к построению средств оценки усталости, включив в их состав в минимальной конфигурации подсчет частоты пульса человека и подсчет времени и числа упражнений. Если подсчитанная частота превышает заданный предел, то ИТС вырабатывает сигнал о необходимости прервать выполнение упражнения. Следует, однако, иметь в виду, что, даже зная индивидуальные показатели нормального пульса, которые могут храниться наряду с другими данными о спортсмене в специальной базе ИТС, невозможно однозначно определить, при каком его значении человек устал. И тем более невозможно точно сказать, что человеку не хватает сил, чтобы сделать еще один подход к снаряду. Тем не менее, показания пульса необходимы, если на тренажере занимается человек с очень плохой физической подготовкой, входящий в оздоровительную группу, или человек, имеющий проблемы с сердечно-сосудистой системой. У таких людей целесообразно измерять пульс во время упражнения. Это будет служить страховкой для физически слабых людей, страдающих сердечными заболеваниями.

В ИТС, использующих отягощенное подтягивание, о степени усталости можно судить по времени, затрачиваемому на подтягивание, и по числу подтягиваний (рисунок 1). Если человек не может подтянуться в течение заданного времени (например, пяти секунд), то нагрузку на его мышцы следует уменьшить. Уменьшить ее следует и в том случае, если человек подтягивается с ней меньше заданного числа раз. После каждого срабатывания таймер и счетчик числа подтягиваний сбрасываются и начинают новый отсчет.

Этот способ оценки усталости прост в реализации и управлении и удешевляет ИТС, что имеет важное значение при ее массовом изготовлении. Однако, он может оказаться довольно грубым при идентификации усталости, особенно для физически слабо развитого контингента школьников и студентов.

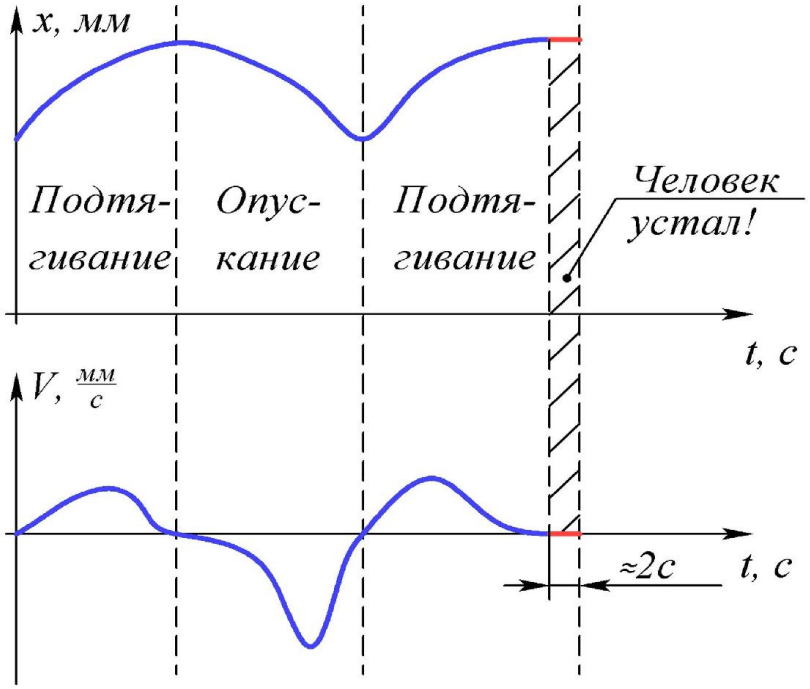


Рис. 1. Графики изменения перемещения и скорости

Для более точного определения усталости предлагается оценку усталости осуществлять путём анализа скорости перемещения плеч человека совместно с информацией о положении этого человека относительно перекладины. В частности, можно уменьшать нагрузку на мышцы человека, если скорость его плеч при движении к перекладине в течение двух секунд примерно равна нулю. Такой критерий обладает однозначностью и позволяет использовать сравнительно несложную систему для определения координаты плеча человека (рисунок 2).

Данная система будет выглядеть следующим образом. Над человеком будет размещена платформа, на которой крепятся два устройства, представляющие собой излучатели и приёмники акустического сигнала. На плечо человека крепится специальный объект, который должен будет отразить сигнал. После того, как отражённый сигнал принят, время движения сигнала от излучателя до объекта наблюдения и обратно делится пополам и умножается на скорость его перемещения в воздухе. В итоге будут получены

два расстояния. По полученным расстояниям строятся две окружности (рисунок 2), нижняя точка пересечения которых даёт информацию о положении объекта наблюдения.

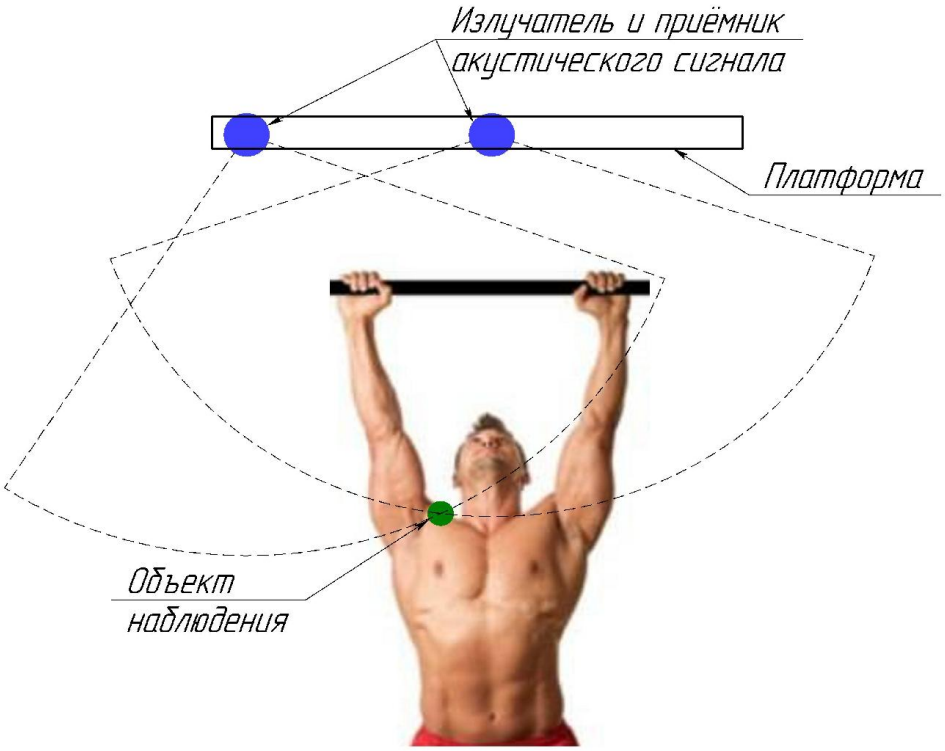


Рисунок 2. Определение координаты перемещения плеча

Кроме системы датчиков для определения положения и скорости перемещения плеч целесообразно также фиксировать, что человек подтянулся до конца. Анализ показывает, что сигнал от этой системы важен не только для фиксирования факта успешного подтягивания и подсчёта их количества. Этот сигнал несёт информацию о том, что до тех пор, пока человек не опустится, менять нагрузку и следить за скоростью перемещения плеч не надо. Как только система определит, что плечи человека начали подниматься, она переключится в режим слежения за скоростью. Работа программы в двух режимах сделает её работу более предсказуемой и стабильной.

Основным достоинством данного способа является гибкость в определении состояния усталости человека. Он естествен для человека. Представьте себе, что Вы стоите и смотрите, как человек подтягивается. Вы наверняка не знаете, какой у этого человека пульс, частота дыхания, не

фиксируете время его подтягивания. Может быть, вы ещё и не всегда замечаете, трясутся ли у человека руки. Вы только видите: человек остановился на полпути и не может подтянуться выше и вам этого уже достаточно, чтобы сказать, что человек устал.

Другим несомненным плюсом этого способа является отсутствие проводов. Всем известно, что современные приёмники FM-радио имеют крошечные габариты и даже встраиваются в сотовые телефоны. Логично ожидать, что, если система определения координаты плеча человека использует в своей работе радиоволны, то приёмники и передатчики радиосигнала будут иметь малые габариты и весить несколько десятков грамм. Такое устройство, присоединив его к зажиму для бумаги, можно легко крепить к вороту человека. Правда, такие устройства приёма и передачи радиоволн необходимо сконструировать (в остальных методах используются уже готовые устройства), а это в итоге может привести к некоторому удорожанию ИТС.

Настоящая статья подготовлена по результатам НИР в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009-2013 годы (государственный контракт №П1264 от 27.08.2009г.).

Литература:

1. Бернштейн Н.А. Физиология движений и активность. М., 1990.

2. Отчет по НИР «Разработка методологии совершенствования учебно-тренировочного процесса учащихся ВУЗов и спортсменов высшей квалификации на основе изучения закономерностей физиологии опорно-двигательного аппарата», этап 1 Государственного Контракта №П1264 от27.08.2010 г., Регистрационный номер РНТД 13244.7701002520.09.1.021.6 от 24.12.2009 г. (Интернет номер документа ТА11221120943)

3. Москатова А.К. Физиологические характеристики и причины развития утомления в различных спортивных упражнениях, М.: «СПРИНТ», 1999 г.

4. Башкин В.М. Исследование изменения функционального состояния нервно-мышечного аппарата спортсменов в течение различных

тренировочных периодов/Научно-теоретический журнал «Ученые записки», № 3(49), СПб, 2009 год
5. Солодков А.С., Сологуб Е.Б. Физиология человека. Общая. Спортивная. Возрастная. // Учебник для высших учебных заведений физической культуры. - М.:Терра-спорт, 2001.
6. Вайн А.А. Явление передачи механического напряжения в скелетной мышце // ТГУ, Тарту, 1990 г., 6

УДК 004.451

ИССЛЕДОВАНИЕ ОСОБЕННОСТЕЙ ВИРТУАЛИЗАЦИИ НА МЕЙНФРЕЙМАХ С ПОМОЩЬЮ СРЕДЫ HERCULES

И.К. Большаков, Е.В. Смирнова

Мейнфрейм – это вычислительная система, изначально ориентированная на бесперебойную работу под исключительно большими смешанными рабочими нагрузками при высоком коэффициенте использования системы. Основным преимуществом серии мейнфреймов IBM над другими системами является длительное время наработки на отказ и практически нулевое время простоя, обусловленное дублированием и горячей заменой компонентов, а также гибким распределение рабочей нагрузки в случае возникновения неисправностей. С начала развития вычислительной техники каждая конкретная модель вычистительной машины была предназначена для использования либо в коммерческой, либо в научной сфере. Историю мейнфреймов принято отсчитывать с появления в 1964 году универсальной компьютерной системы IBM System/360, первой системы, предназначенной для обоих видов вычислений при наличии необходимых программ. Серия System/360 оказала на индустрию неоценимое влияние – именно в рамках проекта по ее созданию были впервые применены многие концептуальные решения, ставшие впоследствии стандартом. Это прежде всего 8-битные байты вместо использовавшихся ранее 4- и 6-битных, 32-разрядная архитектура вычислительной системы, побайтная адресация памяти, сегментация и алгоритм страничной адресации памяти, а также виртуализация памяти.

Перед разработчиками IBM встала задача обеспечения возможности эффективной одновременной работы нескольких пользователей на новом поколении вычислительных систем – мейнфреймах. Одним из способов обеспечения эффективной поддержки многопользовательского режима является создание виртуальной копии оборудования мейнфрейма для каждого пользователя системы. Операционнная система (далее ОС) CP/CMS была разработана группой разработчиков из научного центра IBM в Кембридже в 1972 году и стала первой в мире операционной системой, предназначенной для эмуляции множества операционных систем на мейнфрейме, получив коммерческое название VM/370. Отличительной чертой VM/370 стала отличная производительность и большое количество дополнительных возможностей.

Операционная система VM/370 включает в себя компоненты: управляющая программа, диалоговая система обработки и подсистема удаленной коммуникации и ввода-вывода, соответственно Control Program (далее CP), Console Monitor System(CMS) и Remote Spooling and Communications Subsystem(RSCS). Компонент CP обеспечивает многозадачность и разделение времени между виртуальными операционными системами, а пользователи непосредственно работают с CMS – системой диалоговой обработки, либо вместо нее на некоторые виртуальные машины могут быть установлены OS/360 или ее потомки. Подсистема RSCS обеспечивает передачу информации между виртуальными машинами ОС VM/370. Система VM/370 обеспечивает полную виртуализацию оборудования мейнфрейма, что означает, что операционные системы, которые запускаются в качестве клиентов VM/370 могут использовать набор инструкций, оперативную память, прерывания, исключения и устройства ввода/вывода мейнфрейма не догадываясь о том, что они работают с аппаратной частью не напрямую, а через слой CP. Таким образом, каждый пользователь получает собственную независимую виртуальную копию оборудования мейнфрейма. Полная виртуализация стала возможной благодаря появлению в мейнфреймах серии S/370 поддержки виртуальной памяти для каждой гостевой операционной системы, так как каждая виртуальная машина должна иметь собственную независимое адресное пространство. Когда программа гостевой ОС выполняет системный вызов, он прерывает саму гостевую ОС и выдает команды для ввода/вывода

или другие необходимые команды эмулируемому оборудованию. Эти команды перехватываются слоем CP, который выполняет их на реальном оборудовании.

Для связи между виртуальными машинами операционная система VM/370 использует подсистему RSCS. Подсистема RSCS обеспечивает функционирование виртуальной сети топологии «звезда» с виртуальными машинами на вершинах и управляющей программой CP в качестве центрального узла. Используя уникальный идентификатор для каждой системы, RSCS обеспечивает возможность передачи файлов между виртуальными машинами или между виртуальной машиной и реальным компьютером.

В настоящее время IBM предоставляет свободный доступ к исходному коду VM/370 и поэтому в докладе возможности и особенности VM/370 демонстрируются на персональном компьютере (далее ПК) с помощью эмулятора архитектуры мейнфрейма Hercules. Программа Hercules является проектом с открытым исходным кодом. Она позволяет эмулировать на персональном компьютере архитектуру и внешние устройства всего семейства мейнфреймов IBM – от S/360 до Z-серии. Программа Hercules также эмулирует дисковые устройства IBM 3390 с помощью файлов в файловой системе операционной системы ПК, а терминалы 3270 эмулируются сторонним программным обеспечением, входящим в состав многих дистрибутивов Linux. Для работы с образами магнитных лент и для подготовки образов дисков 3390 используются специальные подпрограммы. Программа Hercules использует протокол TCP/IP для связи с операционной системой ПК.

Эмулируемая на персональном компьютере ОС VM/370 может быть полезна для обучения студентов основам виртуализации на мейнфреймах, поскольку концепции, заложенные в ее основу актуальны до сих пор. Так как нигде не было найдено полной и подробной русскоязычной инструкции по установке VM/370 на ПК, было принято решение поделиться полученным во время экспериментов опытом.

Для демонстрации возможностей системы VM/370 на персональном компьютере должно быть установлено следующее программное обеспечением: ОС Linux, Hercules, эмулятор терминала 3270 x3270, утилиты для работы с образами магнитных лент и дистрибутив VM/370. В данной

работе был использован дистрибутив VM/370 R6, скачанный с сайта www.cbttape.org[5].

Процесс установки начинается с настройки Hercules. Сначала необходимо подготовить конфигурационный файл Hercules, в котором указать конфигурацию эмулируемой системы. В данном случае эмулируется однопроцессорный мейнфрейм S/370 с 2 Мб оперативной памяти. Также в конфигурационном файле указываются адреса виртуальных устройств и расположение эмулирующих их портов или файлов на диске ПК.

Далее необходимо создать пустые образы дисков IBM 3300 емкостью 100 Мб для VM и CP с помощью утилиты dasdinit. После чего следует запустить Hercules, указав в параметрах запуска конфигурационный файл и подключиться к локальному порту 3270 по протоколу telnet. В консоли Hercules при помощи команды ipl (Initial Program Load) необходимо запустить начальную загрузку с образа загрузочной магнитной ленты и начать копирование файлов с образов магнитных лент VMREL6 и CPR6L0 на соответствующие образы жестких дисков.

После завершения процесса система VM/370 готова к запуску. В файле dmkrio.assemble хранится конфигурация устройств ввода/вывода операционной системы VM/370, а в файле dmksys.assemble хранятся ее системные настройки. С их помощью можно конфигурировать систему путем добавления и удаления новых устройств и пользователей.

Для загрузки установленной операционной системы VM/370 необходимо перезапустить Hercules и запустить VM/370 при помощи команды ipl xxxx, где xxxx – номер образа диска с VM. В подключенном терминале telnet требуется ответить на вопросы системы, касающиеся параметров загрузки (режим загрузки, доступные терминалы) и, запустив эмулятор терминала 3270, подключиться к порту 3270. На экране терминала появится приветствие VM/370. Операционная система VM/370 установлена и готова к работе.

Эмулятор Hercules гибко настраивается и может эмулировать большое количество конфигураций мейнфреймов, например количество эмулируемых процессоров может варьироваться от 1 до 8, при этом производительность процессора мейнфрейма, эмулируемого Hercules на процессоре Intel PII по данным М. Кузьминского равна 1-2 MIPS, что вполне достаточно для эмуляции ранних менфреймов. Однако, аналогичная производительность

Intel P4 в 4-5 раз меньше производительности самого младшего процессора мейнфрейма уже предыдущего семейства Z9. Также необходимо отметить многократное преимущество современных мейнфреймов в объеме оперативной памяти, например, мейнфрейм МГТУ им. Баумана серии z10 имеет 64 ГБ оперативной памяти, 12 процессоров, из которых 3 основных процессора предназначены для работы операционных систем серии Z, а остальные для выполнения Java, Linux и вычислений с плавающей запятой. К тому же основным узким местом персональных компьютеров является низкая пропускная способность каналов ввода/вывода данных. Таким образом можно сделать вывод, что эмулятор архитектуры мейнфрейма Hercules пригоден только для решения учебных задач и выполнения унаследованных приложений для устаревших мейнфреймов, не требующих высокого уровня надежности и доступности.

В результате проведенной работы получено русскоязычное руководство по установке VM/370 на эмуляторе Hercules, что может быть полезно для проведения лабораторных работ в ВУЗах, не имеющих доступа к мейнфрейму.

Литература:

1. Кузьминский М. Мейнфрейм на ПК / М. Кузьминский // Открытые системы. – 2006. – N 1.
2. Creasy R.J. The Origin of the VM/370 Time-Sharing System / R.J. Creasy // IBM J. RES. DEVELOP. – 1981. – N 5.
3. Академический центр компетенции IBM. МГТУ. Москва. Web:www.mainframe.bmstu.ru.
4. Hercules – эмулятор мейнфреймов архитектур S/360, S/370, S/390, Z/Architecture. Web:www.hercules-390.org.
5. Сборник дистрибутивов на магнитных лентах. Web:www.cbttape.org.

УДК 004.9

ОСОБЕННОСТИ МОДЕЛЕЙ ПРЕДСТАВЛЕНИЯ ИЗОБРАЖЕНИЙ ДЛЯ НЕЙРОСЕТЕВОГО РАСПОЗНАВАНИЯ СИМВОЛОВ

К.М. Гречищев, Р.С. Самарев

Введение

При распознавании образов независимо от конкретной области применения возникает ряд одинаковых трудностей: зашумленность, искаженность и избыточность входных данных. Существуют различные методы решения этой проблемы, например, использование унитарных преобразований, таких как: преобразование Фурье, Уолша-Адамара, Хоара и другие. Такой подход позволяет не только уменьшить размерность входных данных и уменьшить их избыточность, но и снизить уровень шума за счет фильтрации определенных гармоник.

Для распознавания образов могут быть использованы нейросетевые алгоритмы, поскольку они обладают рядом преимуществ: способностью к обобщению, выявлению признаков по зашумленным образцам, возможностью обучения, а также возможностью параллельной или аппаратной реализации. Кроме того, нейронные сети позволяют выполнять классификацию входных данных.

Данная работа посвящена сравнению эффективности различных моделей представления изображений при использовании нейронных сетей различных конфигураций в контексте распознавания символов.

Выбор моделей представления и проверка их корректности

Для сравнения эффективности распознавания символов было решено использовать четыре модели преставления изображения:

- по пикселям;
- с использованием суммы закрашенных пикселей по каждой строке и столбцу растра;
- с использованием дискретного преобразования Фурье;
- с использованием дискретного преобразования Уолша-Адамара.

Представление изображения по пикселям является самой простой моделью представления. Символ при этом кодируется матрицей m на n, где m и n – размеры изображения. Значение 0 при этом имеют не закрашенные пиксели, а значение 1 – закрашенные. Очевидно, что такое представление

символов не является инвариантным к операциям сдвига, масштабирования и поворота.

При использовании второй модели представления, символ кодируются двумя векторами размерности m и n, которые получаются путем суммирования элементов матрицы изображения по строкам и столбцам соответственно. Ясно, что такой способ позволяет снизить размерность изображения по сравнению с первой моделью с $m \cdot n$ до $m+n$. Кроме того, выполнив сдвиг элементов вектора, можно легко обеспечить инвариантность к операции сдвига, а масштабирование вектора может привести и к инвариантности относительно операции изменения масштаба.

Подвергнув матрицу изображения дискретному двумерному преобразованию Фурье (ДПФ), получаем третью модель представления изображения. В общем случае, преобразование двумерное ДПФ дает матрицу $F = \{ f_{i,j} \}$ комплексных чисел размером m на n. Однако в силу того, что входная матрица состоит только из действительных чисел, то $f_{i,j}$ будет комплексно сопряженным с $f_{i,\ n-j}$ для $\forall\ i = \overline{1\ ..\ m}$, поэтому в результате имеем матрицу размером m на n/2+1. Переходим к представлению символа в виде амплитуды и фазы. Поскольку фаза не несет информации о форме символа, а лишь о его положении, ее можно отбросить, снизив размерность и получив инвариантность относительно сдвига. Получаем представление символа в виде матрицы m на n/2+1 действительных чисел.

Если вместо преобразования Фурье использовать преобразование Уолша-Адамара, получим матрицу размером n на n действительных чисел, которая является четвертой моделью преставления изображения. Достоинством этого преобразования является высокая скорость его выполнения.

Данные на вход нейронной сети подаются в виде вектора. Во всех рассмотренных моделях представления, кроме представления суммами, после выполнения преобразований получаем матрицу. Для получения вектора предлагается построчно записать матрицу в одномерный массив. В результате, из матрицы размером m на n получим вектор длиной $m \cdot n$. При представлении суммами, входной вектор имеет длину m+n и включает в себя сначала вектор построчных сумм, а затем вектор сумм по столбцам.

До анализа пригодности использования для распознавания, все четыре модели исследовались с помощью самоорганизующейся карты Кохонена с размерностью выходного слоя 30 на 30 нейронов. Исследование проводилось независимо для каждой модели представления. Обучающее множество содержало 20 образцов, по одному для каждого символа. Размерность изображения равнялась 32 на 32 пикселя. В процессе организации карты визуализировалось как расстояние между соседними нейронами, так и символ, на который у нейрона был максимальный отклик. Использование различных моделей представления приводит к различной размерности преобразованного изображения, а значит и различному числу входных нейронов сети. Чем больше число входов сети, тем больше времени требуется карте Кохонена для классификации символов:

Модель представления	Количество входов сети	Время классификации, с
По пикселям	256	652
Суммирование	32	372
Фурье	144	514
Уолша-Адамара	256	776

Для всех четырех моделей карта выделяла кластеры символов по внешнему виду, причем кластеры схожих по форме букв располагались ближе друг к другу. Для всех моделей представления число кластеров было одинаковым и равнялось числу символов в обучающем множестве, то есть 20.

Эффективность распознавания символов в различных моделях многослойным персептроном

Опыты по распознаванию символов производились над алфавитом, состоящим из 20-х первых букв русского алфавита. Из рассмотрения исключались буквы «й» и «ё». Для распознавания использовалась сеть из 3 слоев, включая входной. Каждый нейрон i-го слоя был соединен с каждым нейроном слоя i+1. Количество нейронов во входном слое варьировалось в зависимости от используемой модели представления. Количество нейронов в

скрытом и выходном слое было фиксированным и равнялось числу букв алфавита.

Для обучения сети использовались представленные в соответствующей модели изображения символов, не содержащие шума. После обучения сеть подвергалась тестированию. Для этого было сформировано по 1000 искаженных шумом образов каждого символа и для каждой степени искажений, которые поочередно подавались на вход сети. Для формирования шума использовался генератор случайных чисел, формирующий значения по равномерному закону распределения вероятности. Качество распознавания оценивалось по трем критериям: числу верно распознанных, не распознанных и ошибочно распознанных символов. Признаком того, что символ распознать не удалось, являлось наличие в выходном слое нейрона, активность которого отличалась от активности нейрона с максимальным выходом менее чем на 25% от разницы между максимальным и минимальным значением активности. Результаты тестирования представим в виде графиков (рисунок 1, 2).

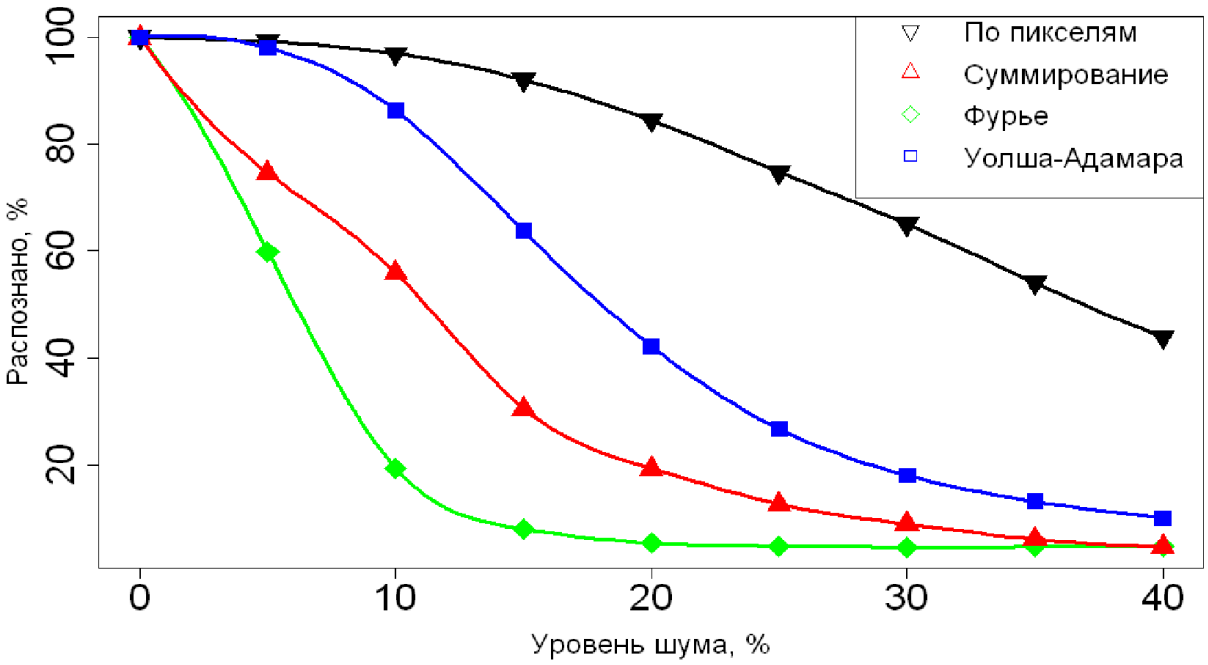


Рис. 1. Процент верно распознанных символов при отсутствии фильтрации

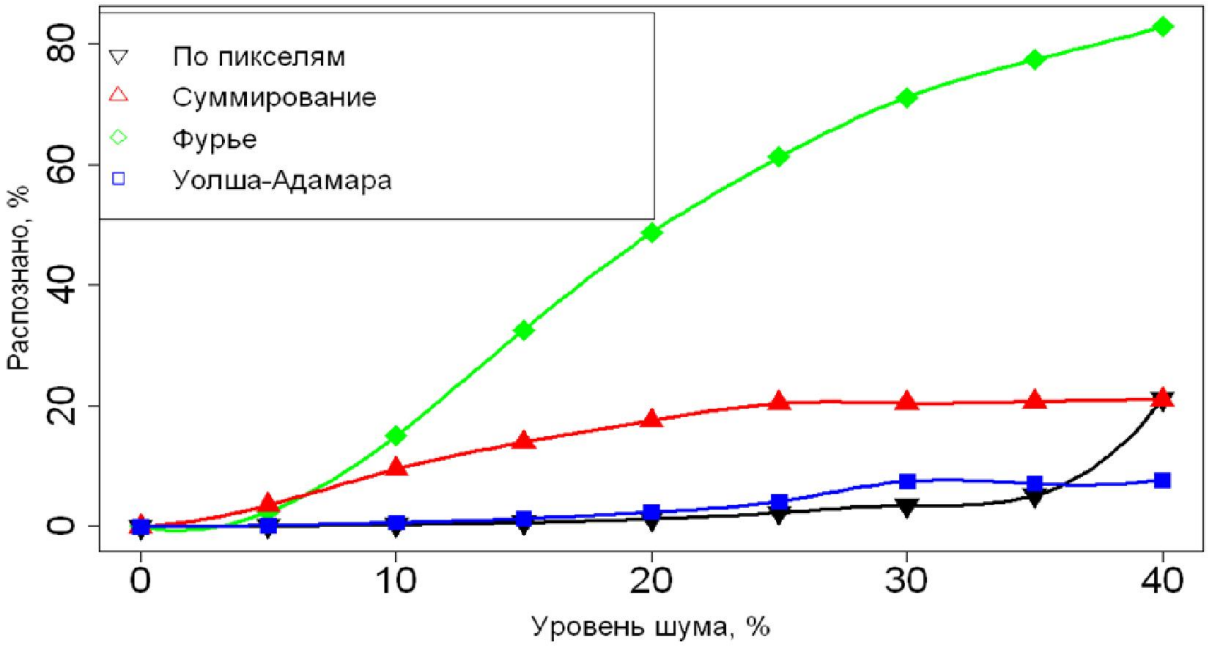


Рис. 2. Процент ошибочно распознанных символов при отсутствии фильтрации

При данных условиях лучшее качество распознавания достигается при использовании модели представления по пикселям, а худшее – при представлении спектром Фурье.

Повышение качества распознавания путем медианной фильтрации

Качество распознавания можно значительно улучшить, если использовать какой-либо фильтр. В данной работе использовался медианный фильтр. Окно этого фильтра имело размер 3 на 3 пикселя и последовательно двигалось по изображению, при этом значение на выходе фильтра равнялось медиане значений внутри окна. При этом для оценки качества распознавания использовались те же 3 критерия, что и в предыдущем случае. Результаты в виде графиков представлены на рисунках 3 и 4.

При применении фильтра, лучшие результаты достигаются при использовании спектральных моделей представления символов (Уолша-Адамара и Фурье), особенно при уровнях шума меньше 25%.

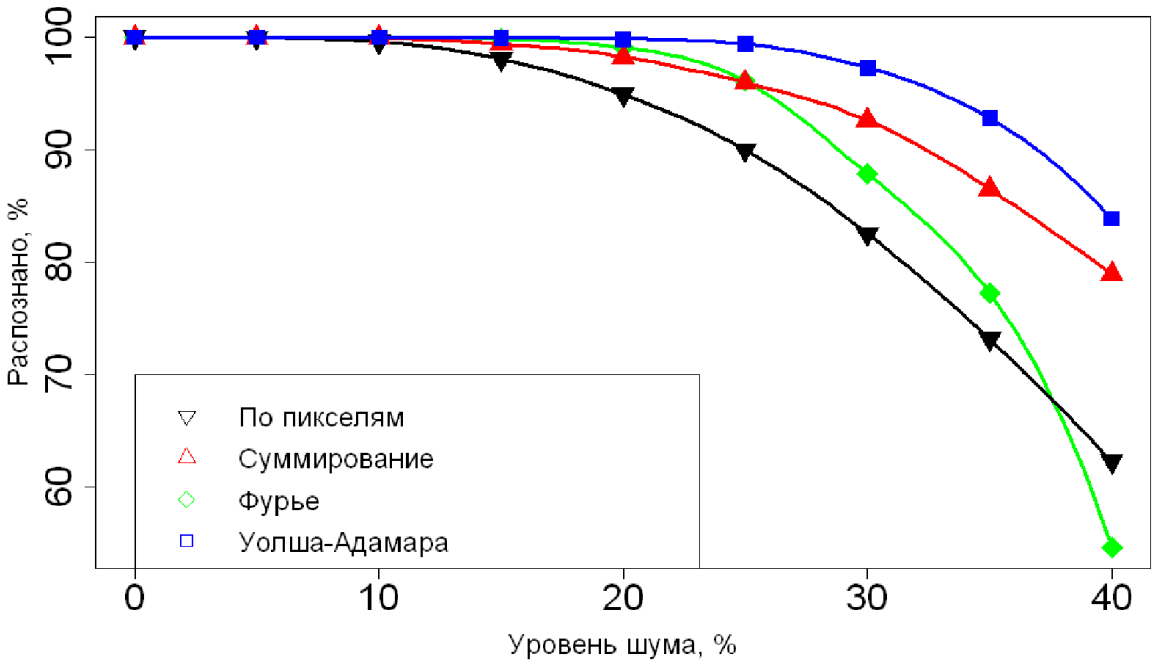


Рис. 3. Процент верно распознанных символов при использовании медианной фильтрации

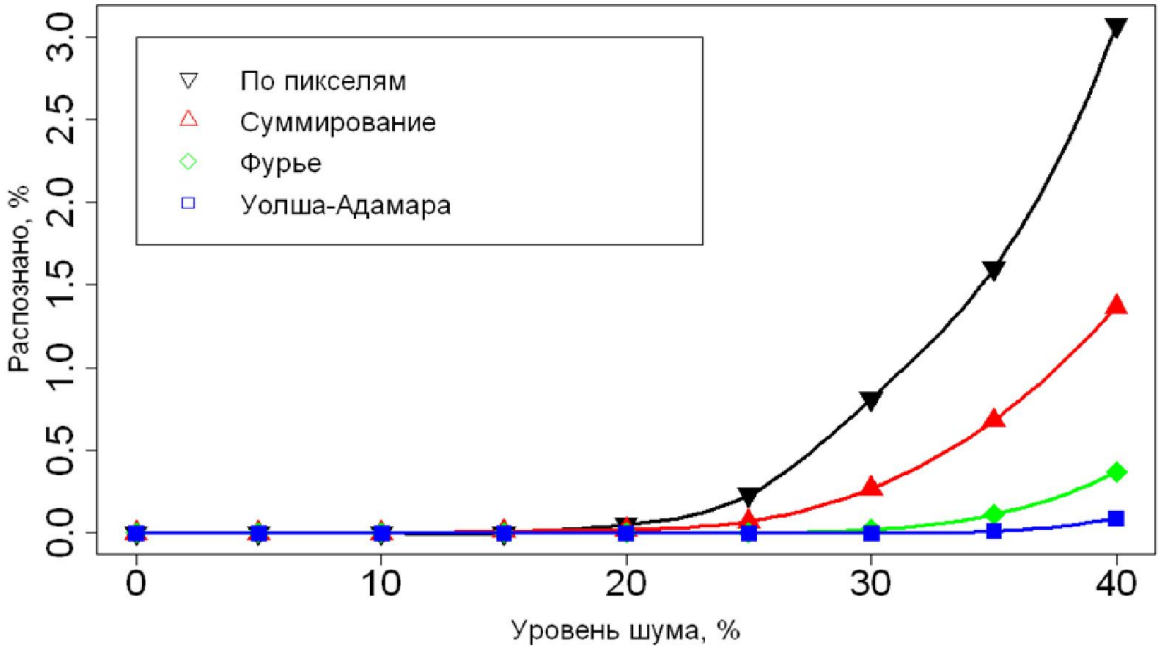


Рис. 4. Процент ошибочно распознанных символов при использовании медианной фильтрации

Получение свойств инвариантности к операции сдвига

Применение некоторых моделей представления позволяет получить инвариантность к операции сдвига. Так, при применении модели представления спектром Фурье, она достигается путем отбрасывания спектра фаз. При применении модели представления суммами предлагается найти

положение «центра масс» вектора с суммами и фиксировать его. Положение «центра масс» можно вычислить следующим образом:

$$C = \frac{\sum_{i=1}^{n} P_i \cdot i}{\sum_{i=1}^{n} P_i},$$

где P_i - элемент вектора суммами с номером i.

Однако такой подход не применим для циклического сдвига, поскольку в этом случае изменяется положение «центра масс» относительно вектора. На практике циклически сдвинутые символы не встречаются, поэтому данное ограничение не является существенным с точки зрения решаемой задачи.

Выводы

В работе были рассмотрены четыре модели представления изображений символов для нейросетевого распознавания. Все они оказались применимы для распознавания символов, как показало исследование на карте Кохонена и результаты распознавания многослойным персептроном. Кроме того, для двух из рассмотренных моделей удалось обеспечить инвариантность представления к операции сдвига.

Удалось значительно улучшить качество распознавания, применив медианную фильтрацию. В этом случае лучше качество распознавания достигается при представлении символа спектром Уолша-Адамара. При отсутствии фильтрации лучше всего распознаются символы, представленные по-пиксельной моделью.

Следует отметить, что при проведении обучения и тестирования использовались символы одного шрифта и начертания. Однако границы символов искажались при проведении фильтрации, поэтому логично предположить, что, при попытке обобщить результаты работы на различные шрифты, использование спектральных преобразований даст хорошие результаты. Кроме того, представление изображения в спектральной области теоретически дает возможность еще большего понижения размерности за счет фильтрации определенных гармоник и использования свойств спектра.

Литература:

1. Хайкин С. Нейронные сети: полный курс, 2ое издание: Пер. с англ.– М.: Издательский дом «Вильямс», 2006.– 1104 с.

2. Уоссермен Ф. Нейрокомпьютерная техника: Пер. с англ.– М.: Мир, 1992.– 184 с.
3. Каллан Р. Основные концепции нейронных сетей: Пер. с англ.– М.: Издательский дом «Вильямс», 2001.– 287 с.
4. Залманзон Л.А. Преобразования Фурье, Уолша и Харра и их применение в управлении, связи и других областях. - М.: Наука, 1989.– 496 с.
5. Претт У. Цифровая обработка изображений: Пер. с англ.– М.: Мир, 1982.– Кн.1 - 312 с.
6. Бутаков Е.А., Островский Е.И., Фадеев И.Л. Обработка изображений на ЭВМ/Е. – М.: Радио и связь, 1987.– 240 с.
7. Ярославский, Л.П. Введение в цифровую обработку изображений / Л.П. Ярославский. – М.: Советское радио, 1979.

УДК 681.324

РЕАЛИЗАЦИЯ АЛГОРИТМА УМНОЖЕНИЯ МАТРИЦ БОЛЬШОЙ РАЗМЕРНОСТИ НА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ ТИПА «ЦИРКУЛЯНТ»

В.А. Григорьев, Ю.М. Руденко

Матрицы и матричные операции широко используются при математическом моделировании самых разнообразных процессов, явлений и систем. Матричные вычисления составляют основу многих научных и инженерных расчетов, например это вычисления в области вычислительной математики, физики, экономики и др. При выполнении умножения матриц, число столбцов в одной матрице должно совпадать с числом строк во второй. Если имеются матрицы большой размерности, то стандартный алгоритм для умножения может стать весьма затратной процедурой в вычислительном отношении: сначала каждая строка одной матрицы поэлементно умножается на каждый столбец второй, а затем вычисляется сумма поэлементных произведений.

Таким образом, для многих методов матричных вычислений характерным является повторение одних и тех же вычислительных действий для разных элементов матриц. Данное свойство свидетельствует о

наличии параллелизма по данным при выполнении матричных расчетов, и, как результат, распараллеливание матричных операций сводится в большинстве случаев к разделению обрабатываемых матриц между процессорами используемой вычислительной системы. Являясь вычислительно трудоемкими, матричные вычисления представляют собой классическую область применения параллельных вычислений. Использование высокопроизводительных многопроцессорных систем позволяет существенно повысить сложность решаемых задач. Выбор способа разделения матриц определяет конкретный метод параллельных вычислений. Разные схемы распределения данных приводят к целому ряду параллельных алгоритмов матричных вычислений.

Рассмотрим ленточное разбиение матрицы, которое является одним из наиболее эффективных способов разделения матриц.

В общем виде умножение матрицы A размера $m \times n$ и матрицы B размера $n \times l$ приводит к получению матрицы C размера $m \times l$, каждый элемент которой определяется в соответствии с выражением:

$$c_{ij} = \sum_{k=0}^{n-1} a_{ik} \cdot b_{kj},\ 0 \le i < m,\ 0 \le j < l$$

Этот алгоритм предполагает выполнение $m \cdot n \cdot l$ операций умножения и столько же операций сложения элементов исходных матриц. При умножении квадратных матриц размера $n \times n$ количество выполненных операций имеет порядок $O(n^3)$.

При параллельном алгоритме умножения матриц, матрицы A и B разбиваются на непрерывные последовательности строк.

Алгоритм представляет собой итерационную процедуру, количество итераций которой совпадает с числом подзадач. На каждой итерации алгоритма каждая подзадача содержит по одной строке матрицы A и одному столбцу матрицы B. При выполнении итерации проводится скалярное умножение содержащихся в подзадачах строк и столбцов, что приводит к получению соответствующих элементов результирующей матрицы C. По завершении вычислений в конце каждой итерации столбцы матрицы B должны быть переданы между подзадачами с тем, чтобы в каждой подзадаче оказались новые столбцы матрицы B и могли быть вычислены новые элементы матрицы C. При этом данная передача столбцов между

подзадачами должна быть организована таким образом, чтобы после завершения итераций алгоритма в каждой подзадаче последовательно оказались все столбцы матрицы B.

Достаточно простая схема организации необходимой последовательности передач столбцов матрицы B между подзадачами представлена топологией информационных связей подзадач в виде кольцевой структуры. В этом случае на каждой итерации подзадача i, $0 \le i < n$, будет передавать свой столбец матрицы B подзадаче с номером $i+1$ (в соответствии с кольцевой структурой подзадача $n-1$ передает свои данные подзадаче с номером 0). После выполнения всех итераций алгоритма необходимое условие будет обеспечено – в каждой подзадаче поочередно окажутся все столбцы матрицы B.

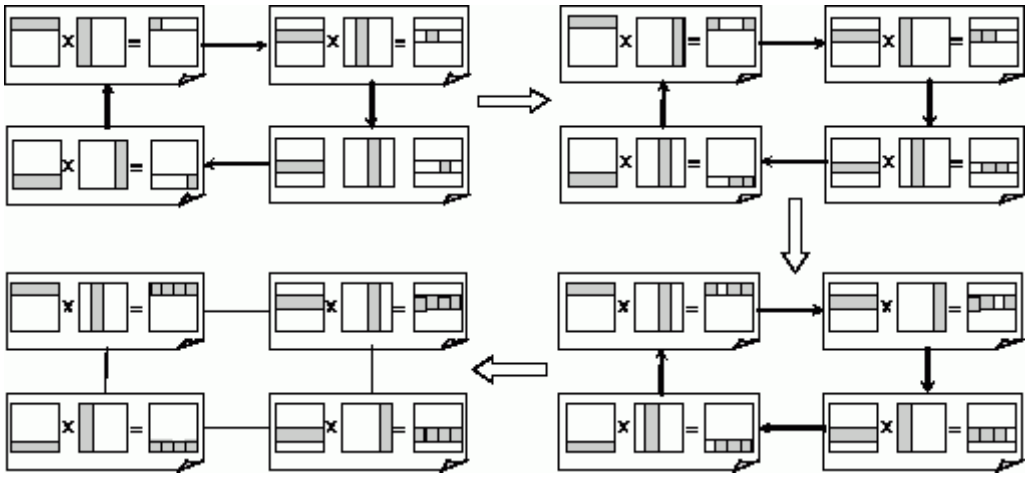


Рис. 1. Общая схема передачи данных для параллельного алгоритма матричного умножения при ленточной схеме разделения данных

На рисунке 1 представлены итерации алгоритма матричного умножения для случая, когда матрицы состоят из четырех строк и четырех столбцов (n=4). В начале вычислений в каждой подзадаче i, $0 \le i < n$, располагаются i-я строка матрицы A и i-й столбец матрицы B. В результате их перемножения подзадача получает элемент c_{ii} результирующей матрицы C. Далее, подзадачи осуществляют обмен столбцами, в ходе которого каждая подзадача передает свой столбец матрицы B следующей подзадаче в соответствии с кольцевой структурой информационных взаимодействий. Далее выполнение описанных действий повторяется до завершения всех итераций параллельного алгоритма.

Выделенные базовые подзадачи характеризуются одинаковой вычислительной трудоемкостью и равным объемом передаваемых данных. Когда размер матриц n оказывается больше, чем число процессоров p, базовые подзадачи можно укрупнить, объединив в рамках одной подзадачи несколько соседних строк и столбцов перемножаемых матриц. В этом случае исходная матрица A разбивается на ряд горизонтальных полос, а матрица B представляется в виде набора вертикальных полос. Размер полос при этом следует выбрать равным $k=n/p$ (в предположении, что n кратно p), что позволит по-прежнему обеспечить равномерность распределения вычислительной нагрузки по процессорам, составляющим многопроцессорную вычислительную систему.

При распределении подзадач между процессорами может быть использован любой способ, обеспечивающий эффективное представление кольцевой структуры информационного взаимодействия подзадач описываемый Dn-графом. Для этого достаточно, например, чтобы подзадачи, являющиеся соседними в кольцевой топологии, располагались на процессорах, между которыми имеются прямые линии передачи данных.

Построение плана распределения операторов по вычислительным модулям (ВМ) вычислительной системы с распределенной памятью проводится поэтапно. На начальном этапе необходимо составить граф-схему.

Затем на основе граф-схемы или алгоритма строится диаграмма нитей. Которая используется для представления матрицы следования и определяет структуру ВС.

Рассмотрим распределение операторов по ВМ вычислительной системы. Пусть размерность матриц составляет 100x100. При ленточном разделении данных построение нитей осуществляется аналогично как для ВС с общей памятью: вычисляются ранние сроки окончания выполнения операторов, затем строится их диаграмма (рисунок 2).

На основании диаграммы видно, что один ВМ передает информацию (строку матрицы) остальным. Передача информации осуществляется транзитным способом через другие ВМ. Затем на каждом ВМ вычисляется элемент результирующей матрицы и сохраняется. Время передачи строки от одного ВМ всем остальным меньше времени вычисления элемента результирующей матрицы. Это действие повторяется для всех строк полосы передающей матрицы. В данном примере 5 полос по 20 строк и 20 столбцов

соответственно. На рисунке 2. представлен алгоритм умножения полосы передающей и принимающий матриц.

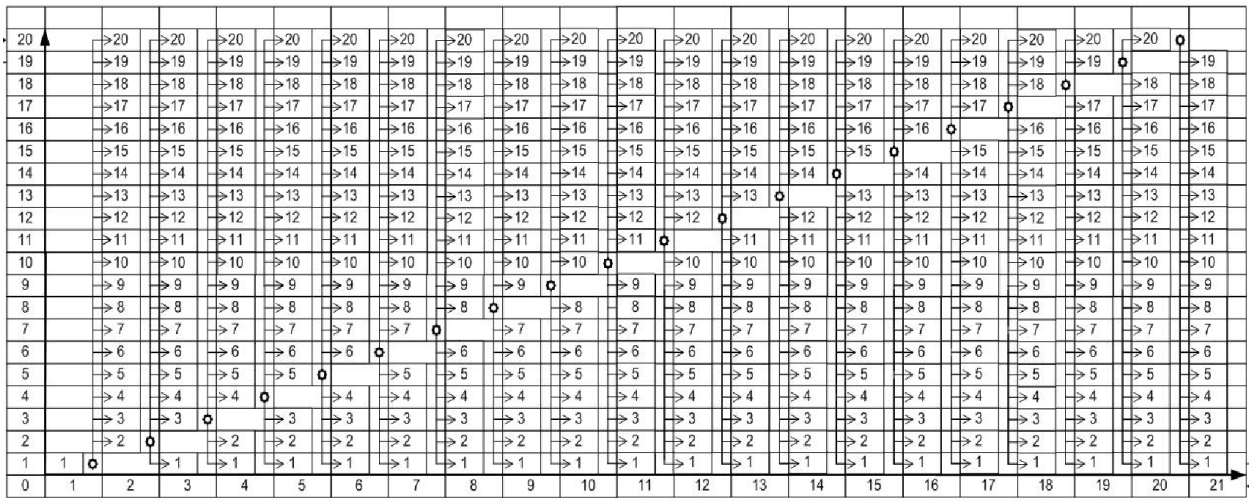


Рис. 2. Диаграмма ранних сроков окончания выполнения операторов алгоритма умножения матриц при ленточном разделении данных

Для реализации диаграммы представления нитей необходимо 20 процессоров (ВМ), диаграмма получается достаточно компактной. На каждом из 20 процессоров выполняется порядка 19 нитей и между ними происходит обмен данными. Затем строится матрица следования и определяется структура ВС.

Решение представленной задачи выполняется на структуре ВС типа циркулянты (20, 1, 3, 7) (рисунок 3).

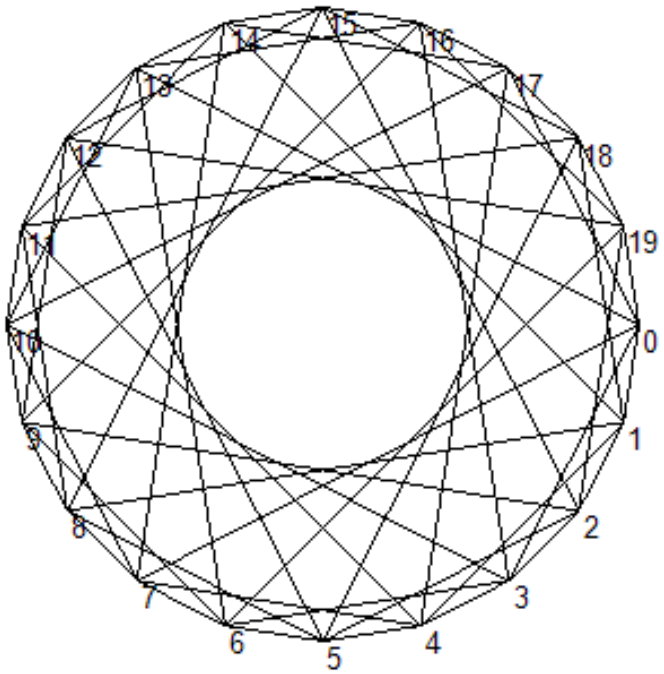


Рис. 3. Загрузка циркулянты (20, 1, 3, 7) нитями диаграммы

Рассмотрим процесс получения элемента результирующий матриц (ЭРМ). На рисунке 3 видно, что при вычислении ЭРМ, ВМ 0 передает свою строку матрицы А ВМ под номерами 1, 3, 7, 13, 17, 19. Затем, каждый ВМ, получивший строку матрицы А, передаёт её другим ВМ под номерами 2, 4, 6, 8, 10, 12, 14, 16, 18. На последнем шаге передачи, информацию принимаю ВМ 5, 9 и соответственно 11 и 15. Затем на каждом процессоре вычисляется ЭРМ и сохраняется в память. Из описания одного цикла вычисления ЭРМ следует, что время транзитной передачи строки матрицы А существенно зависит от параметров циркулянта.

Таким образом, при перемножении матриц большой размерности, когда число ВМ много меньше размерности матриц наиболее эффективной является структура ВС представленная Dn-графом, в частности «циркулянтом».

Литература:

1. Руденко Ю.М., Волкова Е.А. Вычислительные системы.-М.: НИИ РЛ МГТУ им. Н.Э.Баумана, 2010.-212 с.

2. Воеводин В.В., Воеводин Вл.В Параллельные вычисления**.-**СПб.: БХВ-Петербург, 2002.-608 с.

УДК 61:001.89

ПРОГРАММИРОВАНИЕ КОМПЛЕКСНОЙ БИОМЕХАНИЧЕСКОЙ СИСТЕМЫ «ЧЕЛОВЕК НА ШАГАЮЩЕМ ТРЕНАЖЕРЕ»

Д.М. Жук

Способность с помощью пакета Life Modeler моделировать человеко-механические системы позволяет исследовать методом моделирования различные проблемы взаимодействия биологических и механических компонентов таких систем [1-3]. Функциональные возможности такого подхода к разработке биомеханических систем в данной статье иллюстрируются на примере моделирования тренировочного процесса с использованием эллиптического (шагающего) тренажера. Этот пример отражает влияние движения шагом в году, моделируемого моментом сопротивления тренажера, на значение силы мышц и хронометрию процесса

ходьбы. Результаты моделирования будут полезны для исследований, связанных с реакцией тела человека на тренировочный или реабилитационный процесс ходьбы с переменной нагрузкой.

Как и обычно, при использовании системы Life Modeler, в этом случае необходимо выполнить следующие этапы [3]:

- импорт модели тела человека;
- создание мягких тканей;
- объединение модели человека с моделью тренажера;
- добавление отчетов движения в модель;
- моделирование равновесного состояния;
- решение обратной задачи динамики;
- подготовка модели к решению прямой задачи динамики;
- решение прямой задачи динамики;
- выполнение параметрического анализа;
- анализ результатов.

Импорт модели тела человека

На этом этапе для создания модели человеческого тела используется SLF файл из библиотеки моделей, содержащий антропометрические размеры, параметры суставов, параметры осанки и параметры зафиксированных движений (MOCAP данные). Используя параметры, содержащиеся в SLF файле, создаются сегменты тела.

Этот файл содержит информацию об имени субъекта, поле, возрасте, росте и весе. LifeMOD™ использует эту информацию для получения параметров сегментов тела, а масс-инерционные параметры сегментов выбираются из внутренней антропометрической базы данных.

Пассивные суставы создаются для решения обратной задачи динамики в процессе моделирования. Для этой модели пассивные суставы будут созданы для моделирования обратной задачи динамики. Пассивные суставы состоят из трехосевого шарнира (3 DOF), который включает угловые ограничители вращения, моменты жесткости и демпфирования. Этот тип сустава используется прежде всего, чтобы стабилизировать тело в процессе моделирования обратной задачи динамики. Параметры включены в SLF файл.

Шаг 1: Открыть панель импорта

Запустить программу LifeMOD ™ и выбрать CREATE NEW MODEL, чтобы начать новую сессию моделирования. Выбрать XCHANGE в главном меню и IMPORT SLF MODEL FILE в подменю.

Шаг 2: Импортировать модель тела человека

Выбрать Model Library и Full Body Elliptical Exercise Machine как SLF файл библиотеки моделей. Внизу панели должны быть выбраны Body, Joints и Posture. Выбрать APPLY для создания сегментов тела, суставов и осанки. Результирующая модель приведена на рисунке 1.

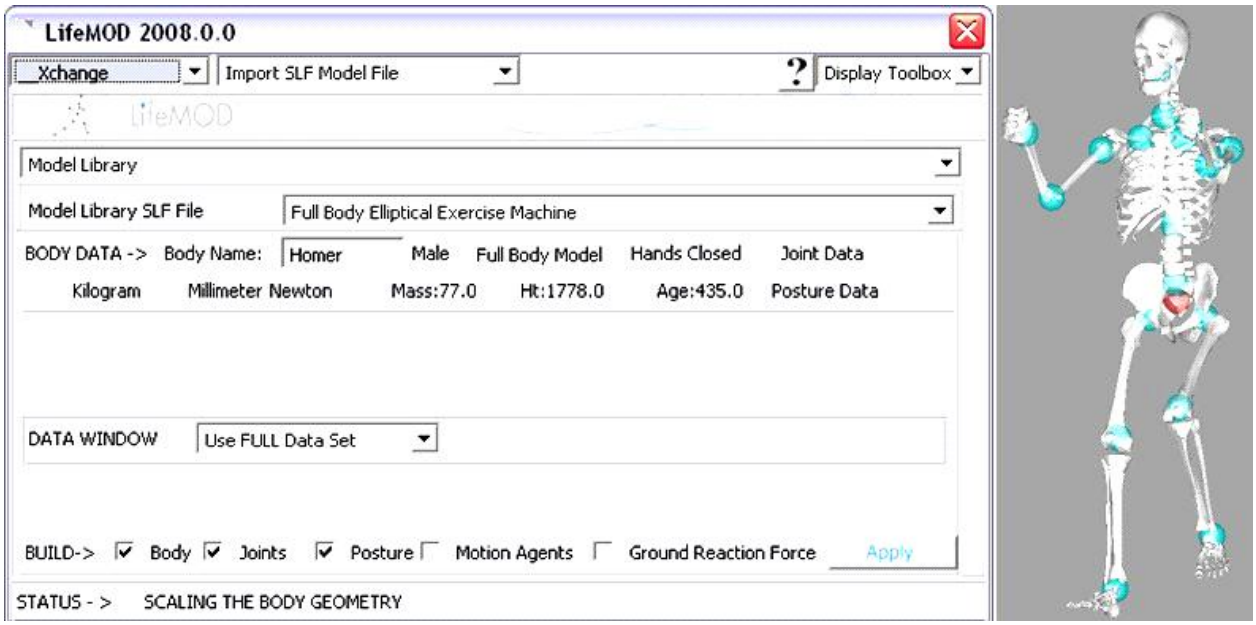


Рис. 1. Результирующая модель тела человека

Создание мягких тканей

Следующим шагом в процессе получения модели должно быть создание мягких тканей (мышц). LifeMOD™ автоматически создает набор основных групп мышц тела человека. Модели мышц содержат регистрирующие или обучаемые элементы. Регистрирующие элементы – простые коллекторы данных, которые записывают хронологию сокращения мышцы в период активности, когда модель движется, используя внешние приводы, такие как агенты движения. Обучаемые элементы могут быть или PID замкнутыми силовыми приводами, или разомкнутыми силовыми приводами, управляемыми кривой привода, приводящими в движение скелет модели. Параметры мышц такие, как физиологическая площадь поперечного сечения (pCSA) и максимальное напряжение мышечной ткани, используются для определения максимального потенциала силы специфической мышцы. LifeMOD™ содержит базу данных, содержащую значения pCSA для каждой

мышцы, которые масштабируются соответственно входным параметрам тела (рост, вес, пол и возраст). Кроме того, значение силы мышцы может масштабироваться от 0 до 200 % для изменения вклада каждой специфической мышцы.

Шаг 3: Открыть панель мягких тканей

Выбрать SOFT TISSUES в главном меню и CREATE BASE TISSUE SET в подменю.

Шаг 4: Установить поля для генерации мышц

Выбрать «Prepare Model with Recording Muscle Elements (To be trained in an inverse-dynamics simulation)», чтобы открыть панель, показанную на рисунке 2.

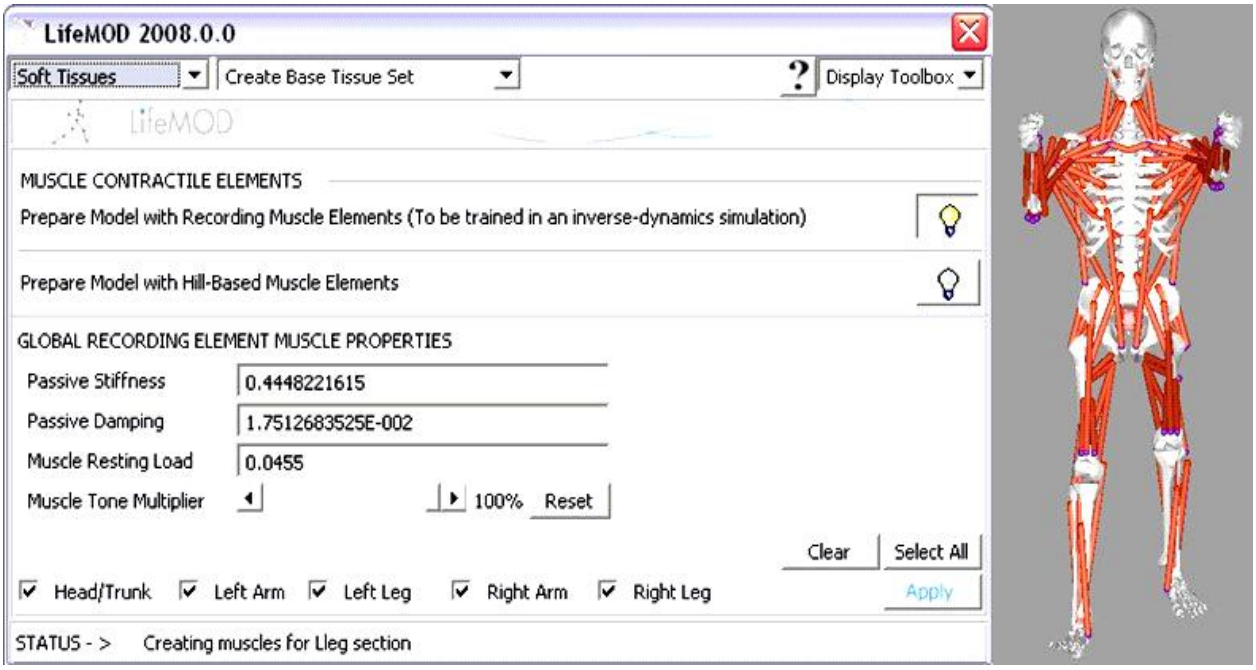


Рис. 2. Набор мышц созданный в модели тела

Установить пассивную жесткость равной 0,4448221615 и пассивное демпфирование равным 1,7512683525E-002, нагрузку мышцы в покое равной 0.0455 и коэффициент мышечного тонуса равным 200 %.

Шаг 5: Создать мягкие ткани на теле

Выбрать APPLY для создания наборов мышц на теле человека.

Объединение модели человека с моделью шагающего тренажера

Модель тренажера была создана заранее и уже содержится в библиотеке моделей LifeMOD. Модель состоит из шести деталей: ручки, шаговые платформы, диск и рама. Для соединения всех частей механизма используются простые шарниры вращения. К диску добавлен момент с линейной зависимостью (рисунок 3).

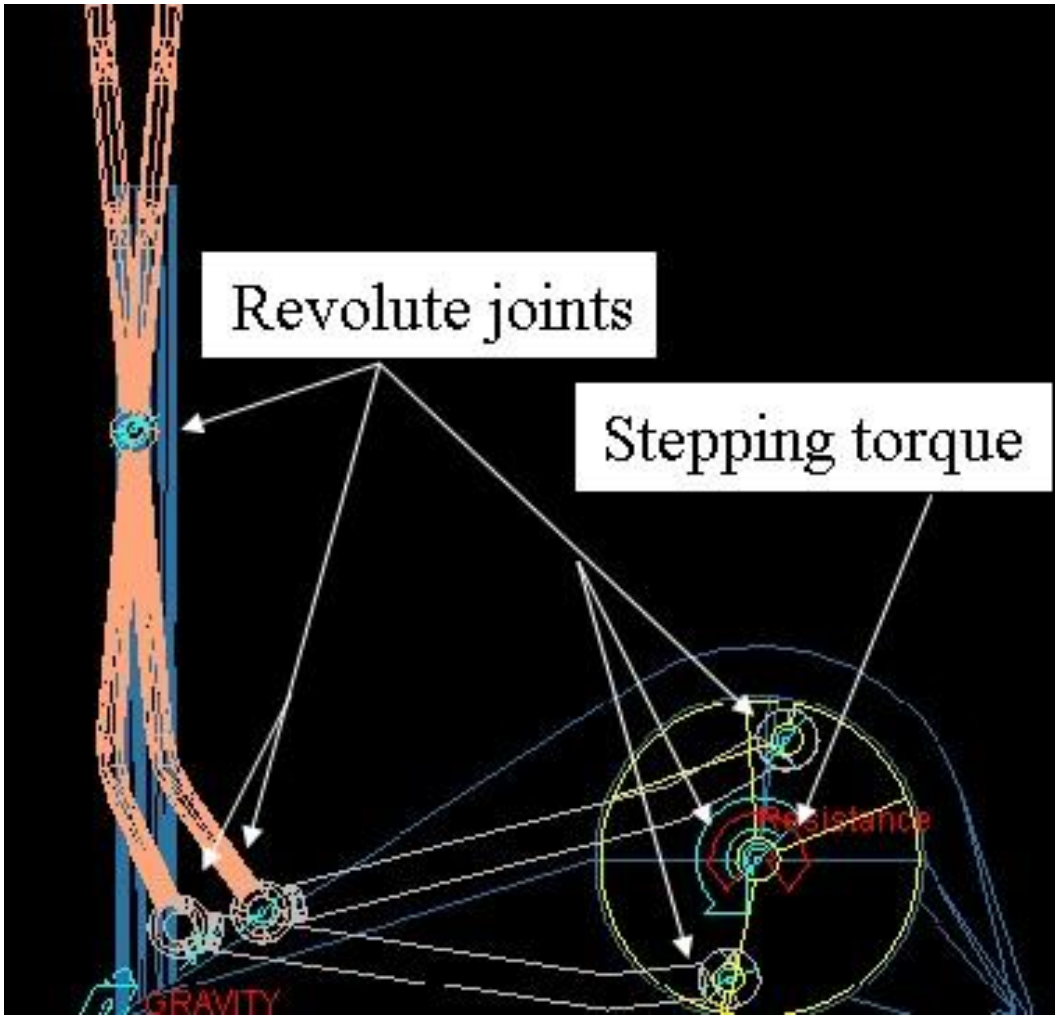


Рис. 3. Предварительно сформированная модель шагающего тренажера, объединенная с моделью человека

Шаг 6: Импортировать модель эллиптического (шагающего) тренажера из библиотеки внешних механизмов.

Выбрать XCHANGE в главном меню и IMPORT MECHANICAL ENVIRONMENT в подменю. Выбрать Stepper Machine как Model Library SLF File и выбрать APPLY для создания модели (см. рисунок 4).

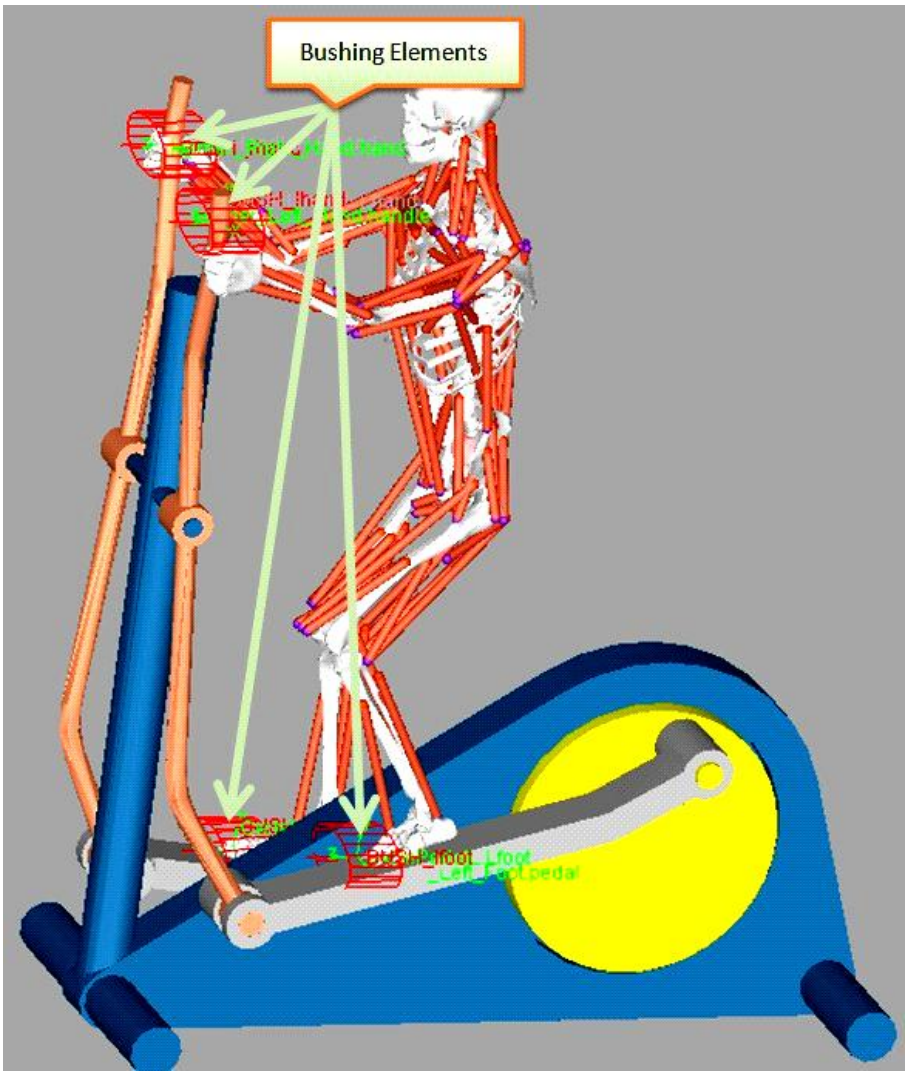


Рис. 4. Модель человека связана с шагающим тренажером через гибкие переходные элементы (bushing) ADAMS

Шаг 7: Создать гибкие переходные элементы (bushing) между левой ногой и левой педалью. Создать гибкий переходной элемент (bushing) между левой ногой и левой педалью, используя основную панель. Выбрать пиктограмму переходного элемента (bushing) в поле constraints внизу основной панели. Выбрать .World. Homer_Left_Foot, как первое тело и .World. CyclePedalL, как второе тело. Щелкнуть правой кнопкой мыши на основании тренажера, чтобы открыть панель навигации под основной панелью. Установить координаты положения равными (133, 311, 413) и выбрать OK. Переименовать .World. BUSH_lfoot. Изменить и установить параметры как показано на рисунке 5.

Использовать следующие команды ADAMS/View для создания нового маркера на ноге и маркера на педали:

marker create marker=.World.Homer_Left_Foot.pedal location=133, 311, -413 ori=0,0,0 rel=.World

marker create marker=.World.CyclePedalL.Lfoot location=133, 311, -413 ori=0,0,0 rel=.World

Использовать следующие команды ADAMS/View для создания переходного элемента (bushing) между маркерами:

force create element bushing bushing=.World.BUSH_lfoot i_mark=.World.Homer_Left_Foot.pedal j_mark=.World.CyclePedalL.Lfoot stiffness=1e4,1e4,1e4

damping=1e3,1e3,1e3 force_preload=0.0,0.0,0.0 tstiffness=1e6,1e6,1e6 tdamping=1e5,1e5,1e5 torque_preload=0.0,0.0,0.0

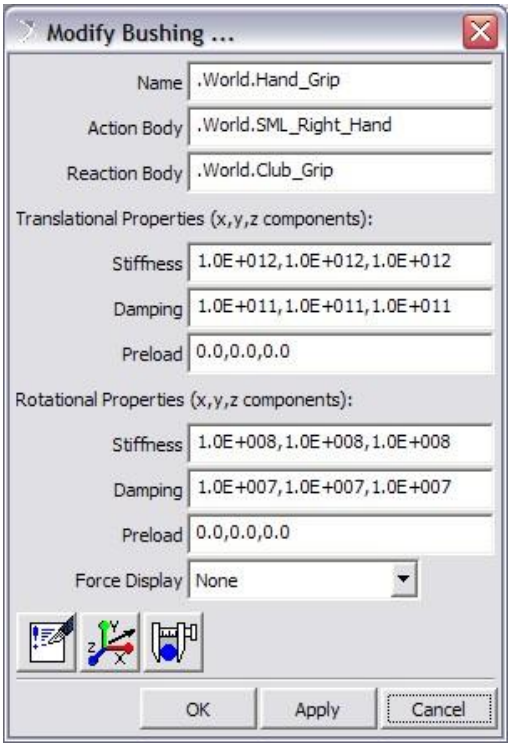


Рис. 5. Параметры переходного элемента (bushing)

Шаг 8: Создать гибкие переходные элементы (bushing) между правой ногой и правой педалью

Создать гибкий переходной элемент (bushing) между правой ногой и правой педалью, используя основную панель. Выбрать пиктограмму переходного элемента (bushing) в поле constraints внизу основной панели. Выбрать.World.Homer_Right_Foot, как первое тело и .World.CyclePedalR, как

второе тело. Щелкнуть правой кнопкой мыши на основании тренажера, чтобы открыть панель навигации под основной панелью. Установить координаты положения равными (-165, 201,-294) и выбрать OK. Переименовать .World.BUSH_rfoot. Изменить и установить параметры, как показано на рисунке 5.

Использовать следующие команды ADAMS/View для создания нового маркера на ноге и маркера на педали:

marker create marker=.World.Homer_Right_Foot.pedal location=-165, 201, -294ori=0,0,0 rel=.World

marker create marker=.World.CyclePedalR.Rfoot location=-165, 201, -294 ori=0,0,0 rel=.World

Использовать следующие команды ADAMS/View для создания переходного элемента (bushing) между маркерами:

force create element bushing bushing=.World.BUSH_rfoot i_mark=.World.Homer_Right_Foot.pedal j_mark=.World.CyclePedalR.Rfoot stiffness=1e4,1e4,1e4

damping=1e3,1e3,1e3 force_preload=0.0,0.0,0.0 tstiffness=1e6,1e6,1e6 tdamping=1e5,1e5,1e5 torque_preload=0.0,0.0,0.0

Шаг 9: Создать гибкие переходные элементы (bushing) между левой рукой и левой ручкой тренажера

Создать гибкий переходной элемент (bushing) между левой рукой и левой ручкой тренажера, используя основную панель. Выбрать пиктограмму переходного элемента (bushing) в поле constraints внизу основной панели. Выбрать.World.Homer_Left_Hand, как первое тело и.World.CycleHandlelL, как второе тело. Щелкнуть правой кнопкой мыши на основании тренажера, чтобы открыть панель навигации под основной панелью. Установить координаты положения равными (253, 1634, 69) и выбрать OK. Переименовать .World.BUSH_lhand. Изменить и установить параметры, как показано на рисунке 5.

Можно еще использовать следующие команды ADAMS/View для создания нового маркера на руке и маркера на ручке тренажера:

marker create marker=.World.Homer_Left_Hand.handle location= 253, 1634, 69 ori=0,0,0 rel=.World

marker create marker=.World.CycleHandleL.Lhand location= 253, 1634, 69 ori=0,0,0 rel=.World

Использовать следующие команды ADAMS/View для создания переходного элемента (bushing) между маркерами:

force create element bushing bushing=.World.BUSH_lhand i_mark=.World.Homer_Left_Hand.handle j_mark=.World.CycleHandleL.Lhand stiffness=1e4,1e4,1e4

damping=1e3,1e3,1e3 force_preload=0.0,0.0,0.0 tstiffness=1e6,1e6,1e6 tdamping=1e5,1e5,1e5 torque_preload=0.0,0.0,0.0

Шаг 10: Создать гибкие переходные элементы (bushing) между правой рукой и правой ручкой тренажера

Создать гибкий переходной элемент (bushing) между правой рукой и правой ручкой тренажера, используя основную панель. Выбрать пиктограмму переходного элемента (bushing) в поле constraints внизу основной панели. Выбрать.World.Homer_Right_Hand, как первое тело и .World.CycleHandlelR, как второе тело. Щелкнуть правой кнопкой мыши на основании тренажера, чтобы открыть панель навигации под основной панелью. Установить координаты положения равными (-253, 1574, -27) и параметры как показано на рисунке 5.

Использовать следующие команды ADAMS/View для создания нового маркера на руке и маркера на ручке тренажера:

marker create marker=.World.Homer_Right_Hand.handle location= -253, 1574, -27 ori=0,0,0 rel=.World

marker create marker=.World.CycleHandleR.Rhand location=-253, 1574, -27 ori=0,0,0 rel=.World

Использовать следующие команды ADAMS/View для создания переходного элемента (bushing) между маркерами:

force create element bushing bushing=.World.BUSH_rhand i_mark=.World.Homer_Right_Hand.handle j_mark=.World.CycleHandleR.Rhand stiffness=1e4,1e4,1e4

damping=1e3,1e3,1e3 force_preload=0.0,0.0,0.0 tstiffness=1e6,1e6,1e6 tdamping=1e5,1e5,1e5 torque_preload=0.0,0.0,0.0

Добавление агентов движения

В фазе моделирования обратной задачи динамики, тренажер будет фактически генерировать движение, а модель человека будет реагировать на активность тренажера. Агенты движения, добавленные в области таза и головы, в течение этой фазы используются для стабилизации модели в процессе обратного моделирования. Они будут зафиксированы в пространстве и подключены к модели через пружины с малыми значениями жесткости и демпфирования (см. рисунок 6).

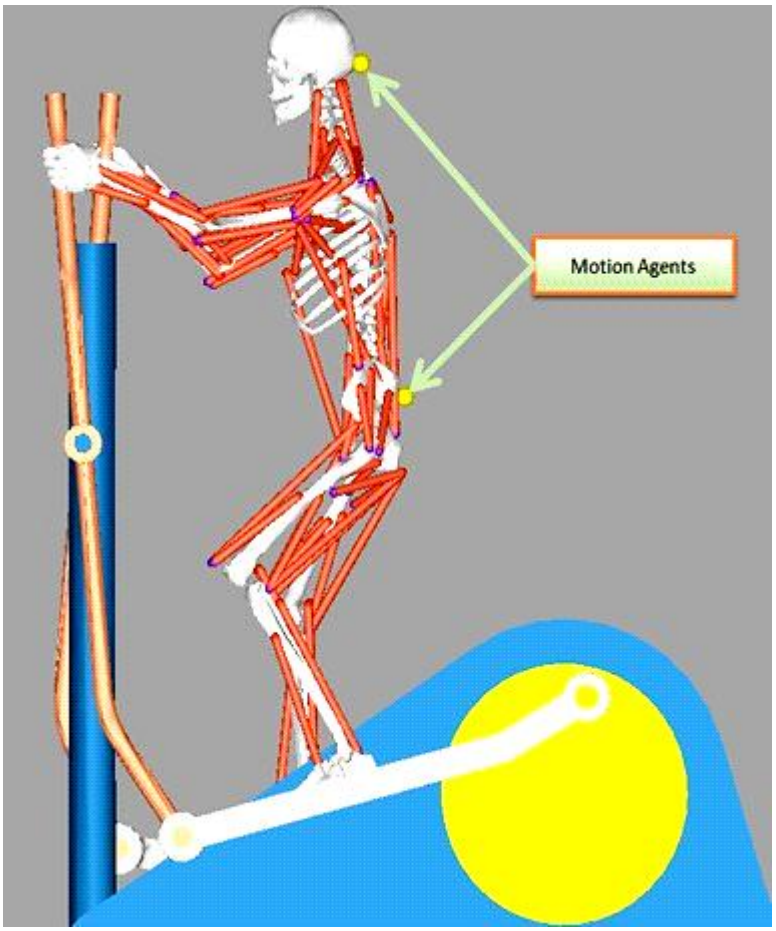


Рис. 6. Агенты движения добавляются в области таза и головы для стабилизации модели в процессе обратного моделирования

Шаг 11: Открыть панель создания агентов движения

Выбрать MOTION в главном меню и CREATE INDIVIDUAL MOTION AGENT в подменю.

Шаг 12: Создать агент движения в области таза

Выбрать.World.Homer_Lower_Torso как анатомический сегмент и выбрать Manually Select Location для выбора метода позиционирования

агента, используя положение (-4, 1092, -661). Установить значение жесткости на панели рисунка 7. Указать все степени свободы, которые должны быть зафиксированы.

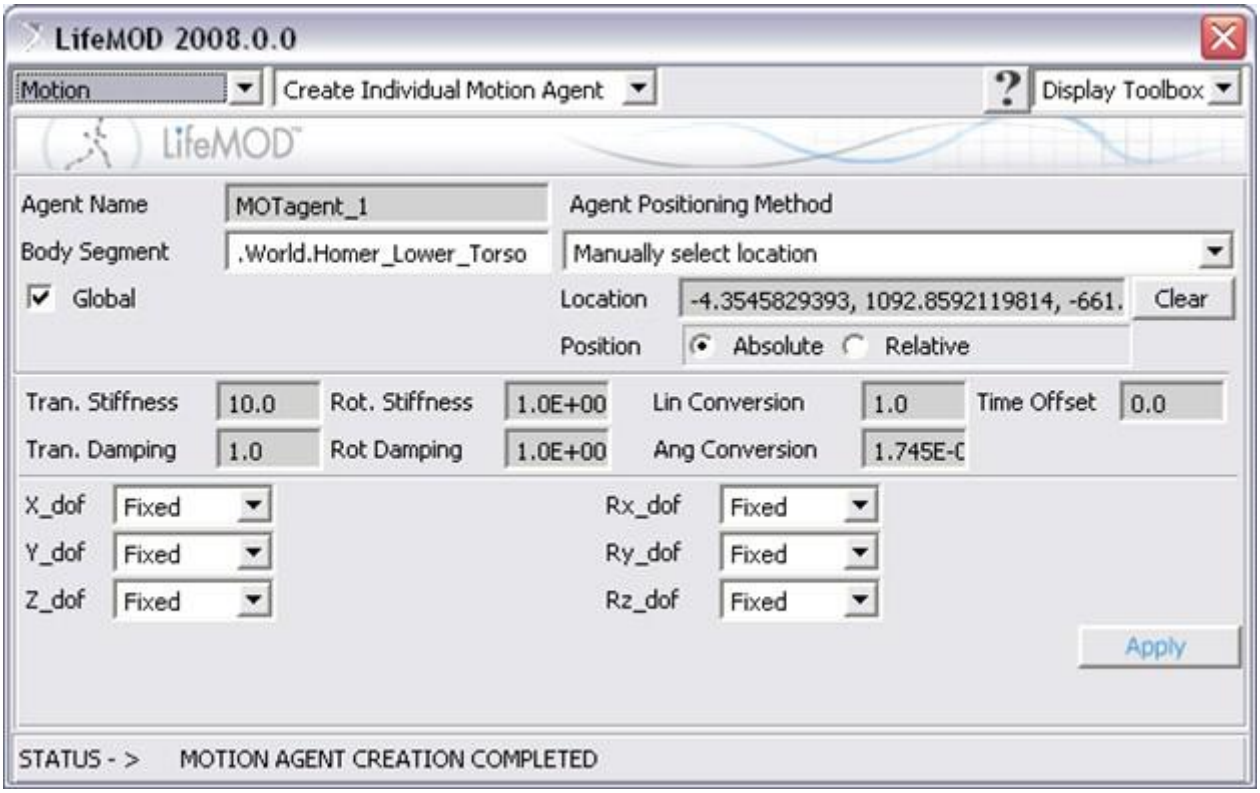


Рис. 7. Панель создания агентов движения

Шаг 13: Создать агент движения в области головы

Выбрать World.Homer_Head как анатомический сегмент и выбрать Manually Select Location для выбора метода позиционирования агента, используя положение (5, 1757, 568). Установить значение жесткости на панели рисунка 7. Указать все степени свободы, которые должны быть зафиксированы.

Моделирование равновесного состояния

Чтобы обеспечить качественное моделирование как для обратной задачи динамики так и для прямой задачи динамики, рекомендуется, чтобы было выполнено моделирование состояния равновесия, чтобы уравновесить силы в модели. В этом случае устанавливается устойчивое равновесие между моделью человека и силой тяготения. Для этого необходимо следующее.

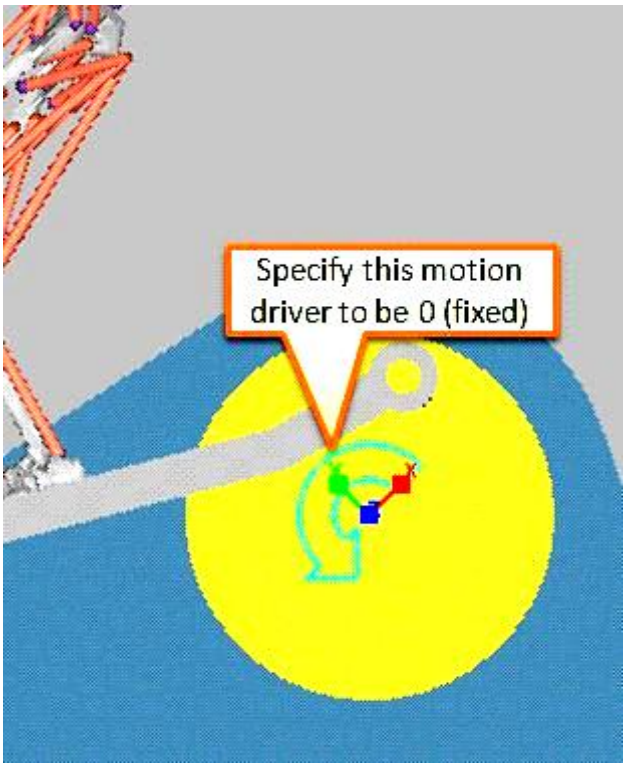


Рис. 8. Соединения ограничивающие связь модели со средой

Шаг 14: Установить привод вращения колеса в 0

Щелкнуть правой кнопкой мыши на Motion: Driver и выбрать MODIFY. Установить значение функции равным 0. Выбрать OK (см. рисунок 8).

Можно использовать следующие команды ADAMS/View для модификации привода вращения:

constraint modify motion motion_name = .World.Driver function = "0"

Шаг 15: Вызвать панель анализа

Выбрать ANALYZE в главном меню и DYNAMICS в подменю. Выбрать метод интегрирования "Default".

Шаг 16: Выполнить моделирование

Установить силу тяготения равной -9806.65 в направлении y и задать конечное время интегрирования равным 1 сек и число шагов равным 100. Выбрать ANALYZE.

Шаг 17: Отобразить анимацию

Использовать панель ADAMS/View для анимации модели.

Шаг 18: Обновить конфигурацию модели по результатам статического анализа

Выбрать "Update Model Posture with Equilibrium Results" для изменения позиции тела в соответствии с последними данными моделирования.

Моделирование обратной задачи динамики

После создания модели человека в требуемой позиции, модели связанной с шагающим тренажером и установленных агентов движения, может быть выполнено моделирование обратной задачи динамики. В этой фазе тренажер будет фактически управлять моделью. Это моделирование выполняется для записи хронологии сокращения мышц при ходьбе. При моделировании прямой задачи динамики, которое будет выполнено позже, хронология сокращения мышц используется в элементах сокращения в мышцах для создания сил, позволяющих модели воспроизводить движение.

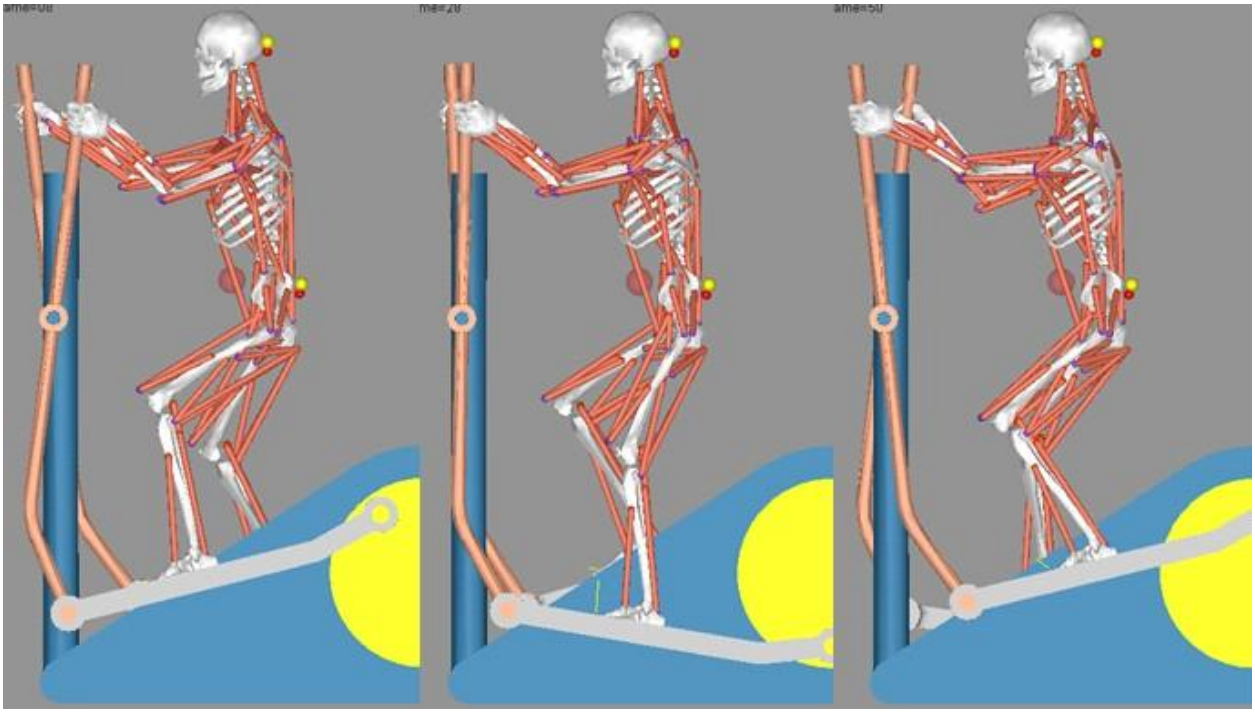


Рис. 9. Последовательные кадры анимации при моделировании обратной задачи динамики

Шаг 19: Обновить привод вращения на колесе

Щелкнуть правой кнопкой мыши на Motion: Driver и изменить значение функции на -360d*time.

Использовать следующие команды ADAMS/View для обновления привода вращения на колесе:

constraint modify motion motion_name = .World.Driver function = "-360d*time"

Шаг 20: Выполнить моделирование динамики

Проверить поле силы тяготения и установить у-значение равным -9806.65. Выбрать параметры метода интегрирования "Default". Задать конечное время интегрирования равным 3 с и число шагов равным 150 временным интервалам. Установить параметры метода интегрирования по умолчанию и выбрать ANALYZE.

Шаг 21: Отобразить анимацию

Использовать панель ADAMS/View для анимации модели (см. рисунок 9).

Подготовка модели для решения прямой задачи динамики

Хронология сокращения мышц, записанная при моделировании обратной задачи динамики, теперь используется в линейной PD-Servo формулировке для генерации сил при обновлении хронологии движения. В результате этого процесса деактивируются агенты движения и модифицируются функции мышц.

Агенты движения удаляются из модели и в качестве их замены устанавливается «Агент-трекер». Агент-трекер является агентом движения, расположенным в центре области таза, который обеспечивает силовую стабилизацию при моделировании прямой задачи динамики. В процессе моделирования обратной задачи динамики записываются позиция и ориентация базиса агент-трекера (при этом в процессе моделирования обратной задачи динамики силы не генерируется). Информация о положении и ориентации агент-трекера затем может использоваться для управления агент-трекером при моделировании прямой задачи динамики. Обычно «освобождаются» некоторые степени свободы, чтобы допустить требуемое динамическое взаимодействие. Для этого примера движение в направлении нормали к поверхности пола определяется как свободное, чтобы допустить правильное определение реакции между ногами и педалями с учетом силы тяготения.

В этом примере агент-трекер учитывает тот факт, что верхняя часть тела исключается из модели. Его присутствие компенсирует усилия рук и верхней части тела, передаваемые через туловище. Создание агента-трекера

выполнятся с помощью панели, приведенной на рисунке 10, и требует осуществления следующих шагов.

Шаг 22: Открыть панель обучения мышц

Выбрать SOFT TISSUES в главном меню и TRAINING в подменю.

Шаг 23: Установить элемент сокращения в состояние ACTIVE

Выбрать «Install Trained Closed-loop Contrractile Elements on Muscles».

Рис. 10. Панель для создания агента-трекера

Шаг 24: Установить соответствующие поля и обновить суставы

Задать с помощью панели рисунка 11 значение пропорционального приращения равным 1e7, приращения интегралов равным 1e6 и приращения производных равным 1e4. Эти величины управляют тем, как точно PID-servo актуаторы будут отслеживать требуемое сокращение мышц на каждом временном шаге анализа. Заметим, что отдельная мышца не может развивать силу больше физического предела, определяемого физиологическим поперечным сечением мышцы (pCSA) и максимальным удельным усилием в тканях. Выбрать APPLY для обновления определения мышц.

Шаг 25: Открыть панель для создания агента-трекера

Выбрать MOTION в главном меню и CREATE TRACKER AGENT в подменю.

Шаг 26: Создать агент-трекер

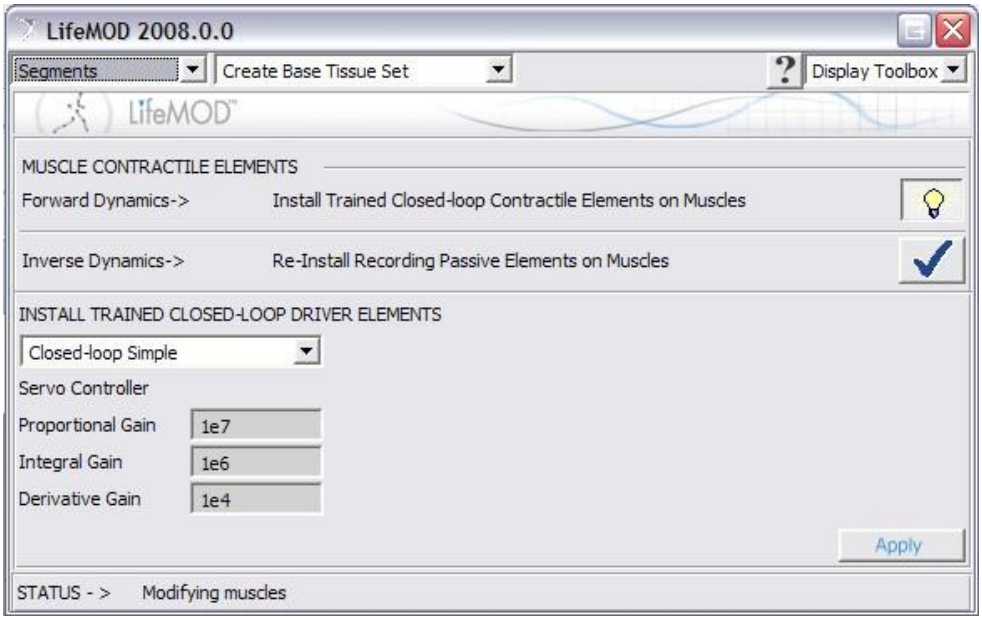


Рис. 11. Панель для установки замкнутых приводных элементов мышц

Задать параметры жесткости/демпфирования так, как показано на рисунке 10. Определить все степени свободы как ведомые кроме Y-Dof. Выбрать APPLY

Шаг 27: Отключить привод колеса

Выбрать привод вращения колеса (см. рисунок 12) и выбрать (DE)ACTIVATE.

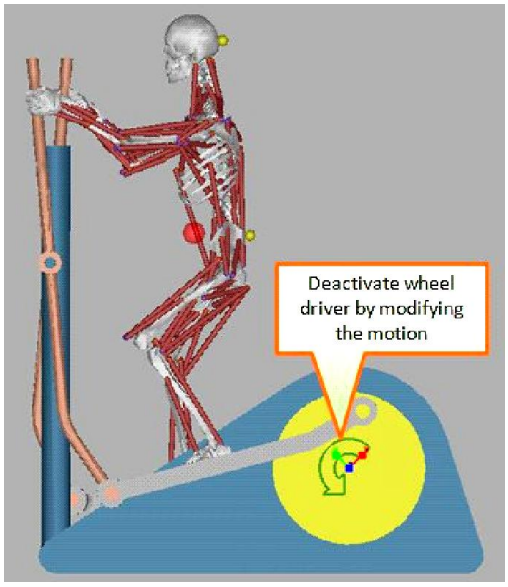


Рис. 12. Отключение привода колес и активация момента сопротивления колеса для анализа прямой задачи динамики

Моделирование прямой задачи динамики

При отключенном приводе колеса и наличии момента сопротивления на колесе модель человека готова управлять шагающим тренажером, используя силу мышц.

Шаг 28: Открыть панель анализа

Выбрать ANALYZE в главном меню и DYNAMICS в подменю.

Шаг 29: Отключить агенты движения и запустить моделирование прямой задачи динамики (см. рисунок 13).

Выбрать "Disable Motion Agents". Запустить моделирование в течение 3.0 с и с 300 временными шагами, используя параметры метода интегрирования по умолчанию. Выбрать ANALYZE

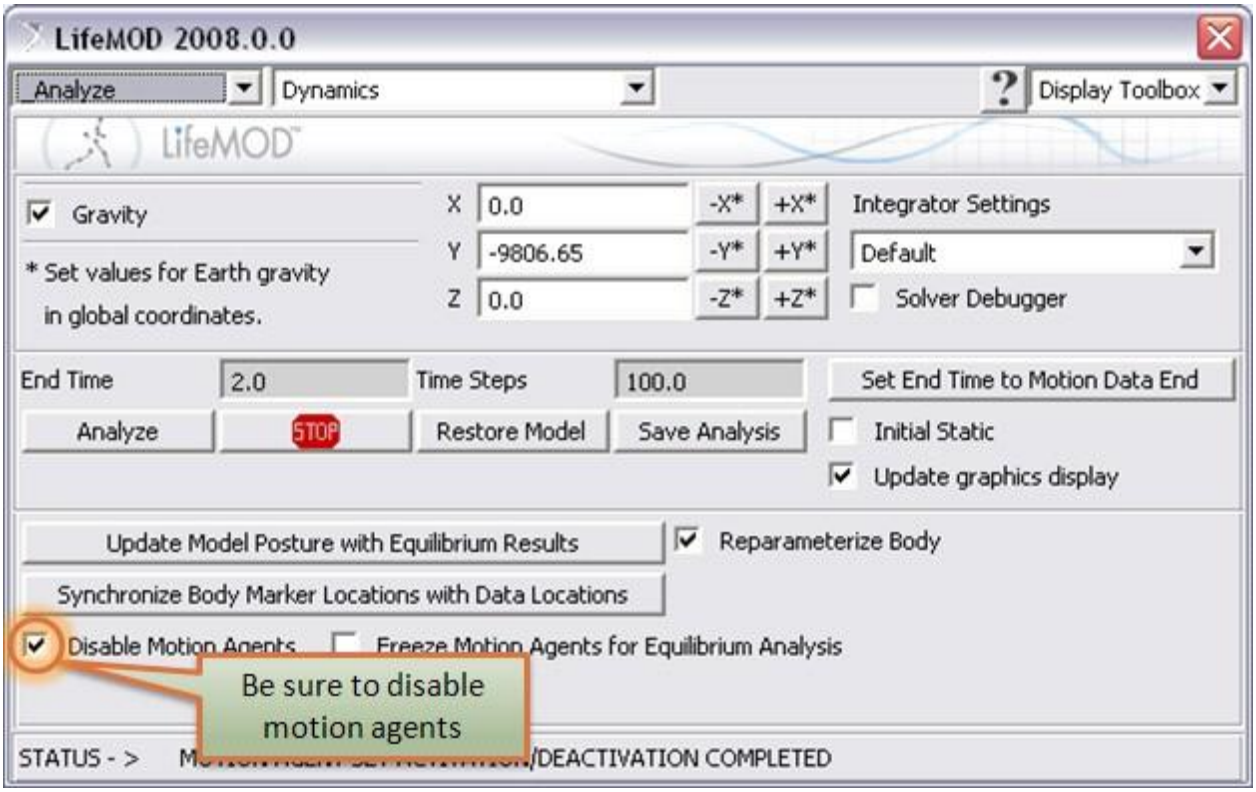


Рис. 13. Процедура отключения агентов движения и запуска моделирования прямой задачи динамики

Шаг 30: Отобразить анимацию результатов моделирования

Использовать панель ADAMS/View для анимации результатов моделирования.

Шаг 31: Сохранить результаты моделирования

На панели анализа выбрать Save Analysis и ввести Case_1, затем - OK.

Шаг 32: Открыть панель результатов

Выбрать RESULTS в главном меню и DATA DISPLAY в подменю. Выбрать Soft Tissues как Data Type.

Шаг 33: Отобразить ленточные диаграммы мышц левой и правой ноги

Чтобы наблюдать изменение сил развиваемых мышцами, выбрать Homer_VasLat_Rtiss_1 как Soft Tissue, Tension как Characteristic и выбрать «Create Strip Chart Measure». Выбрать Homer_Iliac_Ltiss_1 как Soft Tissue, Tension как Characteristic и выбрать «Create Strip Chart Measure».

Шаг 34: Включить масштабирование графики мышц

Выбрать ANIMATION в подменю. Включить глобальное масштабирование графики мышц, выбрав Scale Joint/Tissue Graphics, Tissues, Scale Globally и пиктограмму включенной лампы для масштабирования графики мышц.

Шаг 35: Выключить отображение тренажера и отобразить анимацию

Для наглядности выключить отображение графики тренажера, используя следующую команду Adams/View: group att vis=off group=machine

Использовать панель ADAMS/View для отображения анимации модели. Выбрать правый вид и запустить анимацию (см. рисунок 14).

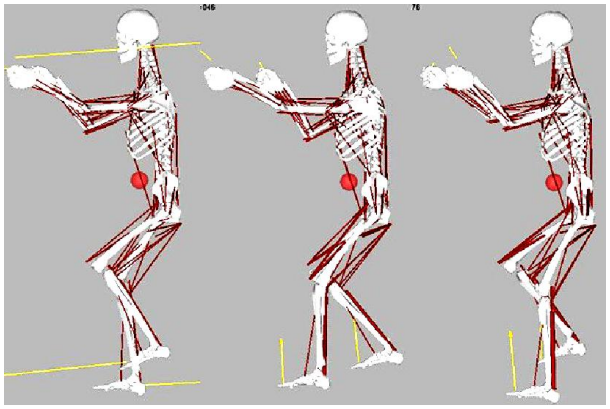


Рис. 14. Кадры анимации моделирования прямой задачи динамики (изображение тренажера выключено для наглядности)

Шаг 36: Сохранить кривые анимации

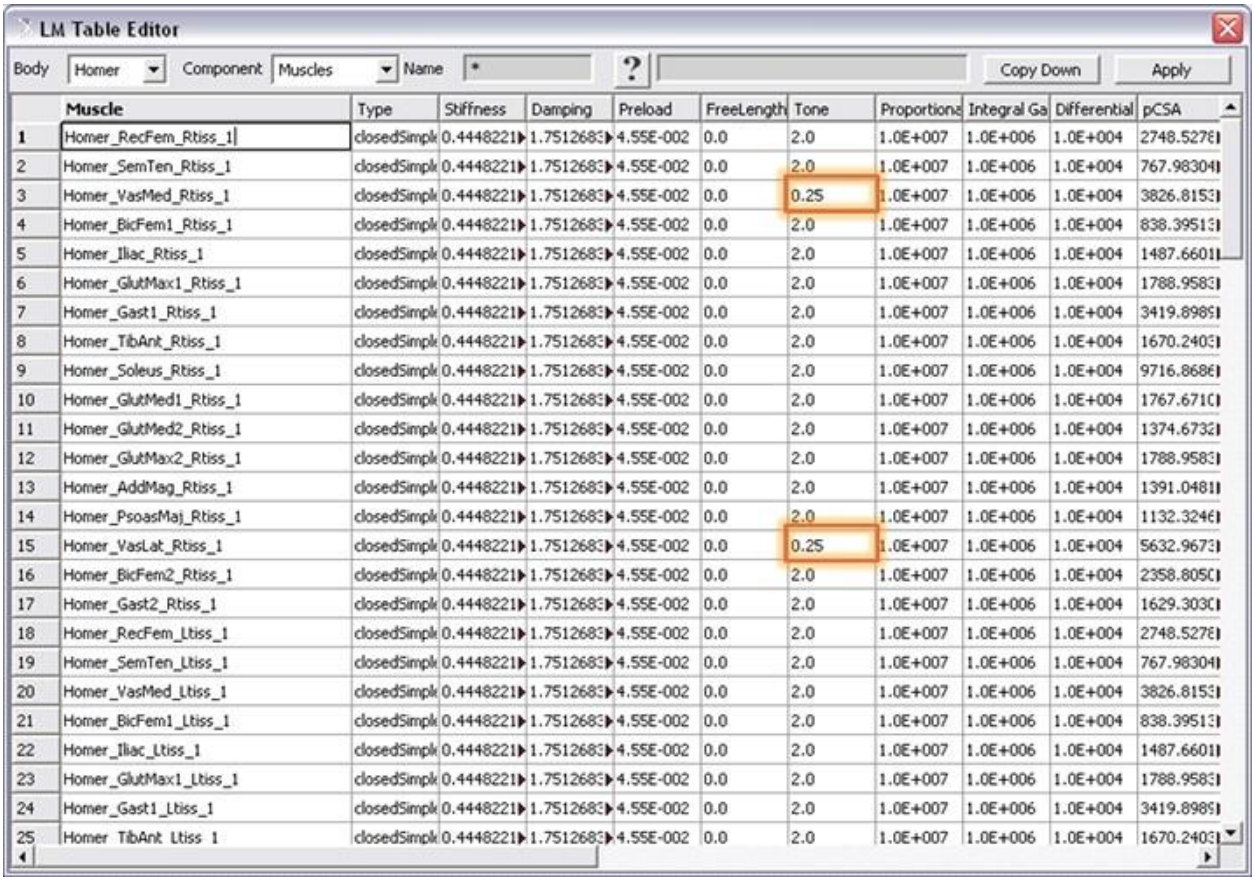
На каждой ленточной диаграмме щелкнуть правой кнопкой мыши на графике и выбрать «save».

Моделирование системы «человек на шагающем тренажере»

LifeMOD ™ решает проблему избыточных мышц в механике тела человека, допуская равный вклад для каждой мышцы, участвующей в движении сустава. Этот вклад создается максимальным выходом силы

каждой мышцы и может в дальнейшем быть изменен пользователем. В этом разделе пользователь будет уменьшать вклады трех мышц, чтобы исследовать эффекты влияния перераспределения нагрузки на другие мышцы.

Результаты моделирования показывают, что уменьшение силы разгибателя колена на правой ноге, должно компенсироваться увеличением силы разгибателя бедра на левой ноге. Шаги программирования модели следующие. Используется панель рисунка 15.



	Muscle	Type	Stiffness	Damping	Preload	FreeLength	Tone	Proportiona	Integral Ga	Differential	pCSA
1	Homer_RecFem_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	2748.527
2	Homer_SemTen_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	767.98304
3	Homer_VasMed_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	0.25	1.0E+007	1.0E+006	1.0E+004	3826.8153
4	Homer_BicFem1_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	838.39513
5	Homer_Iliac_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1487.6601
6	Homer_GlutMax1_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1788.9583
7	Homer_Gast1_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	3419.8985
8	Homer_TibAnt_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1670.2403
9	Homer_Soleus_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	9716.8686
10	Homer_GlutMed1_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1767.671
11	Homer_GlutMed2_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1374.673
12	Homer_GlutMax2_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1788.9583
13	Homer_AddMag_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1391.0481
14	Homer_PsoasMaj_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1132.324
15	Homer_VasLat_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	0.25	1.0E+007	1.0E+006	1.0E+004	5632.967
16	Homer_BicFem2_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	2358.805
17	Homer_Gast2_Rtiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1629.303
18	Homer_RecFem_Ltiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	2748.527
19	Homer_SemTen_Ltiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	767.98304
20	Homer_VasMed_Ltiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	3826.8153
21	Homer_BicFem1_Ltiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	838.39513
22	Homer_Iliac_Ltiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1487.6601
23	Homer_GlutMax1_Ltiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1788.9583
24	Homer_Gast1_Ltiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	3419.8985
25	Homer_TibAnt_Ltiss_1	closedSimpl	0.4448221	1.7512683	4.55E-002	0.0	2.0	1.0E+007	1.0E+006	1.0E+004	1670.2403

Рис. 15. Панель свойств мышц правой ноги, используемая для уменьшения вклада мышц Vastus Medialis и Vastus Lateralis до 25%

Шаг 1: Открыть панель редактирования обучения тканей

Выбрать TABLE EDITOR в правом верхнем углу. Выбрать MUSCLES как компонент.

Шаг 2: Уменьшить вклад широких мышц бедра правой ноги

Уменьшить тонус широкой медиальной и широкой латеральной мышц бедра с 1,0 до 0,25 и выбрать APPLY.

Шаг 3: Открыть панель анализа

Выбрать ANALYZE в главном меню и DYNAMICS в подменю.

Шаг 4: Отключить агенты движения и запустить моделирование прямой задачи динамики

Выбрать «Disable Motion Agents». Запустить моделирование в течение 3.0 seconds и с 300 временными шагами, используя параметры метода интегрирования по умолчанию. Выбрать ANALYZE.

Шаг 5: Отобразить анимацию результатов моделирования

Использовать панель ADAMS/View для отображения анимации модели.

Шаг 6: Сохранить результаты моделирования

В панели анализа выбрать Save Analysis и ввести Case_2, затем - OK.

После завершения моделирования модель может быть анимирована и могут быть проанализированы результаты моделирования. По итогам моделирования прямой задачи динамики могут быть представлены различные результаты, включая:

- всю историю изменения мышечных сил;
- всю историю изменения мышечных сокращений;
- контактные силы между руками и тренажером;
- контактные силы между ногами и тренажером.

Для этого необходимо выполнить такие шаги.

Шаг 7: Открыть панель результатов

Выбрать RESULTS в главном меню и DATA DISPLAY в подменю. Выбрать Soft Tissues как Data Type. Выбрать кнопку «Results Window» для получения результатов.

Шаг 8: Построить график изменения момента сопротивления тренажера

В нижнем окне вывода, пролистать до конца требований и выбрать Machine_Torque и компонент U2. Выбрать ADD CURVES.

Шаг 9: Анимировать вид сбоку

Выбрать ANIMATION в подменю. Выбрать вид справа, разделить окно. Выбрать PLAY.

Шаг 10: Сравнить контактные реакции левой ноги/тренажера для двух случаев.

Использовать панель ADAMS/View для генерации графика силы реакции ноги, используя запрос .World.case_1.Interface_Lfoot и компонент U2 для случая 1 , World.case_2.Interface_Lfoot и компонент U2 для случая 2.

Шаг 11: Включить векторы сил контакта с тренажером и анимировать изометрический вид.

Включить векторы сил контакта рук и ног с тренажером, используя следующие команды ADAMS/View:

entity attributes entity_name = .World."*hand_force_graphic_1*" visibility = on

entity attributes entity_name = .World."*foot_force_graphic_1*" visibility = on

Шаг 12: Построить график изменения сил бицепса бедра (biceps femoris) - Case_1.

Выбрать DATA DISPLAY в подменю. Ввести Case_1 как вариант анализа. Выбрать Homer_Bicfem2_Ltiss_1 для получения характеристик силы и напряжения мышцы. Выбрать низкочастотный фильтр данных Butterworth с частотой отсечки равной 5,0 и порядком равным 1. Нажать «New Plot» и выбрать CREATE FULL PLOT для построения кривой.

Шаг 13: Построить график изменения сил бицепса бедра (biceps femoris) левой ноги – Case_2.

Ввести Case_1 как вариант анализа. Выбрать Homer_Bicfem2_Ltiss_1 для получения характеристик силы и напряжения мышцы. Выбрать низкочастотный фильтр данных Butterworth с частотой отсечки равной 5.0 и порядком равным 1. Выбрать CREATE FULL PLOT для построения кривой.

Шаг 14: Включить масштабирование графики мышц

Включить глобальное масштабирование графики мышц, выбрав Scale Joint/Tissue Graphics, Tissues, Scale Globally и включенную лампу для масштабирования графики мышц.

Шаг 15: Анимировать вид справа

Выбрать вид справа, разделить окно. Выбрать PLAY.

Шаг 16: Построить график изменения напряжения трехглавой мышцы плеча (medial triceps) правой руки – Case_1

Выбрать Homer_tric3_Rtiss_1 для получения характеристик силы и напряжения мышцы. Выбрать низкочастотный фильтр данных Butterworth с

частотой отсечки равной 5,0 и порядком равным 1. Нажать «New Plot» и выбрать CREATE FULL PLOT для построения кривой.

Шаг 17: Построить график изменения напряжения трехглавой мышцы плеча (medial triceps) правой руки – Case_2

Выбрать Homer_tric3_Rtiss_1 для получения характеристик силы и напряжения мышцы. Выбрать низкочастотный фильтр данных Butterworth с частотой отсечки равной 5,0 и порядком равным 1. Выбрать CREATE FULL PLOT для построения кривой.

Шаг18: Анимировать вид справа

Выбрать вид справа, разделить окно. Выбрать PLAY..

Шаг 19: Построить график изменения напряжения мышцы разгибающей спину (Erectus Spinaes) – Case_1

Выбрать Homer_ErecSpin1_Rtiss_1 для получения характеристик силы и напряжения мышцы. Выбрать низкочастотный фильтр данных Butterworth с частотой отсечки равной 5,0 и порядком равным 1. Нажать «New Plot» и выбрать CREATE FULL PLOT для построения кривой.

Шаг 20: Построить график изменения напряжения мышцы разгибающей спину (Erectus Spinaes) – Case_2

Выбрать Homer_ErecSpin1_Rtiss_1 для получения характеристик силы и напряжения мышцы. Выбрать низкочастотный фильтр данных Butterworth с частотой отсечки равной 5,0 и порядком равным 1. Нажать «New Plot» и выбрать CREATE FULL PLOT для построения кривой.

Шаг 21: Анимировать вид сзади

Выбрать вид сзади, разделить окно. Выбрать PLAY.

Результаты моделирования в виде анимации и графиков приведены на рисунках 16-21.

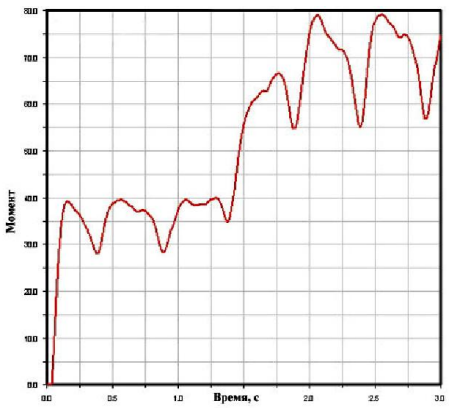


Рис. 16. График изменения момента сопротивления тренажера

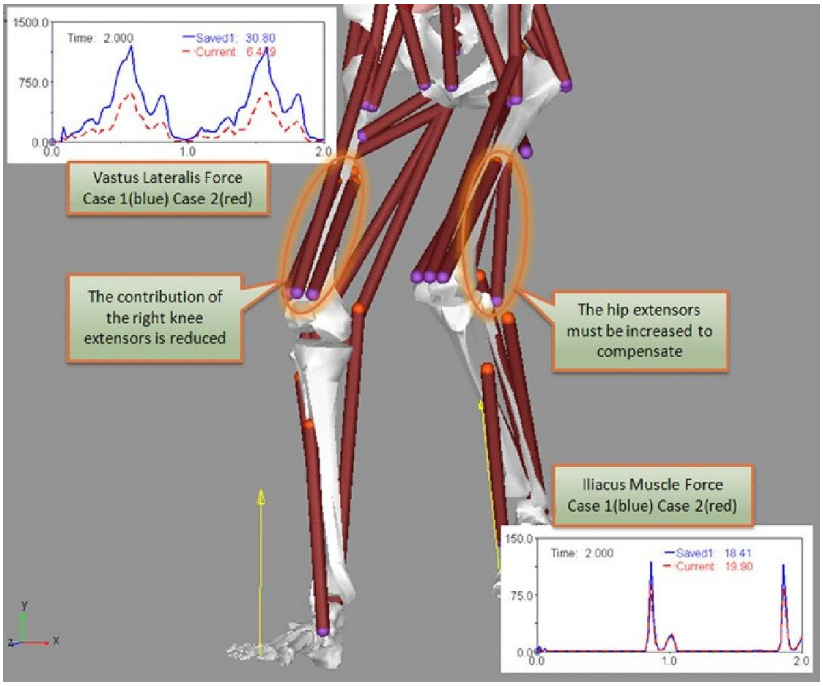


Рис. 17. Последовательность анимации и отображение информации по результатам сравнения случая 1 (до изменения вклада мышц) и случая 2 (после уменьшения GM и Soleus мышц)

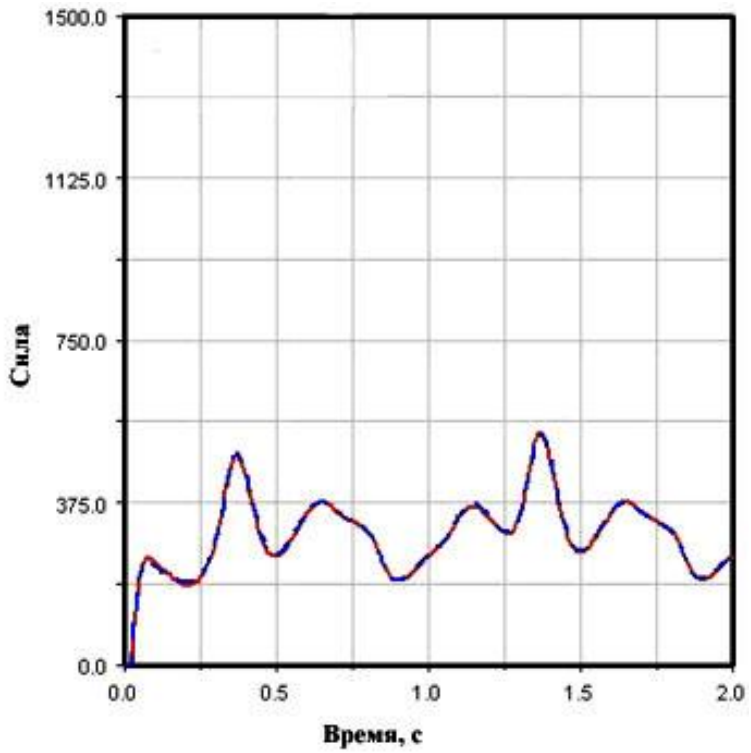


Рис. 18. График изменения сил взаимодействия левой ноги (красная линия) и тренажера (синяя линия)

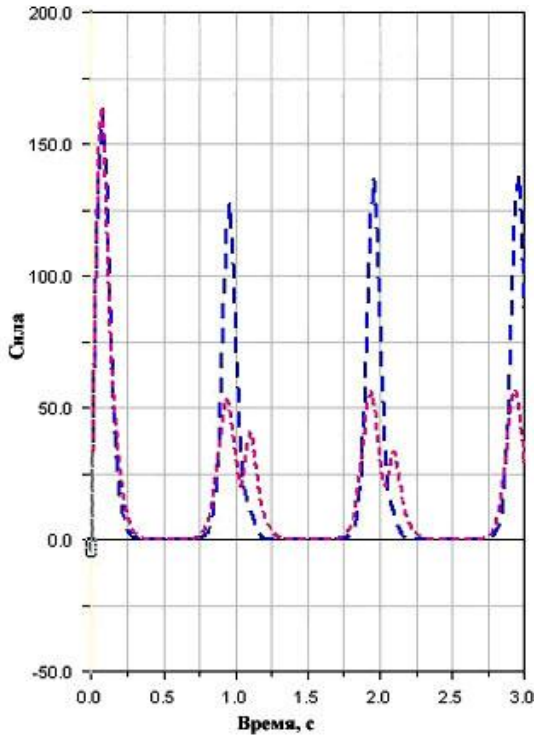


Рис. 19. Графики изменения силы трицепса правой руки для случая 1 (синий) и случая 2 (красный)

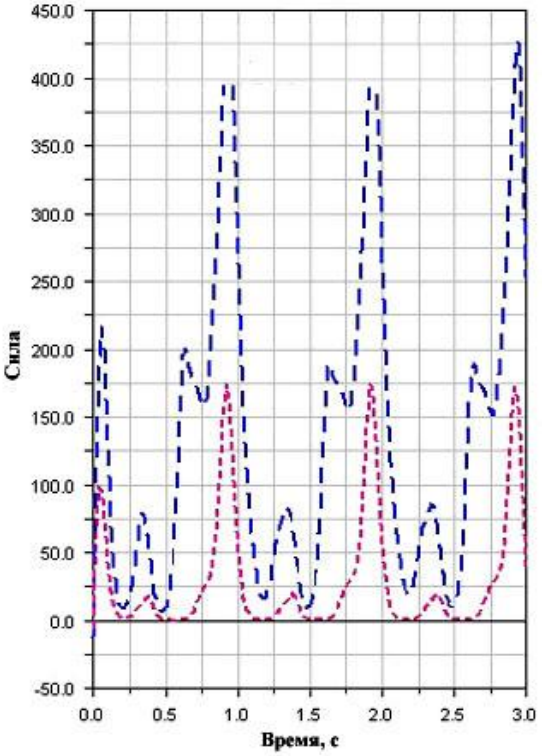


Рис. 20. Графики изменения силы, разгибающей мышцы спины для случая 1 (синий цвет) и случая 2 (красный цвет)

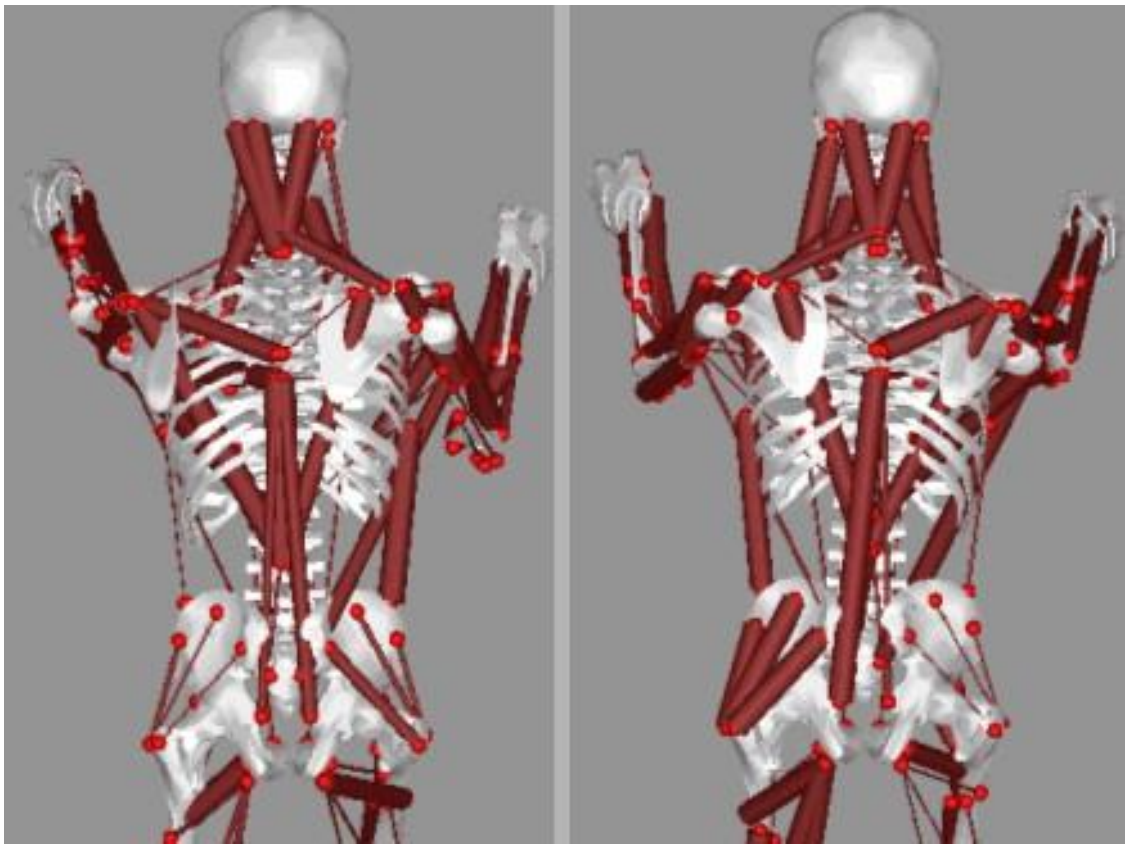


Рис. 21. Кадры анимации отображающие масштабирование графики силы мышц

Результаты экспериментального исследования компьютерных моделей подтвердили их адекватность, гибкость и большие функциональные возможности. Методология моделирования и библиотеки типовых составляющих человека и его движений носят общий характер и могут быть с успехом использованы при программировании различных спортивных упражнений и состязаний в разнообразных видах спорта, включая и игровые. Их внедрение в тренировочный процесс позволяет повысить его эффективность за счет учета конкретных антропологических данных спортсмена и действия реальных нагрузок, а так же за счет сокращения времени его выхода на пик спортивной формы. Используемые технологии и модели полностью соответствуют современному уровню компьютерного моделирования в области биомеханики и спорта.

Настоящая статья подготовлена по результатам НИР в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России» на 2009-2013 годы (государственный контракт №П1264 от 27.08.2009г.).

Литература:

1. Воронов А.В. Имитационное биомеханическое моделирование как метод изучения двигательных действий человека // Теория и практика физической культуры, 2004, №2, с.48-49.

2. Воронов А.В. Анатомическое строение и биомеханические характеристики мышц и суставов нижней конечности. – М.: Физкультура, образование, наука, 2003. – 203с.

3. Новожилов И.В. и др. Математическое моделирование сгибательно-разгибательных движений нижних конечностей при изменении вертикальной позы человека. – М.: Изд-во механико-математического факультета МГУ, 2001. – 52с.

УДК 681.3. 07

НЕОБХОДИМОСТЬ ЭЛЕКТРОННО-ЦИФРОВОЙ ПОДПИСИ В ЭЛЕКТРОННЫХ ДОКУМЕНТАХ

Е.С. Кузина, Б.И Ващенко

Обычный бумажный документ имеет две составляющие – материальный бумажный носитель и зафиксированную на нем информацию. Бумажная основа связывает основную часть документа с дополнительным фрагментом, обычно называемым набором реквизитов документа. Бумажная основа документа и его реквизиты служат для доказательства того факта, что основная часть информации документа создана и зафиксирована на носителе тем лицом, от чьего имени документ получен или кем подписан, а также свидетельством того, что документ не изменен неуполномоченным на то лицом.

В поведении человека, определяющего свое отношение к бумажному документу, можно выделить две составляющие. Во-первых, его решение об отношении к документу на основе совокупности критериев его подлинности и неизменности. Документ на бумажном носителе вызывает подозрения на подложность, если, к примеру, сам носитель поврежден, текст документа содержит исправления, смазанные печати, подписи, отличающиеся от образцов и так далее. Наличие на документе «генетически» сформированной рукописной подписи человека обеспечивает решение проблемы подлинности

и юридической значимости документа, поскольку она обладает следующими важнейшими свойствами:

- подпись достоверна. Она убеждает получателя в том, что подписавший сознательно подписал документ;
- подпись неподдельна. Она доказывает, что именно подписавший, и никто иной, сознательно подписал документ;
- подпись не может быть использована повторно. Она является частью документа, мошенник не сможет перенести подпись на другой документ;
- подписанный документ нельзя изменить. После того, как документ подписан, его невозможно изменить;
- от подписи невозможно отречься. Подпись и документ имеют материальную основу. Подписавший не сможет впоследствии утверждать, что он не подписывал документ.

Процесс принятия решения о доверии документу на основании целостности носителя и не вызывающих сомнения реквизитов, а, следовательно, факту исполнения документа именно тем лицом, от которого он получен, есть первый этап восприятия документа человеком, называемый иногда аутентификацией документа. На практике мы настолько привыкли к этой процедуре, что выполняем ее автоматически, не задумываясь о месте, роли и значении выполняемых действий в процессе работы с документом.

Корректность восприятия документов достигается совокупностью следующих факторов:

- недвусмысленность толкования текста, изложенного общепонятным языком;
- применение метрической системы мер с принятыми сокращениями и кодировками;
- устоявшаяся терминология;
- действующее законодательство;
- применение государственных, межведомственных гаи иных стандартов, определяющих формы документов и их содержание.

В отличие от бумажных документов, история развития электронных документов исчисляется несколькими десятками лет. За точку отсчета можно принять всего лишь семидесятые годы прошлого столетия. В настоящее

время электронные документы стали неотъемлемой частью нашей жизни в связи с широким распространением современных информационных технологий обработки и передачи данных.

В 1976 году в открытой печати появились первые работы Диффи и Хеллмана, применяющие односторонние функции для решения так называемой задачи открытого распределения ключей. На основе этих материалов в дальнейшем был разработан механизм электронной цифровой подписи (ЭЦП), заменяющий традиционную рукописную подпись и другие защитные атрибуты бумажных документов.

ЭЦП представляет собой блок дополнительных данных (реквизит), приписываемый к защищаемым данным электронного документа (ЭД). ЭЦП зависит от содержания подписываемого ЭД и уникального секретного элемента (ключа), которым обладает только один участник защищенного обмена электронными документами.

Если провести параллель между технологиями восприятия бумажного и электронного документа (ЭД), то аутентификация (проверка подлинности) ЭД осуществляется посредством проверки ЭЦП. При проверке ЭЦП файла проверяется, применялся ли при выработке данной цифровой подписи конкретный ключ, принадлежащий отправителю документа, и не претерпел ли файл изменений в процессе пересылки адресату. Если программа проверки подписи формирует запись «ЭЦП верна», то ЭД «аутентифицирован» (то есть является подлинным). Для последующего восприятия ЭД требуется механизм перевода бинарной информации, составляющей файл ЭД, в читаемую человеком форму, определенным образом трактующего содержимое данной формы. Необходимость подобного механизма накладывает определенные требования на структуру файла ЭД. Между файлом ЭД и человеком всегда присутствуют две программы. Одна из них проверяет ЭЦП и сообщает результат проверки, либо отображая его на экране, либо распечатывая на бумаге в виде протокола проверки подписи. Другая программа либо отображает файл ЭД на экране, либо печатает на бумаге. Стороны, доверяя программе формирования ЭЦП и ее проверки, становятся заложниками данной программы. Для исключения рисков некорректной работы этой программы применяется механизм сертификации программного обеспечения. Изготовитель программы вправе, обратившись в государственные органы, заказать проведение экспертизы на предмет

выявления соответствия или несоответствия алгоритмам хеширования и ЭЦП, имеющим статус ГОСТов. В случае соответствия ГОСТам и удовлетворения дополнительным критериям на ПО, выдается сертификат. Целостность ПО и его соответствие предъявленному на экспертизу эталону обеспечивается подписыванием ЭЦП разработчика исполняемых файлов. Если обеспечение достаточного доверия программе формирования и проверки ЭЦП вполне возможно, например, способом сертификации, то, что касается доверия программам отображения файлов в однозначно понимаемые человеком формы, его достижение требует иного подхода.

Экранная и печатная формы документа могут быть различными, так как они обусловлены удобством работы с ними. Как правило, печатная форма совпадает с видом бумажного аналога. Вместе с тем, любой файл можно отобразить на бумаге как последовательность символов простейшего кода, например, шестнадцатеричного, двоичного или ASCII-кода. Очевидно, что программа отображения файла в понятную форму, по сути, является переводчиком, который должен быть простым, понятным и ответственным. К сожалению, выбрать «ответственный» программный переводчик или сделать его таковым практически невозможно. Следовательно, представляется единственно правильным вопрос доверия переводчику решать путем уменьшения вплоть до исключения его роли в разнообраз1ш представлений файла в понятном человеку виде. На первый взгляд, с текстовым редактором мы ничего сделать не можем. Однако, это неверно. Мы можем минимизировать его востребованную функциональность путем максимального упрощения структуры файла. Поясним это на следующем примере. Известна терминология – «язык программирования низкого уровня» и «язык программирования высокого уровня». На бытовом уровне понимания различий одна команда языка программирования низкого уровня соответствует одной операции, одна же команда языка программирования высокого уровня соответствует целой совокупности операций. По аналогии с языками программирования введем терминологию – «файл низкого уровня» и «файл высокого уровня». Под «файлом низкого уровня» понимается набор символов (байт) в зафиксированной кодировке, который:

- во-первых, будучи распечатанным, на бумажном носителе или отображенным на экране произвольным редактором, воспринимается человеком однозначно;

- во-вторых, правило восприятия его смысла описано настолько просто, что позволит его легко понять постороннему лицу, а именно арбитру при разрешении возможных конфликтов.

В ряде случаев существует некоторая спорность в возможности однозначного отображения файла произвольным редактором. Однако при определении круга применяемых средств отображения цель становится достижимой. Отнесем к «файлам высокого уровня», такие файлы, как *.doc, *.rtf, *.xls, *.html, *.pdf. Правила описания их смысла для восприятия человеком настолько сложны, что арбитраж по поводу конфликтных документов представляется невозможным. Механизм же фиксирования эталонов программного обеспечения отображения подобных файлов на экране и на бумаге достаточно сложен для его практической реализации. Следует отметить, что с нестабильной работой компьютера (нечитаемость текстов, зависания, подверженность вирусам при чтении «файлов высокого уровня») сталкивался практически каждый пользователь. Таким образом, возможность двоякого прочтения «файлов высокого уровня» вполне вероятна не столько по вине злоумышленника, сколько из-за нештатной работы компьютера. Итак, напрашивается вывод: электронным документом может быть исключительно «файл низкого уровня». Мы не станем перечислять типы файлов, подходящих на эту роль, так как мы и не задавались такой целью. Принцип же отнесения той или иной структуры файла к допустимой для применения в технологии электронного документооборота вполне сформулирован. В некоторой степени подходящим на роль «файла низкого уровня» является текстовый файл, содержащий символы оговоренной кодировки и фиксированные разделители полей. Однако даже в этом случае отсутствие соглашений относительно допустимых редакторов просмотра сообщений может иногда вызывать казусы.

ЭЦП: электронный документ (документ в электронно-цифровой форме отображения) – информация, представленная в форме набора состояний элементов электронной вычислительной техники, иных электронных средств обработки, хранения и передачи информации, способная быть преобразованной в форме, пригодной для однозначного восприятия человеком, и имеющей атрибуты (реквизиты) идентификации (и возможно аутентификации) документа.

Важнейшими реквизитами любого ЭД являются:

- временная метка создания (или подписания) документа;
- номер или идентификатор документа;
- ЭЦП документа.

ЭЦП – аналог собственноручной подписи для придания электронному документу юридической силы, равной бумажному документу, подписанного собственноручной подписью правомочного лица и/или скрепленного печатью. ЭЦП обеспечивает проверку целостности документов, конфиденциальность, установление лица, отправившего документ. Это позволяет усовершенствовать процедуру подготовки, доставки, учета и хранения документов, гарантировать их достоверность.

Использование ЭЦП основывается на следующих свойствах:

- подпись аутентична, то есть с ее помощью получателю документа можно доказать, что она принадлежит подписывающему (на практике это определяется графологической экспертизой);
- подпись неподделываема, то есть служит доказательством, что только тот человек, чей автограф стоит на документе, мог подписать данный документ, и никто другой не смог бы этого сделать;
- подпись непереносима, то есть является частью документа и поэтому перенести ее на другой документ невозможно;
- документ с подписью является неизменяемым, то есть после подписания его невозможно изменить, оставив данный факт незамеченным;
- подпись неоспорима, то есть человек, подписавший документ, в случае признания экспертизой, что именно он засвидетельствовал данный документ, не может оспорить факт подписания;
- любое лицо, имеющее образец подписи, может, удостовериться в том, что данный документ подписан владельцем подписи.

ЭЦП является средством защиты информации, обеспечивающим возможность контроля целостности и подтверждения подлинности электронного документа. При этом ЭЦП должна обеспечивать выполнение следующих функций.

- Во-первых, подтверждать, что подписывающее лицо не случайно подписало электронный документ (ЭД).

- Во-вторых, ЭЦП должна подтверждать, что только подписывающее лицо, и только оно, подписало электронный документ.
- В-третьих, ЭЦП должна зависеть от содержания подписываемого документа и времени его подписания.
- В-четвертых, подписывающее лицо не должно иметь возможности впоследствии отказаться от факта подписи документа.

Литература:

1. ГОСТ Р 34.10-2001 Информационная технология. Криптографическая защита информации.
2. ГОСТ Р 34.10-94 Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма.
3. http://www.security.ase.md/publ/ru/pubru86/ Анализ алгоритмов электронной цифровой подписи.
4. http://ru.wikipedia.org/wiki Электронный документ.

УДК 681.527.5

АВТОМАТИЗИРОВАННАЯ ОПТИКО-ЭЛЕКТРОННАЯ СИСТЕМА ДЕТЕКТИРОВАНИЯ И ИДЕНТИФИКАЦИИ ЗАГОТОВОК НА МЕТАЛЛУРГИЧЕСКОМ ПРОИЗВОДСТВЕ

Э.С. Литвак, Ю.Н. Литвак, А.Ю.Фоменко

Одним из путей увеличения экономической эффективности производства и повышения качества производимой продукции является автоматизация производственных процессов. Автоматизация не только упрощает сам процесс производства, снижает нагрузку на рабочих, исключает человеческий фактор, а также позволяет в режиме реального времени осуществлять контроль качества отдельных операций, составляющих производственный процесс. Это позволяет соответствовать международным стандартам систем менеджмента качества, сертификаты соответствия которым являются необходимым условием для вывода продукции компаний на мировые рынки. Как правило, для осуществления

контроля качества продукции на промежуточных стадиях производства, необходимо приостанавливать процесс, дополнительно оснащать линию, включая в нее необходимые точки контроля и комплектовать производство дополнительными кадрами. Современные технологии позволяют встраивать внешние по отношению к производственному процессу системы, выполняющие функции контроля, которые могут определить изделия, не удовлетворяющие техническим требованиям, до поступления их на следующий этап производства.

Рис. 1. Цех по производству слябов

В данной работе рассмотрено разработанное решение одной из таких задач.

На Магнитогорском металлургическом комбинате контроль качества продукции (соответствие состава стали отливок определенным маркам стали) проводится после поступления ее на склад. При отсутствии контроля за логистикой в цехе и при обнаружении брака на складе необходимо либо маркировать весь объем выпущенной за день продукции, как сталь марки ниже, либо проводить контроль каждой отливки, что практически не возможно.

Проблема возникает на линии производства слябов – полупродуктов металлургического производства, представляющих собой стальную плиту прямоугольного сечения с большим отношением ширины к высоте. Слябы

формируются из расплавленного металла, доставляемого порционно на линию конвейера. Деление на порции производится газо-плазменной резкой непрерывного потока, вытекающего на линию, двигающуюся с определенной скоростью (рисунок 2).

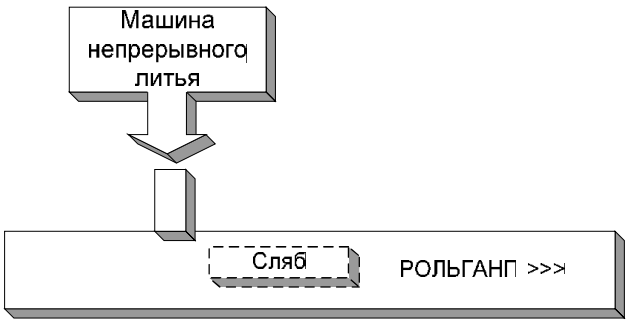


Рис. 2. Схема производства слябов на одной из линий производства

Производство состоит 16 одинаковых линий (ручьев), слябы которых необходимо уметь различать, так как в итоге они поступают на выходной контроль единой очередью. Важно то, что брак сляба одной из линий указывает на брак других слябов той же линии, поскольку брак определяется составом расплава стали. В итоге на выходе необходимо уметь выделять слябы каждой конкретной линии.

Необходимо было предложить техническую систему, автоматически выполняющую функции контроля логистики слябов внутри цеха путем сбора информации о слябе на выходе из машины непрерывного литья заготовок, а после – их идентификации на входе склада.

Идея системы и реализация аппаратной части

Для устранения проблемы предложено следующее решение: на этапе движения по рольгангу необходимо маркировать или детектировать каждый из слябов по некоторому характерному только для него признаку – создавать образ, а на этапе разбора образцов – распознавать текущий образ сравнением с образами, накопленными в базе данных.

Необходимым условием правильной работы системы является уникальность создаваемых образов слябов. Формировать образ решено было на основании характерного следа на торцевой стенке сляба, который

образуется при отрезании сляба автоматизированным газовым резаком на выходе машины непрерывного литья заготовок.

Функциональная схема устройства детектирования имеет следующий вид (рисунок 3):

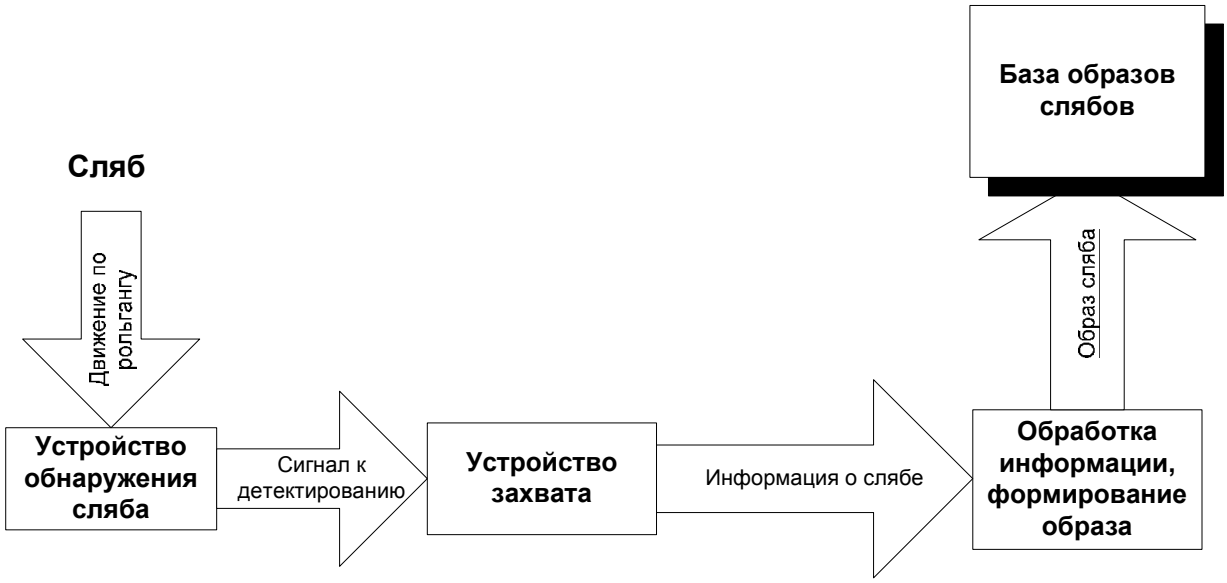


Рис. 3. Функциональная схема устройства детектирования сляба

Соответственно на выходе цеха стоит подобное устройство, которое не только получает образ текущего сляба, но и сравнивает его с образами слябов из базы данных (рисунок 4):

Рис. 4. Функциональная схема устройства идентификации сляба

Для обнаружения сляба используется оптопара. Устройством захвата изображения является фотоаппарат, управляемый сигналом от оптопары о прохождении сляба по рольгангу. Фотоаппарат соединен с компьютером, на котором программа с заданной частотой проверяет появления новых снимков. Появившийся снимок обрабатывается по специальному алгоритму, позволяющему для сляба создать уникальный характеристический массив значений, который и является образом сляба. Далее образ сравнивается с базой данных либо добавляется в нее.

Следует отметить, что для получения пригодных для создания образов снимков следует организовать определенные условия освещенности объекта.

Задачи программной части

Задачи программной части заключаются в выделении на фотографии сляба области интереса – торцевой грани, ее обработке и преобразовании к оговоренному формату.

Входное изображение сляба проходит ряд преобразований, принципиальными из которых являются:

- перевод в черно-белую цветовую гамму;
- фильтрация высокочастотных помех (одиночные или сгруппированные в зону выбросы яркости, суммарная площадь которых составляет менее половины точек окна 3x3) медианным фильтром;
- повышение резкости изображения.

Выделение границ торца сляба возможно несколькими способами. В результате имеем координаты интересуемого четырехугольника. Поскольку на реальных слябах волнистости, оставленные газовой резкой, имеют преимущественно вертикальное направление, в качестве характеристического массива сляба берутся средние значения яркости в столбцах точек вдоль плоскости основания сляба в полосе 0,5h±5%, т.е. используется полоса шириной в 10% от высоты сляба. Такой способ прост в реализации, а так же позволяет компенсировать неточность определения горизонтальных границ торца сляба при этом дает достаточно информации для идентификации слябов.

В ходе работы программы идентификации путем сравнения вновь полученного образа сляба на выходе из цеха и образов из базы данных вычисляется их корреляция, и производится выделение образов с

коэффициентами корреляции больше порогового значения, а также с максимальным значением.

Анализ работы программы показал, что значение коэффициента корреляции между различными образами одного и того же объекта существенно выше (от 3 до 10 раз) по сравнению с коэффициентом корреляции для образов разных объектов, что позволяет говорить о возможности распознавания слябов или других объектов обработки по их фотографиям.

В дополнение следует отметить, что данная система позволяет существенно снизить стоимость решения данной частной задачи путем замены нумерации слябов (привлечением дополнительных рабочих либо установкой дорогостоящего оборудования) внедрением решения, которое реализуется при помощи неспециализированного оборудования широкого потребления. Кроме того, такая компьютеризированная система помимо основной своей функции способна решать задачи учета путем ведения электронного журнала.

Литература:

1. *Гонсалес Р., Вудс Р.* Цифровая обработка изображений. М.: Техносфера, 2005. 1093

УДК 004.4'422

ОСОБЕННОСТИ КОМПИЛЯЦИИ ПРОГРАММ НА С ДЛЯ AVR И ARM МИКРОКОНТРОЛЛЕРОВ ФИРМЫ ATMEL

А.Ю. Маянц, В.Я. Хартов

Применяемые компиляторы

Микропроцессоры и микроконтроллеры разрабатываются множеством компаний во всём мире. Естественно, для успешного вывода на рынок любого устройства такого рода необходимо не только обеспечить его конкурентоспособные характеристики, но и подготовить весь спектр необходимой документации, а также аппаратных и программных продуктов поддержки. К последним, прежде всего, относится компилятор и прочие

средства разработки, поскольку без них невозможно создание программ для новой платформы.

Разработка компилятора – весьма трудоёмкий, ответственный и, следовательно, дорогой процесс, требующий привлечения высококвалифицированного персонала [1]. В то же время, под давлением рынка многие фирмы-разработчики процессоров и контроллеров вынуждены предоставлять средства разработки для своей продукции в открытый доступ, чтобы привлечь потенциальных клиентов и упростить её освоение.

Схожие цели преследуют и производители специализированных средств разработки программного обеспечения. Их продукты должны обеспечивать, помимо собственно компиляции кода для одной или нескольких выбранных платформ, широкий диапазон сервисных. Поэтому разработчики стремятся упростить наиболее сложный и дорогой этап проектирования своих систем.

Перечисленный соображения приводят к тому, что широкий ряд компаний стремится применить при разработке своих компиляторов общедоступные программные компоненты. Наиболее широко распространённым набором программ такого рода является GCC – GNU Compiler Collection. Его применение позволяет разработчикам аппаратуры создавать не весь компилятор, а лишь генератор кода для конкретной платформы. С другой стороны, лицензирование на условиях GPL заставляет разработчиков выпускать свои модули также под свободной лицензией, что способствует развитию свободного программного обеспечения.

Архитектура компилятора GCC

Аббревиатура GCC расшифровывается как GNU Compiler Collection – коллекция компиляторов GNU. Как следует уже из названия, GCC поддерживает большое число языков программирования, а также целевых архитектур. Это возможно благодаря его модульной, допускающей расширение архитектуре.

Процесс компиляции в GCC разделяется на 3 стадии. На первой производится синтаксический анализ входной программы и строится её представление в виде абстрактного синтаксического дерева. Синтаксические деревья позволяют представить код на любом языке программирования в

универсальной форме, что позволяет упростить последующий процесс анализа и оптимизации программы.

Представление дерева на первом этапе является промежуточным и носит название GENERIC. На втором этапе оно преобразуется в упрощённое представление GIMPLE, после чего выполняются различные процедуры оптимизации. В их число может входить упрощение арифметических выражений, свёртка констант, вывод инвариантного кода за пределы цикла и т. д. Затем полученная оптимизированная программа преобразуется в представление на языке регистровых пересылок (RTL – Register Transfer Language), который является своего рода обобщённым языком ассемблера. После этого выполняется ещё один этап оптимизации, и производится генерация машинного кода [2].

Схема процесса компиляции приведена ниже на рисунке.

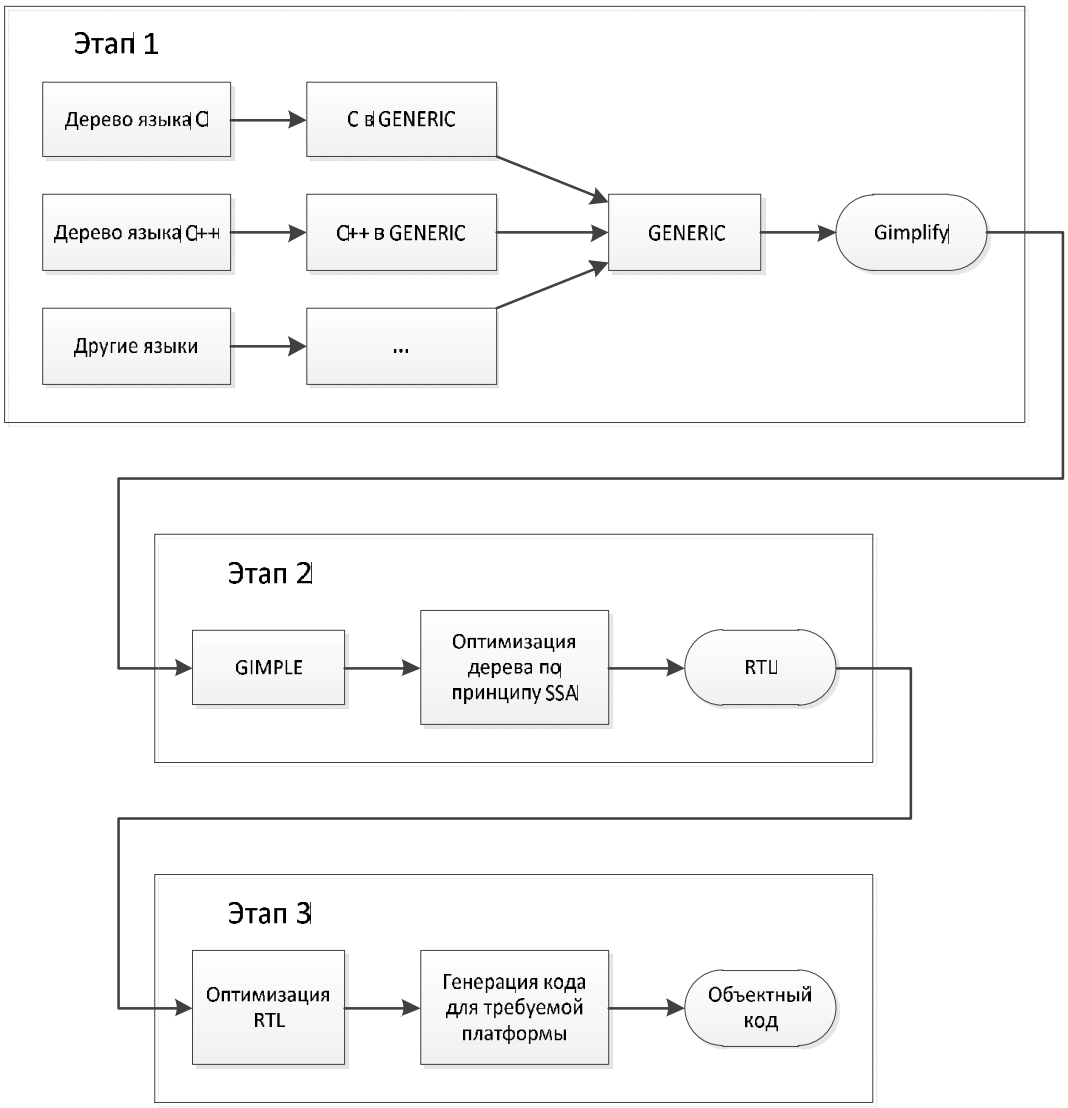


Рис. Процесс компиляции в GCC

Разработчикам решений для различных аппаратных платформ достаточно написать только свой генератор кода и, возможно, модифицировать генерацию регистрового кода для учёта всех особенностей целевой системы. Наиболее сложная часть работы – синтаксический разбор кода на языках программирования высокого уровня – выполняется существующими модулями компилятора GCC.

В простейшем случае команда для компиляции программы program.c в бинарный файл program.elf для архитектуры ARM будет выглядеть следующим образом:

```
arm-elf-gcc -o program.elf program.c
```

К наиболее часто используемым общим опциям компилятора можно отнести, помимо уже отмеченного указания выходного файла `-o`, следующие:

1) опции управления путями поиска файлов (`-I` для заголовочных файлов, `-L` для компоновщика, `-B` для файлов самого GCC);

2) опции оптимизации (`-O` (`-O0` — `-O3`, `-Os`) и `-f` с указанием конкретного приёма оптимизации);

3) опции компоновщика (`-l` (компонуемые библиотеки), `-Wl` (управление компоновщиком) и др.);

4) опции, зависящие от используемого языка программирования или целевой платформы;

5) другие опции, полное описание которых представлено в [3].

Компиляция программы в среде WinARM

Набор инструментов для разработки WinARM включает компилятор GCC, ассемблер, инструменты GNU для работы с двоичными файлами, а также ряд служебных и системных утилит. Запуск компилятор производится из командной строки, при этом есть возможность автоматического вызова компилятора из поддерживающих такую возможность программ с графическим интерфейсом, таких, например, как Programmers Notepad или Notepad++. Для корректной работы WinARM следует добавить путь к его исполняемым файлам в переменную окружения PATH.

Архитектура ARM реализована в широком ряде процессоров различных производителей, поэтому многие возможности компилятора GCC для ARM обусловлены их поддержкой. Основные опции для архитектуры ARM:

1) выбор версии целевой архитектуры (`-march=name`);
2) выбор целевого процессора (`-mcpu=name`, `-mtune=name`);
3) выбор выходного двоичного интерфейса (`-mabi=name`);
4) управление поддержкой операций с вещественными числами (`-mhard-float`, `-msoft-float`, `-mfloat-abi=name`, `-mfpu=name`);
5) управление поддержкой инструкций Thumb (16-разрядный набор инструкция для ARM, позволяющий снизить требуемое пространство памяти для кода) (`-mthumb-interwork`, `-mthumb`, `-mtpcs-frame`, `-mtpcs-leaf-frame`, `-mcallee-super-interworking`);
6) управление порядком байтов (`-mlittle-endian`, `-mbig-endian`, `-mwords-little-endian`);
7) ряд других опций, связанных с вызовом процедур, особенностями архитектуры ARM и отладкой.

Одной из особенностей RISC-архитектуры ARM является упрощение доступа к памяти. Во-первых, чтение и запись производятся только 32-разрядными словами, что следует учитывать при написании программ, так как использование более коротких переменных char и short приводит к необходимости преобразования типов, что увеличивает как размер, так и время выполнения программы. Во-вторых, доступ к памяти должен быть также выровнен по словам (по крайней мере, до версии ARMv6), из-за чего структуры могут иметь разный размер в зависимости от порядка следования их членов. При оптимизации использования памяти этот факт следует учитывать [4].

Для распределения памяти, в частности, размещения программы в памяти ROM или RAM, применяются специальные сценарии компоновщика GCC. Выбор сценария производится с помощью параметра `-T` *имя*. В состав WinARM входит ряд скриптов для различных платформ и примеры их

использования. При необходимости, в них можно внести требуемые изменения.

Компиляция программы в среде AVR Studio

AVR Studio – среда разработки для микроконтроллеров AVR, бесплатно распространяемая их разработчиком, компанией Atmel. Она позволяет писать программы как на ассемблере, так и на C, а также производить их отладку с эмуляцией реального контроллера и прошивку устройств при использовании совместимого программатора. Для компиляции программ на C используется компилятор GCC, что позволяет добавить к среде поддержку других языков.

Опции компилятора GCC для архитектуры AVR таковы:

1) выбор типа целевого микроконтроллера (`-mmcu=mcu`);
2) управление стеком (`-minit-stack=N`, `-mtiny-stack`);
3) управление прерываниями (`-mno-interrupts`, `-mno-tablejump`);
4) управление размером целочисленных переменных (`-mint8`);
5) вывод размера инструкций в ассемблерный файл (`-msize`);

Для доступа к памяти EEPROM микроконтроллеров AVR применяются функции из заголовочного файла `<avr/eeprom.h>`: `eeprom_read_byte`, `eeprom_write_byte`, `eeprom_read_word`, `eeprom_write_word`, `eeprom_read_block` и `eeprom_write_block`. Следует иметь в виду, что частое обновление EEPROM может привести к быстрому износу этой памяти.

Для помещения переменных в память ROM (совместно с кодом) и работы с ними, можно воспользоваться макроопределениями и функциями из заголовочного файла `<avr/pgmspace.h>`. В частности, строки, хранимые в ROM, задаются с помощью определённого в этом модуле типа `PGM_P`. Функции для работы с переменными в ROM подобны функциям стандартной библиотеки C, но имеют немодифицируемый аргумент соответствующего типа. Имена функций соответствуют стандартным функциям C с добавлением суффикса «`_P`», например, `strlen_P` или `memcpy_P`.

Ряд наиболее простых и дешёвых микроконтроллеров AVR серии ATtiny не имеет встроенной памяти данных, что затрудняет программирование на C, поскольку приводит к отсутствию возможности

использования стека. Работа с такими контроллерами всё же возможна, поскольку регистры AVR располагаются в адресном пространстве памяти. Данная методика описана в [6].

Можно отметить меньшее богатство опций при работе с архитектурой AVR. Это обусловлено, прежде всего, большей простотой этих 8-битных микроконтроллеров. С другой стороны, на них программирование производится «ближе к железу», что может позволить добиться высокой эффективности программ. Конечно, микроконтроллеры AVR уступают по своим возможностям и мощности 32-разрядным процессорам ARM, но в своём классе они обладают превосходными характеристиками.

Помимо AVR Studio, компилятор GCC для AVR применяется в частности в наборе инструментов WinAVR.

Использование make-файлов

В процессе работы над программой разработчику приходится многократно производить компиляцию программы. Поэтому ему полезен инструмент, позволяющий автоматизировать эту задачу. Это особенно важно при разработке крупных проектов или применении системы контроля версий несколькими разработчиками, когда желательно избежать рекомпиляции всего исходного кода и ограничиться только изменёнными модулями. Также подобный инструмент бывает полезен при распространении программ в виде исходных кодов, не требующем от пользователя досконального изучения применяемого языка программирования.

В качестве такого инструмента широко применяется утилита GNU Make. Она позволяет автоматизировать компиляцию исходного кода программ в соответствии с определённым заданным набором правил. При этом make автоматически определяет, какие из исходных файлов были обновлены, и перекомпилирует только то, что необходимо. Помимо собственно сборки программ, make может выполнять и другие схожие задачи, связанные с автоматическим обновлением одних файлов при изменении других.

Для работы make требуется так называемый make-файл (makefile), который описывает зависимости между файлами программы и содержит команды для обновления этих файлов. Как правило, исполняемый файл программы зависит от объектных файлов, которые, в свою очередь,

получаются в результате компиляции соответствующих файлов с исходными текстами.

Простейший make-файл состоит из набора правил следующего вида:

цель ... : пререквизит ...

команда

...

...

Пререквизитами цели могут являться другие цели, необходимые для достижения данной, или файлы. Целью по умолчанию является первая описанная в make-файле. Если какой-либо файл был изменён со времени последней компиляции, все зависящие от него правила выполняются повторно. Таким образом, make-файл представляет собой описание дерева зависимостей внутри компилируемого проекта [8].

Make-файл может также содержать переменные, которые применяются для упрощения его составления.

В больших проектах make-файлы могут оказаться достаточно трудоёмкими для ручного написания. Для автоматизации этого процесса могут применяться такие утилиты, как GNU Autoconf/Automake или более простые, например, MakeFile ARM. Среды разработки, такие как AVR Studio, самостоятельно генерируют make-файлы при запуске сборки проекта. Существуют также стандартизованные шаблоны Make-файлов, которые можно применять, следуя определённым соглашениям о структуре проекта, см. например [9].

В докладе детально рассмотрены примеры написания make-файлов для конкретных применений.

Заключение

В данной работе рассмотрены основные особенности архитектуры компилятора GCC и его применения при разработке программ GCC для различных платформ. Подробно рассмотрены вопросы компиляции программ для платформ ARM и AVR с использованием пакетов WinARM, WinAVR и AVR Studio. Также продемонстрированы возможности автоматизации процедур компиляции проектов с применением утилиты GNU Make.

GCC позволяет осуществлять компиляцию широкого ряда языков программирования для обширного списка платформ. Это, в частности, делает

его удобным для разработки приложений для микроконтроллеров, поскольку позволяет снизить затраты, связанные со сменой используемого устройства. При этом особенности функционирования программы, которые в случае работы на низком уровне тем или иным образом зависят от применяемого компилятора, останутся неизменными.

Литература:

1. Зуев И. Редкая профессия // PC Magazine/RE — 1997. — №10.
2. Novillo D. From Source to Binary: The Inner Workings of GCC // Red Hat Magazine — 2004. — №2.
3. GCC online documentation: A GNU Manual / Free Software Foundation, Inc.
   1988 — 2005.
4. C компилятор ARM / ООО «Фирма Фитон». 2010.
5. Редькин П. П. 32/16-битные микроконтроллеры ARM7 семейства AT91SAM7 фирмы Atmel. Руководство пользователя. М.: Издательский дом «Додэка-XXI», 2008. — 704 с.: ил.
6. Lightner B. D. Programming Atmel ATtiny15L Using AVR-GCC. 2007.
7. Программирование на языке С для AVR и PIC микроконтроллеров. / Сост. Ю. А. Шпак — К.: «МК- Пресс». 2006. — 400 с.: ил.
8. The GNU Make Manual / Free Software Foundation, Inc. 2010.
9. AVR Project Organization: Standardized AVR Makefile Template / Psychogenic Inc. 2003.

УДК 681.3.07

ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

Мин Тхет Тин, А.В. Брешенков

Разработанная в начале 80-х годов прошлого века доменная система имён Интернета (DNS) можно сказать лишена каких бы то ни было механизмов обеспечения информационной безопасности. Вся современная DNS работает на доверии сторон, пользующихся системой, друг другу. Вероятно, два десятка лет назад доверие в качестве фундамента системы ещё

как-то можно было оправдать. В современном же Интернете, насыщенном разнообразнейшими хакерскими группировками, вооружёнными богатым инструментарием из всевозможных атак, доверие может быть полезно лишь в том случае, если оно подкреплено надёжными мерами информационной безопасности.

Электронная цифровая подпись (ЭЦП) – реквизит электронного документа, предназначенный для защиты данного электронного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа электронной цифровой подписи и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе, а также обеспечивает неотказуемость подписавшегося.

Схема электронной подписи обычно включает в себя:

- алгоритм генерации ключевых пар пользователя;
- функцию вычисления подписи;
- функцию проверки подписи.

Цифровая подпись обеспечивает:

- Удостоверение источника документа. В зависимости от деталей определения документа могут быть подписаны такие поля, как «автор», «внесённые изменения», «метка времени» и т. д.
- Защиту от изменений документа. При любом случайном или преднамеренном изменении документа (или подписи) изменится хэш, следовательно, подпись станет недействительной.
- Невозможность отказа от авторства. Так как создать корректную подпись можно, лишь зная закрытый ключ, а он известен только владельцу, то владелец не может отказаться от своей подписи под документом.
- Предприятиям и коммерческим организациям сдачу финансовой отчетности в государственные учреждения в электронном виде;
- Организацию юридически значимого электронного документооборота.

Схемотично создание и проверка цифровой подписи показаны на рисунке.

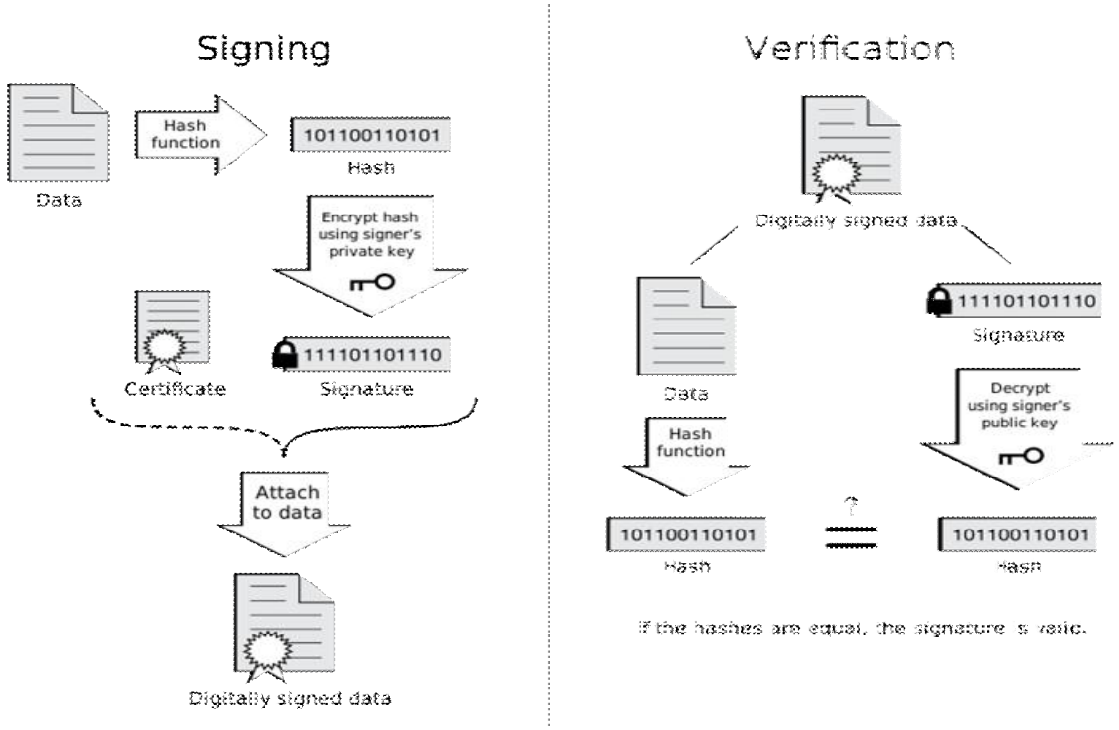


Рис. Цифровая подпись: создание и проверка

Возможные атаки на ЭЦП таковы:

Подделка подписи. Получение фальшивой подписи без наличия закрытого ключа – задача практически нерешаемая даже для очень слабых шифров и хэшей.

Подделка документа (коллизия первого рода). Злоумышленник может попытаться подобрать документ к данной подписи, чтобы подпись к нему подходила. Однако в подавляющем большинстве случаев такой документ может быть только один.

Получение двух документов с одинаковой подписью (коллизия второго рода). Куда более вероятна атака второго рода. В этом случае злоумышленник фабрикует два документа с одинаковой подписью, и в нужный момент подменяет один другим. При использовании надёжной хэш-функции такая атака должна быть также вычислительно сложной.

Литература:

1. Система Доменных Имён(DNS). http://ru.wikipedia.org/wiki/DNS
2. Domain Name System. http://en.wikipedia.org/wiki/Domain_Name_System
3. Сравнение программного обеспечения сервера DNS. http://en.wikipedia.org/wiki/Comparison_of_DNS_server_software

4. Ramaswamy Chandramouli, Scott Rose. Secure Domain Name System (DNS) Deployment Guide. NIST Special Publication 800-81,2006

УДК 004.658

ВНЕДРЕНИЕ ОБЪЕКТНЫХ БАЗ ДАННЫХ ДЛЯ WEB-ПРИЛОЖЕНИЙ

Ю.Ю. Митрушенков, А.В. Брешенков

При создании автоматизированных систем управления, успех в большой степени зависит от правильного выбора системы управления базами данных (СУБД) – платформы любой информационной системы.

Существует типичный набор проблем, связанных с СУБД: недостаточная производительность приложений, сложности при построении распределенных систем, трудности моделирования данных, скорость и простота разработки Web-приложений.

Широко используемые в настоящее время реляционные СУБД (РСУБД) хоть и остаются довольно популярными среди разработчиков, но проблемы и недостатки, связанные с СУБД заставляют вводить новые методики, изменения и дополнения к существующим методам построения БД. И в данном случае следует воспользоваться, так называемым, «объектным» подходом.

Объектные базы данных (так же известные как объектно-ориентированные базы данных) – это модель базы данных, в которой информация представляется в форме объектов, аналогичных объектам в объектно-ориентированном программировании. Результатом совмещения возможностей баз данных и возможностей объектно-ориентированных языков программирования являются объектно-ориентированные системы управления базами данных (ООСУБД). ООСУБД расширяет языки программирования, прозрачно вводя долговременные данные, управление параллелизмом, восстановление данных, ассоциированные запросы и другие возможности.

Некоторые объектно-ориентированные базы данных разработаны для плотного взаимодействия с такими объектно-ориентированными языками программирования как Python,Java, C#, Visual Basic .NET, C++, Objective-

C и Smalltalk; другие имеют свои собственные языки программирования. ООСУБД используют точно такую же модель, что и объектно-ориентированные языки программирования.

Объектно-ориентированные базы данных обычно рекомендованы для тех случаев, когда требуется высокопроизводительная обработка данных, имеющих сложную структуру. Именно поэтому создавая сложные Web-приложения, администраторы информационных систем начинают взаимодействовать с объектно-ориентированными базам данных.

Одним из фундаментальных понятий объектно-ориентированного подхода является понятие объекта. В общих случаях, объектные базы данных предназначены для эффективного хранения, управления и организации доступности со сложными типами данных, которые, собственно, и составляют основу Web-приложений, – документами, изображениями, аудио и видео.

Объектная технология и объектные базы данных являются практическим результатом работы по моделированию деятельности мозга. Мозг может, с одной стороны, хранить очень сложную и разнородную информацию, а с другой, – манипулировать ею, используя единые подходы. Точно также, сложное поведение объектов реального мира должно быть реализовано в программах таким образом, чтобы скрыть эту сложность.

Существуют различные способы использования объектных баз данных с Web-приложениями. При этом можно отметить преимущества объектного подхода, которые делают его все более популярным для разработчиков:

- возможность разбить систему на совокупность независимых сущностей-объектов и провести их строгую независимую спецификацию;
- простота эволюции системы за счет использования таких элементов объектного подхода, как наследование и полиморфизм;
- возможность объектного моделирования системы, позволяющего проследить поведение реальных сущностей предметной области уже на ранних стадиях разработки;
- возможность в полной мере использовать выразительные возможности объектных и объектно-ориентированных языков;
- повышение уровня унификации разработки и пригодность для повторного использования не только программ, но и проектов;

- возможность построения систем на основе стабильных промежуточных описаний, что упрощает процесс внесения изменений (это дает возможность системе развиваться постепенно и не приводит к полной ее переработке даже в случае существенного изменения исходных требований);
- ориентация объектной модели на человеческое восприятие мира.

Клиент-серверная архитектура подразумевает, что как сервер СУБД, так и клиент имеют собственное локальное хранилище объектов. Иными словами, клиент помимо локальных хранилищ оперативных данных имеет полноценные персональные базы данных. Такой подход приводит к тому, что сервер выполняет меньшее число обращений к хранилищу данных, а клиент выполняет меньшее число обращений к серверу по сравнительно медленному сетевому протоколу.

Существуют различные способы использования объектных баз данных с Web-приложениями. Например, это может быть Web-узел, ядром которого служит объектная база данных**. Многие** компании могут использовать объектные базы данных для создания сложных моделей обработки данных и предоставления их для просмотра через Web-браузеры.

Почему объектные технологии все чаще выбираются для разработки новых Web-приложений баз данных? Дело в том, что с их помощью разработка комплексных приложений идет гораздо быстрее, и дальнейшие модификации происходят намного легче, т.е.:

- структура данных объекта более емкая, за счет чего можно естественным образом описывать предметы и события реального мира. При этом упрощается задача взаимопонимания между заказчиком и разработчиком Web-приложений;
- программирование упрощается, т.к. легче следить за самим процессом разработки и за спецификациями программ;
- принцип «черного ящика» позволяет совершенствовать внутреннее устройство объекта, не нарушая работу других частей Web-приложения;
- объекты упрощают взаимодействие с другими Web-технологиями и приложениями;
- объектная технология прекрасно сочетается с Java, что облегчает Web-разработку.

Java является объектно-ориентированным языком программирования со строгим контролем типов, который интересен, прежде всего, возможностью разработки приложений Web. Это простой, объектно-ориентированный, распределенный, интерпретируемый, устойчивый, безопасный, не зависящий от архитектуры, переносимый, высокопроизводительный и динамичный язык.

Взаимодействие между СУБД и приложением происходит с помощью стандарта Java DataBase Connectivity (JDBC) – соединение с базами данных на Java. JDBC определяет интерфейс доступа к базе данных, поддерживающий основные функции языка SQL и позволяющий осуществить доступ к широкому кругу СУБД. Благодаря пакету JDBC язык Java может использоваться в качестве базового языка программирования приложений баз данных.

Объектные базы данных в будущем станут основным элементом клиентской части Web-приложений и предоставят настольным системам возможность локальной обработки с большей степенью интерактивности, чем при работе с помощью обычных HTML-браузеров.

Литература:

1. Кирстен В., Ирингер И., Рёриг Б., Шульте П. СУБД Caché: объектно-ориентированная разработка приложений. – СПб,«Питер», 2001.

2. Data Integration Glossary, U.S. Department of Transportation, August 2001.

3. O'Brien, J. A., & Marakas, G. M. (2009). Management Information Systems (9th ed.). New York, NY: McGraw-Hill/Irwin.

4. Bancilhon, Francois; Delobel,Claude; and Kanellakis, Paris. Building an Object-Oriented Database System: The Story of O_2. Morgan Kaufmann Publishers, 1992. ISBN 1-55860-169-4.

5. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2003. — 1440 с. ISBN 5-8459-0527-3

УДК 004.832.23

ПОЛИТИКО-ОРИЕНТИРОВАННОЕ ПЛАНИРОВАНИЕ В МНОГОДОМЕННОЙ ИЕРАРХИЧЕСКОЙ СРЕДЕ

П.Е. Николаев

Вступление

Среди всех существующих техник планирования HTN планирование (Hierarchical Task Network – Иерархическая Сеть Задач) наиболее широко используется для решения практических задач [1, 2]. В настоящей статье рассматривается задача HTN планирования в многодоменной иерархической среде, предлагается ряд решений для облегчения спецификации задач планирования и для увеличения эффективности их решения. Примером такой задачи планирования является задача составления совместных программ обучения между ВУЗами разных стран.

HTN планирование позволяет обеспечить большую эффективность планирования, по сравнению с другими планировщиками за счет использования предопределенного набора процедур решения задач (методов) в рассматриваемой среде. Исходная задача, так же, как и промежуточное решение, представляется с помощью сети задач $TN = <T,C>$, где T – множество задач, C – множество ограничений для задач, в том числе последовательной детализации имеющейся сети задач с помощью следующих операций детализации: декомпозиция задач (при применении методов) и детализация ограничений (принудительное выполнение ограничений). В [3] было показано, что для широкого круга задач HTN планирования наиболее эффективной стратегией выбора операции детализации является стратегия FAF (Fewest Alternatives First). В соответствии с этой стратегией на каждом шаге планирования выбирается операция детализации, имеющая минимальный коэффициент ветвления, что позволяет сократить пространство поиска для планировщика (сузить дерево поиска).

Выделяют следующие недостатки HTN планирования:

- сложность спецификации задачи планирования за счет использования процедурных проблемно-ориентированных знаний при планировании [1]. Одна задача планирования может специфицироваться несколькими экспертами;

- необходимость увеличения выразительности предусловий в задачах планирования [4];
- для некоторых планировщиков необходима повторная компиляция при переключении на новый тип задач [4].

Политико-ориентированное HTN планирование

Для частичного устранения этих недостатков предлагается использование политико-ориентированного управления в качестве дополнительного средства спецификации предусловий в схемах действий. При обращении к функции проверки политик запрос к точке принятия решения на основе политик (PDP) формируется на основе параметров оцениваемого действия и данных из текущего состояния планировщика. Так как семантика политик и предусловий действий идентична, то действие может быть использовано в плане, если PDP одобрила проверяемый запрос. Для формирования запрос оценки политик на основе состояния планировщика был предложен механизм отображения, который основывается на введении стандартной схемы именования атомов.

Политики обладают следующими преимуществами, позволяющими частично решить указанные недостатки HTN планирования:

- политики обладают свойством композициональности. Несколько политик могут быть объединены в единую политику с помощью операций композиции. Такой подход к спецификации политик упрощает их спецификацию, предоставляет возможность для совместного независимого определения политик.
- языки спецификации политик более выразительны, чем предусловия в HTN планировании.
- политики – это дескриптивные правила работы системы, которые могут меняться в процессе ее работы, не требуя ее повторной компиляции.

Многодоменная иерархическая среда

Основным элементом многодоменной иерархической среды (МИС) является дерево доменов ***G =<L, E>,*** представляющее организационную структуру системы (L_x^y – домен, (организационная единица системы), ***E*** – ребро дерева, показывает отношение вхождения: $E(L_{x1}^{y1}, L_{x2}^{y2}) = L_{x2}^{y2} \in L_{x1}^{y1}$). Каждый домен содержит политики, определенные для данной

организационной единицы и действующие во всех вложенных доменах. Домены-листья дерева ***G*** также содержат множество объектов (управляемые объекты, ***МО***). Каждое действие в плане применяется к одному управляемому объекту. Последовательность управляемых объектов, используемых в плане-решении, образует *трек* решения. Множество возможных решений задачи планирования определяется с помощью частичного трека $PT = < L_{x1}^{y1}, L_{x2}^{y2}, \dots, L_{xm}^{ym} >$, где L_{xi}^{yi} – домен, определяющий, что в позиции i трека может находиться только ***МО*** , входящие в домен L_{xi}^{yi}.

Для задачи составления совместных программ обучения доменами-листьями являются университеты, которые имеют набор программ обучения (управляемых объектов), а доменами верхних уровней – регионы, страны, объединения стран. Трек решения является совместной программной обучения, треком обучения студента.

Планирование в многодоменной иерархической среде

Основной сложностью планирования в МИС является выбор управляемых объектов, включаемых в решение, так как в системе существует большое количество однотипных управляемых объектов и на их согласованное использование в плане налагаются ограничения с помощью политик. В соответствии с принципом FAF, при большом количестве доступных управляемых объектов ***n(MO)***, их выбор при планировании должен быть отложен до момента, когда он будет необходим для продолжения планирования, так как ***n(MO)»n(MO), n(MO)»N(V)***, где ***n(MO), n(M), n(V)*** – количество возможных: управляемых объектов, применяемых методов ***(M)***, значений для переменной ***(V).*** Таким образом, оптимальная стратегия планирования для такой задачи – стратегия поздней фиксации управляемых объектов.

Используя свойства МИС, возможно увеличить эффективность планирования в ней, введя промежуточные шаги при инициализации управляемых объектов. Для этого применяется *принцип нисходящей оценки политик*. При этом в процессе планирования любая задача в сети задач представлена в виде картежа: $t = < T, L_x^y >$, где ***T*** – задача, которая специфицируется пользователем изначально при описании задачи планирования, L_x^y – домен, ограничивающий множество возможных ***МО*** для

этой задачи. В начале планирования все задачи определены в корне дерева доменов. При осуществлении декомпозиции задач с помощью методов домен задач не изменяется. Для управления доменами задач была введена дополнительная операция детализации: *детализация доменов*. При осуществлении детализации домена для задачи ***t*** осуществляется преобразование $t \rightarrow \left(t' \equiv < T, L_{x'}^{y'} >\right)$, домен для этой задачи заменяется доменом, входящим в него ($L_{x'}^{y'} \in L_x^y$), при этом должны выполняться ограничения на домены, определенные с помощью частичного трека. Каждый раз после осуществления детализации домена можно осуществить проверку политик для домена, присвоенного задаче. Если проверка политик заканчивается неудачно, то текущий план не имеет решения и должен быть отброшен. Проверка политик для доменов верхнего уровня позволяет отсечь большое количество управляемых объектов на ранних стадиях планирования без осуществления проверки всех политик для этих объектов, что позволяет увеличить эффективность планирования.

Операция детализации доменов была реализована в виде набора методов, изменяющих домены задач при выполнении определенных условий. Эти методы специфицированы в той же нотации, что и методы декомпозиции задач. В плане-решении задачи планирования все задачи должны быть связаны с конкретным управляемым объектом (не с доменом). При применении операции детализации доменов $R_D(TN,t_i)$ к сети задач, для одной задачи может быть несколько возможных производных доменов. В этом случае результатом применения операции детализации доменов является множество возможных сетей задач $\Gamma:=R_D(TN,t_i)$, $\Gamma=(TN_1, TN_2, \ldots, TN_n)$.

Для управления процессом планирования в МИС была разработана стратегия выбора операции детализации, основанная на стратегии FAF. Принятие решения о применении операции детализации доменов для определенной задачи осуществляется при выполнении следующих условий: предусловия хотя бы одного метода детализации доменов выполняются для этой задачи в сети задач ***TN*** (необходимое условие), коэффициент ветвления для этой операции детализации минимальный по сравнению с другими возможными операциями детализации (достаточное условие). Коэффициент ветвления для операции детализации доменов равен мощности множества. В

качестве вторичной стратегии для выбора операции детализации при равенстве коэффициентов ветвления используется веденный коэффициент детализации. Коэффициент детализации для каждой операции детализации показывает, насколько эта операция увеличивает меру точности проверки политик для сети задач. Так как в процессе планирования для каждой переменной, на основе которой формируется запрос проверки политик, может быть определен список возможных значений, или значение может быть не определено, то проверка политик на основании такого запроса является частичной. Мера точности проверки политик показывает, насколько точно можно выполнить проверку политик на основании такого запроса.

Заключение

В настоящей статье были приведены недостатки HTN планирования, для их частичного устранения, в дополнение к предусловиям действий, предложено использование языка спецификации политик. Была рассмотрена задача планирования в МИС, одним из примеров таких задач является задача составления совместной программы обучения в международной образовательной среде. Для увеличения эффективности планирования в МИС был разработан принцип нисходящей оценки политик, который реализован с помощью введения операции детализации доменов. Для управления процессом планирования была предложена стратегия выбора операции детализации, которая основана на FAF стратегии, а в качестве вторичного фактора использует коэффициент детализации политик.

Литература:

1. Nau D., Ghallab M., Traverso P. Automated Planning: Theory & Practice / San Francisco, CA: Morgan Kaufmann Publishers Inc., 2004. 635 c.
2. Boukerche H., Lounis K. A hierarchical task network planning approach for optimizing energy production // Information and Communication Technologies (ICTTA), 2006.
3. Nau D., Ilghami O., Kuter U., Murdock J. W., Wu D., Yaman F. Shop2: An HTN planning system // Journal of Artificial Intelligence Research. № 20. 2003. С. 379–404.
4. Tsuneto R., Nau D., Hendler J. Plan-refinement strategies and search-space size // European Conference in Planning, 1997. С. 414–426.

УДК 681.324

ПРЕДСТАВЛЕНИЕ ТРАНСЛЯЦИОННО-ЦИКЛИЧЕСКОГО ОБМЕНА В ВИДЕ ГРАФ-СХЕМЫ

А.Е. Павлов, Ю.М. Руденко

В этой статье рассматривается способ представления параллельных алгоритмов с помощью граф-схем [1]. Для таких схем представляет интерес рассмотрение стандартных методов обмена информацией в вычислительных системах (ВС), таких как:

- дифференцируемый;
- трансляционный;
- трансляционно-циклический;
- конвейерно-параллельный;
- коллекторный.

При этом имеется следующая статистика частоты использования этих методов обмена, представленная в таблице 1 [2].

Таблица 1.

Тип обмена	Частота использования, %
Дифференцируемый	2
Трансляционный	17
Трансляционно-циклический	40
Конвейерно-параллельный	34
Коллекторный	7

Как видно из таблицы, значительная доля использования методов коллективного обмена приходится на трансляционно-циклический обмен. Он реализует трансляцию информации из каждой ветви параллельного алгоритма во все остальные. Следовательно, если трансляционный обмен выполняется за 1 такт, то трансляционно-циклический – за ***n*** тактов. Один из способов представления трансляционно-циклического обмена в виде граф-схемы приведен на рисунке 1.

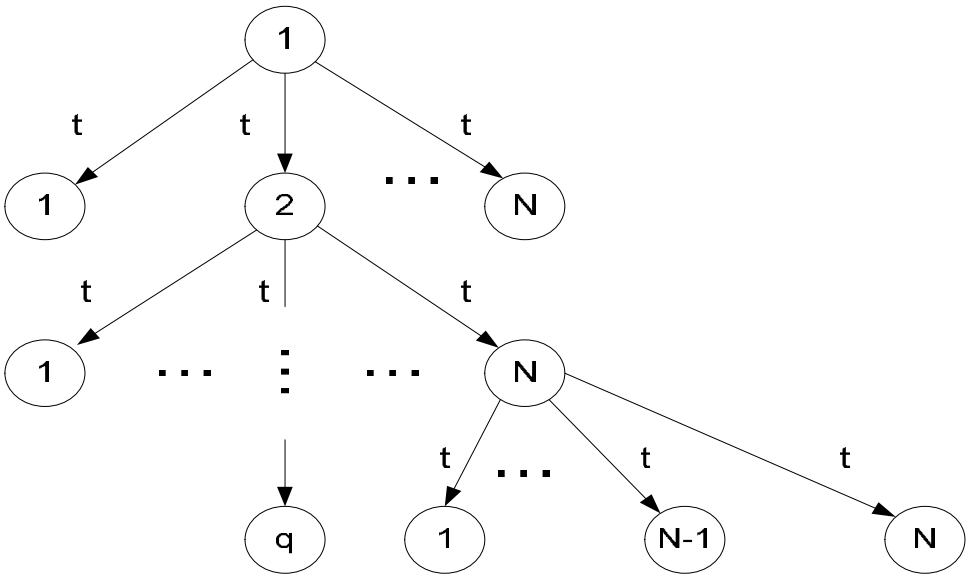


Рис. 1. Представление трансляционно-циклического обмена в виде граф-схемы

Как видно из рисунка 1, в случае большого количества процессоров и итераций передачи, при таком отображении, мы будем иметь громоздкую и ненаглядную граф-схему. В связи с этим часто возникает проблема представления данного типа обмена в виде граф-схемы.

В рассматриваемом случае время передачи информации во всех ветвях равно t. На n процессорах решается n^2 операторов. Построим диаграмму размещения операторов по вычислительным модулям (ВМ) по методике, описанной в [1], используя ранние и поздние сроки окончания выполнения операторов. Для определённости примем n=9 (рисунок 2).

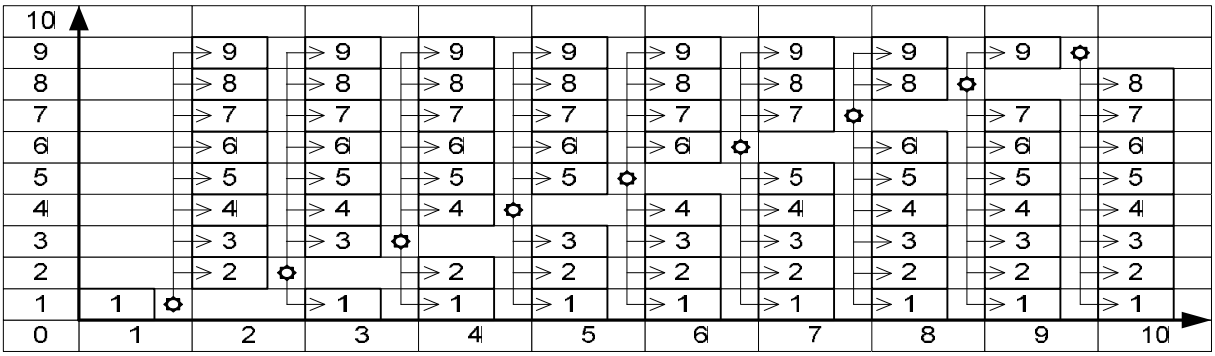


Рис. 2. Диаграмма размещения ВМ и операторов при организации трансляционно-циклического обмена. Символом « » обозначен процессор-передатчик информации

Основная сложность реализации такого обмена заключается в том, что должна быть обеспечена коммутация каждого из процессоров с каждым.

Впрочем, в данном примере ситуация несколько облегчается тем, что циклы обмена и обработки информации обладают заданной периодичностью.

Нетрудно заметить, что при большом количестве процессоров n в вычислительной системе возникают трудности с выбором коммутационной структуры и последующего расположения процессорных нитей по ВМ. Результат разложения представляется в виде матрицы расстояний [1] между ВМ, содержание которой показывает количество узлов, заключенных между анализируемыми ВМ.

Таким образом, основными требованиями к представлению трансляционно-циклического обмена в виде граф-схемы являются:

1. наглядность$
2. наращиваемость$
3. удобство перехода от граф-схемы к диаграмме размещения ВМ и операторов.

Под наглядностью далее мы будем понимать удобство восприятия граф-схемы, то есть отсутствие «нагромождённости» при отображении трансляционно-циклического обмена.

Требование наращиваемости подразумевает выбор такой структуры отображения трансляционно-циклического обмена, при которой будут однозначно определяться следующие его составляющие:

- ВМ-передатчик информации;
- ВМ-приёмники информации;
- Время передачи информации (здесь и далее будем полагать, что время передачи информации между ветвями t – одинаково для всех процессоров).

Кроме того, свойство наращиваемости подразумевает легкость объединения всех итераций трансляционно-циклического обмена в один замкнутый цикл с заданным порядком передачи «права на рассылку» между процессорами.

Одним из способов решения поставленной задачи является «упаковка» каждой итерации трансляционно-циклического обмена в некий универсальный блок, характеризующийся некоторыми параметрами, которые однозначно определяют наполнение этого блока. При этом в граф-схеме

такой блок мы будем обозначать специальным символом, а расшифровку структуры блока давать однократно.

Введём следующее обозначение одной итерации трансляционно-циклического обмена, представленное на рисунке 3.

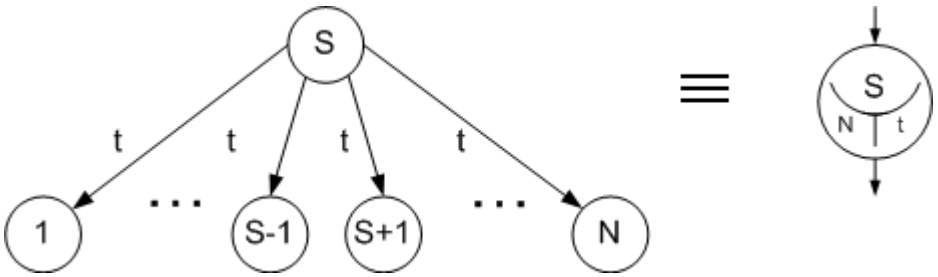


Рис. 3. Условное обозначение одной итерации трансляционно-циклического обмена. S – номер процессора-передатчика; N – количество процессоров, участвующих в обмене; t – время передачи информации между ветвями

Таким образом, используя введённое условное обозначение i-й итерации трансляционно-циклического обмена, весь цикл обмена можно представить в виде последовательности блоков-итераций. Построим граф-схему для ранее рассмотренного примера с числом процессоров n=9 и временем передачи информации между ветвями t.

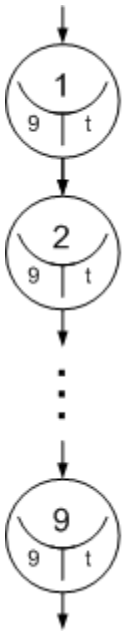


Рис. 4. Граф-схема трансляционно-циклического обмена, построенная на блоках-итерациях

Как видно, из рисунков 3 и 4, в общем случае, при построении граф-схемы трансляционно-циклического обмена, следует:

1. ввести символическое обозначение блока-итерации с указанием номера процессора-передатчика, количества процессоров, участвующих в обмене, и времени передачи между ветвями;

2. однократно представить детализированное отображение наполнения блока-итерации;

3. построить последовательную цепочку итераций в заданном порядке передачи «права на рассылку» между процессорами;

4. при необходимости указать рядом с символом блока-итерации время обработки информации, полученной процессорами-приёмниками от процессора-передатчика.

В заключение следует отметить, что новое представление трансляционно-циклического обмена является наглядным, наращиваемым и удобным для перехода от граф-схемы к диаграмме размещения ВМ и операторов. Используя новую структуру вместо старой, мы получим более компактную и легко читаемую граф-схему. При этом по-прежнему достаточно просто перейти к диаграмме размещения операторов и ВМ, выбрать коммутационную структуру ВС и построить таблицу расстояний. Кроме того, автоматизированное построение граф-схемы трансляционно-циклического процесса в новом представлении значительно легче, чем построение граф-схемы в виде n-мерных деревьев.

Литература:

1. Руденко Ю.М., Волкова Е.А. Вычислительные системы. Архитектура и составляющие ее компоненты. Представление параллельных алгоритмов и оптимизация времени их решения. Изд. МГТУ им. Баумана, Москва, 2010 г. 211 с. : ил.

2. Хорошевский В.Г. Архитектура вычислительных систем. Изд. МГТУ им. Баумана, Москва, 2005 г. 511 с. : ил.

УДК: 519.178

ПРИМЕНЕНИЕ ГРАФ-СХЕМ ДЛЯ ИЗОБРАЖЕНИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ

М.С. Красовский, Ю.М.Руденко

С развитием аппаратной части вычислительных систем, увеличением числа процессоров у все большего числа рабочих станций, проблема распределенных, параллельных вычислений становится острее. Несколько лет назад она была актуальна для вычислительных центров, обладавших соответствующим оборудованием, однако теперь каждый четвертый персональный компьютер в развитых странах имеет более одного процессора, что ставит задачу распределения операций во главу угла.

Теперь, при разработке программного обеспечения, использование последовательных алгоритмов является непроизводительным решением. Для использования аппаратных ресурсов в полной мере рациональным ходом считается использование параллельных алгоритмов, что влечет за собой задачу преобразования последовательных алгоритмов в параллельные. Разработчики таких программных пакетов должны четко понимать принцип преобразования, знать соответствующие методы. Для их лучшего понимания, освоения и представления используются граф-схемы.

Граф-схема алгоритма представляет собой некоторую последовательность свёрток – развёрток, образующих композицию из логических функций, зависящих от n переменных, где $n \in \{1,\ldots,\Psi\}$, где Ψ – максимальное количество дуг в свёртках и развёртках. Граф-схема может служить удобным средством для изображения параллельного алгоритма.

Граф-схемы алгоритмов представляются с помощью выражения G = (X, P, D),

X – множество вершин граф-схемы, $X \in \{1, \ldots, m\}$. Множество вершин графа X соответствует множеству операторов параллельного алгоритма.

P = {1,...,Pm} – множество весов вершин графа. Pi может быть скалярной величиной или вектором, $i \in \{1, \ldots, m\}$. Если Pi – скаляр, то рассматривается решение этой задачи на однородной ВС (вычислительная система имеет одинаковые процессоры). Тогда, как правило, Pi – время решения i-ого программного модуля. Если $\vec{P}_i$ – вектор, то предполагается решение этой задачи на неоднородной ВС (вычислительная система имеет

разные типы процессоров). И, тогда, если система содержит S разнотипных процессоров, то вектор $\vec{P}_i$ = { Pi1 ,… , Pis}, где Pi1 ,… , Pis – набор времен решения i-ой процедуры на различных типах процессоров из множества S.

D – множество дуг графов. Дуги бывают трёх типов: di ∈ D0, dj ∈ D2 , dk ∈ D3. D = D0∪D2∪D3. D0 – множество одиночных дуг графа, соответствующих элементарным свёрткам или развёрткам k-х вершин граф-схемы. D2 – множество логических дуг графа, реализующих функции «Исключающее ИЛИ» граф-схемы, D3 – множество дуг графа, реализующих функции «И» граф-схемы,. Обозначим D1= D0∪D3.

Однако можно заметить, что перечень типов дуг является неполным, если рассматривать его в разрезе алгоритмических языков. Дополним перечень типов дуг, введя dl ∈ D4, где D4 – множество логических дуг графа, реализующих функции «И-ИЛИ». Отличием в графическом начертании от D3 будет наличие «пустых кружков», вместо «полных».

Можно считать, что при изображении параллельных алгоритмов можно ограничиться обозначениями логических выходов операторов типа «IF», «CASE» в виде «№.n», где № – номер рассматриваемого логического оператора, n – номер выхода из логического оператора. Такое обозначение позволяет упорядочить существующие обозначения: «истина», «ложь», «FALSE», «TRUE», «>», «<» ,«<>» и т.д., что создает определённые удобства при дальнейшем анализе схем алгоритмов в виде граф–схем. Однако, после введения D4, такое определение является неполным.

Чтобы корректно отразить вариативность в схеме будем использовать обозначения вида «№.n v №.n», где «v» – знак дизъюнкции. Такая запись очень точно отражает реальные ситуации в алгоритмическом программировании.

Рассмотрим ситуацию описания конструкции типа «CASE»:

```
Case 1: Action1;
Break;
Case 2: Action2;
Break;
Case 3: Action3;
Break;
```

Здесь: при истинности первого условия выполняется Action1, при истинности второго условия выполняется Action2, при истинности третьего условия выполняется Action3.

Такая ситуация легко описывается с использованием D2. Ветви графа будут иметь обозначения 1.1, 1.2, 1.3 соответственно. Но внесем некоторые изменения:

Case 1: Action1;

Case 2: Action2;

Break;

Case 3: Action3;

Break;

Здесь: при истинности первого условия выполняется Action1, затем Action2, при истинности второго условия выполняется Action2, при истинности третьего условия выполняется Action3.

В данном случае алгоритм описывается с использованием D4, а ветви графа будут иметь обозначения 1.1, 1.1v1.2, 1.3 соответственно.

Приведенная в данной работе методика записи структуры алгоритмического программирования типа «И-ИЛИ» позволяет гибче использовать теорию графов для описания параллельных алгоритмов.

Литература:

1. Руденко Ю.М., Волкова Е.А. “Вычислительные системы”.-М.: НИИ РЛ МГТУ им. Н.Э.Баумана, 2010.-212 с.

2. Антонов А.С. «Параллельное программирование с использованием технологии MPI».-М.: Изд-во МГУ, 2004.-71 с.

3. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженера. – М.: Энергоатомиздат. 1988. – 478 с.

4. Водяхо А.И., Горнец Н.Н., Пузанков Д.В. Высокопроизводительные системы обработки данных. – М.: Высшая школа. 1997. – 150 с.

5. Корнеев В.В. Параллельные вычислительные системы. – М.: Наука. 1999. – 312 с.

6. Белоусов А.И., Ткачев С.Б. Дискретная математика: Учеб. для вузов / Под ред. В.С.Зарубина, А.П. Крищенко. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 774 с. (Сер. Математика в техническом университете; Вып. XIX).

УДК 004.272.26

УЧЁТ ВРЕМЁН ЗАДЕРЖЕК НА ВЫЧИСЛИТЕЛЬНЫХ МОДУЛЯХ ПРИ РЕАЛИЗАЦИИ ГРАФ-СХЕМ

Л.В. Мусина, Ю.М. Руденко

Высокая производительность вычислительных систем достигается благодаря методам, основанным на параллельной обработке информации. Вычислительная система содержит несколько процессоров или операционных устройств, способных одновременно, но с необходимой синхронизацией, выполнять доли возлагаемых на них работ по реализации алгоритма решения задачи.

Проблема её программирования – это проблема планирования распараллеливания, что требует разбиения алгоритма, в общем случае, на частично упорядоченное множество работ, назначения этих работ процессорам с учетом синхронизации их выполнения в соответствии с обязательным порядком следования.

Ориентированный граф, представляющий некоторую схему алгоритма, не содержащую циклов, т.о., что каждому блоку алгоритма соответствует вершина, а связям между блоками – дуги, называется граф-схемой алгоритма.

В общем случае, граф-схема алгоритма – это сеть. Циклы из граф-схем исключены. Это сделано по нескольким причинам. Во первых, распараллеливание осуществляется для сложных программ, где каждый блок схемы алгоритмов – программный модуль, в который всегда можно включить небольшие по времени циклы и, тем самым, упростить составление граф-схем. Во-вторых, циклы по параметру не поддаются распараллеливанию, и их бесполезно изображать на граф-схеме, а лучше включить в программный модуль. Циклы по счетчику циклов распараллеливаются введением дополнительных вершин в граф-схему. Сеть можно представить как совокупности сверток и разверток.

Сверткой k-й вершины граф-схемы называется наличие у k-й вершины граф-схемы n входящих дуг, где $n \geq 1$ и является конечным числом.

Разверткой k-й вершины граф-схемы называется наличие у k-й вершины граф-схемы n выходящих дуг, где $n \geq 1$ и является конечным числом (см. рисунок).

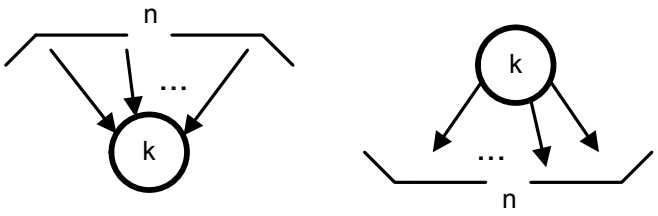


Рисунок. Свертка и развертка граф-схемы

Изолированной k-й вершиной граф-схемы называется вершина, у которой отсутствуют входящие и выходящие дуги.

Элементарной сверткой или разверткой k-й вершины граф-схемы называется наличие у k-й вершины граф-схемы одной входящей или выходящей дуги соответственно.

Вопрос о получении или передаче значений параметров для этих случаев представляет несомненный интерес при создании граф-схем алгоритмов, предназначенных для выполнения на вычислительных системах.

Рассмотрим случай свертки k-й вершины граф-схемы. Будем полагать, что каждый вход в k-ю вершину рассматривается как логическая переменная, которая принимает значение «true», если на рассматриваемый вход приходит информация, полученная на предшествующей вершине, в заданный интервал времени. В противном случае, логическая переменная принимает значение «false».

Интервал времени вычисляется соответствующим способом. Учитывая, что возможны некоторые непредвиденные отклонения, не влияющие существенно на результаты вычислений, при обработке и передаче информации с предшествующей i-й вершины в k-ю, введем интервал времени $[-\nabla t^*_{ik}, \nabla t^*_{ik}]$. Т.о., время прихода информации с i-й вершины определится как:

$$T_{ik} = t_i + t_{ik} + \nabla t_{ik},\ \nabla t_{ik} \in [-\nabla t^*_{ik}, \nabla t^*_{ik}], \qquad (1)$$

где t_i – время выполнения программного модуля, представленного i-й вершиной, t_{ik} – время передачи информации с i-й вершины в k-ю вершину, ∇t_{ik} – отклонение времени выполнения модуля, представленного i-й вершиной, и передачи информации по k-й связи от его среднего значения. Т.о., образуется множество времен $\{T_{ik}\}, i \in n$, т – множество дуг, входящих

в k-ю вершину. Свертку в вершине k можно трактовать как логическую функцию n переменных.

В качестве примера использования данной концепции рассмотрим две логические функции, реализованные на свертках или развертках k-х вершин, которые полностью перекрывают возможности, предоставляемые соответствующими ЕСПД по изображению последовательных алгоритмов. Функция «ИСКЛЮЧАЮЩЕЕ ИЛИ» вызывает срабатывание k-й вершины при появлении информации на k-й дуге свертки или развертки и реализует наиболее быстрый проход k-й вершины, т.к. отсутствует ожидание прихода информации на другие дуги рассматриваемой свертки или развертки. Кроме того, эта функция обеспечивает наиболее надежный узел для срабатывания, т.к. вероятность неприхода информации на рассматриваемую вершину, в общем случае, минимальна.

Функция «И», наоборот, реализует наиболее медленные проход k-й вершины и наименее надежный узел для срабатывания, т.к., в общем случае, должна прийти информация на все n дуг свертки или развертки. Все другие логические функции занимают промежуточное положение по надежности и времени срабатывания.

Рассмотрим времена срабатывания для развертки с k-й вершиной для функций «ИСКЛЮЧАЮЩЕЕ ИЛИ» и «И». Функция «ИСКЛЮЧАЮЩЕЕ ИЛИ» для случая срабатывания i-го выхода реализуется за время:

$$T_{ki} = t_{ki} + t^*_{ki} + \nabla t_{ki},\ \nabla t_{ki} \in [-\nabla t^*_{ki}, \nabla t^*_{ki}], \qquad (2)$$

где t_{ki} – время выполнения k-го модуля при формировании i-го выхода, t^*_{ki} – время передачи информации, сформированной на k-й вершине, к i-й вершине при формировании i-го выхода, ∇t_{ki} – отклонение времени выполнения модуля, представленного k-й вершиной, и передачи информации по i-му каналу от его среднего значения.

Время срабатывания для функции «И» также определяется по формуле (2), т.к. одновременность срабатывания всех выходов не требуется.

Время срабатывания для свертки с k-й вершиной для функции «ИСКЛЮЧАЮЩЕЕ ИЛИ» также определяется по формуле (2), а для функции «И» для обеспечения одновременного прихода сигнала в k-ю вершину требуется время:

$$T_{ik}^{*} = \max\{T_{ik}\}, i \in n, \quad (3)$$

где T_{ik} – времена, вычисленные по формуле (1), n – число входов в k-ю развертку.

Очевидно, что любая граф-схема алгоритма представляет собой некоторую последовательность сверток-разверток, образующих композицию из логических функций, зависящих от n переменных, где $n \in \{1, ..., \Psi\}$, где Ψ – максимальное количество дуг в свертках и развертках. Следует отметить особую роль времени t_{ki}^{*} при формировании граф-схем алгоритмов.

С помощью рассмотренных логических функций могут быть реализованы другие часто используемые конструкции. На развертках могут быть реализованы все типы основных конструкций языков программирования, используемых для создания последовательных алгоритмов.

На функции «И» k-й развертки может быть реализован цикл по счетчику циклов. Если количество дуг, исходящих из k-й вершины, ограничено по некоторым причинам есть n, а количество повторений в цикле – F, то количество обращений k-го блока к нижестоящим, связанным с ним блокам определяется соотношением $]\frac{F}{n}[$, где][– функция выделения целой части положительного числа, большей или равной этому числу.

Время прохождения информации по этому каналу определяется по формуле:

$$T_{ik}^{m} =]\frac{F}{n}[^{*} T_{ik} \quad (4)$$

Время выполнения цикла может быть уменьшено до $T_{ik}^{m} = T_{ik}$, если увеличить число каналов (число процессоров) до F.

Реализация функции «ИСКЛЮЧАЮЩЕЕ ИЛИ» на два или несколько выходов может представлять логическую функцию на два выхода (true, false), три выхода (>0, <0, =0), а также функцию переключения на один из каналов. Кроме того, задавая ту или иную логическую функцию, можно получить любую комбинацию передающих каналов, ориентированных на использование операторов – переключателей, используемых в различных языках программирования или для каких либо других целей.

При рассмотрении вычислительных систем с общей памятью (системы, в которых несколько процессоров имеют общую оперативную память) методы анализа и решения алгоритмов значительно проще, и время передачи информации, как правило, не учитывается. Учет этого времени порождает класс вычислительных систем с разделяемой памятью (система, где каждый процессор имеет свою оперативную память, и для обеспечения обмена информацией процессоры соединены каналами связи), что приводит к усложнению методов анализа и решения алгоритмов.

Литература:

1 Руденко Ю. М., Волкова Е. А. Вычислительные системы.- М.: МГТУ им. Баумана, 2010.– 211с.

2 Воеводин В. В., Воеводин Вл. В. Параллельные вычисления.- СПб.: БХВ-Петербург, 2002.- 600с.

3 Барский А. Б. Параллельные процессы в вычислительных системах.- М.: Радио и связь, 1990.- 256с.

4 Гергель В. П., Стронгин Р. Г. Основы параллельных вычислений для многопроцессорных вычислительных систем.- Н.Новгород: Изд. ННГУ им. Лобачевского, 2003.- 184с.

5 http://www.oldunesco.kemsu.ru/mps.

УДК 681.326

ТЕХНОЛОГИИ INTERNET И БАЗЫ ДАННЫХ

Б.И. Ващенко

Internet – это объединение компьютерных сетей, работающих по различным протоколам, связывающих всевозможные типы компьютеров, физически передающих данные по всем доступным типам линий – от витой пары и телефонных проводов до оптоволокна и спутниковых каналов. World Wide Web (WWW) – самый популярный сервис Internet и удобный способ работы с информацией. Сегодня существуют десятки тысяч серверов WWW. Именно за счет WWW Internet растёт так стремительно. Сегодня с развитием Internet эта технология все чаще привлекает взоры разработчиков

программного обеспечения. В мире накоплено огромное количество информации по различным вопросам. Чаще всего эта информация хранится в базах данных. Чтобы опубликовать её в Internet необходимо экспортировать базы данных в HTML – документы. В настоящий момент в мире существует масса информационных источников, доминирующим средством хранения которых являются системы управления базами данных (СУБД). Проблема предоставления удобного доступа к имеющимся в наличии базам данных очень актуальна для многих организаций, компаний, научных учреждений, и решение ее видится только в свете применения Web-технологии. World Wide Web позволяет осуществлять доступ к базам данных, предоставляя средства для разработки простого, удобного интерфейса пользователя и средства взаимодействия с прикладными программами. Интерфейс пользователя разрабатывается на основе языка гипертекстовой разметки HTML, и в частности, с использованием HTML-форм, которые являются наиболее удобным механизмом представления и передачи запросов к базам данных.

Средства по обеспечению доступа к базам данных можно классифицировать по месту обработки данных внешними программами.

- Обеспечивающие доступ к базе данных на стороне Web-сервера (средства подключения внешнего программного обеспечения – спецификация CGI, FastCGI и API-интерфейс прикладных модулей).
- Работающие на стороне клиента (средства программирования приложений - интерпретируемые языки Javascript и Java).

Средства доступа к базам данных на стороне сервера

CGI (Common Gateway Interface) – это спецификация интерфейса взаимодействия Web-сервера с внешними прикладными программами. Главное назначение CGI – обеспечение единообразного потока данных между сервером и работающим на нем приложением. CGI определяет:

1. порядок взаимодействия сервера с прикладной программой, в котором сервер выступает инициирующей стороной;
2. механизм реального обмена данными и управляющими командами в этом взаимодействии, что не определено в протоколе HTTP. Такие понятия, как метод доступа, переменные заголовка, MIME, типы данных, заимствованы из HTTP и делают спецификацию прозрачной для тех, кто знаком с самим протоколом.

При применение спецификации CGI для обмена данными с внешними прикладными программами можно выделить следующие преимущества:

- прозрачность использования;
- «языковая» независимость – CGI-программы могут быть написаны на любом языке программирования или командном языке, имеющим средства работы со строками;
- процессная изолированность – при запуске CGI-программы на сервере порождается отдельный процесс;
- открытость стандарта – CGI интерфейс применим на каждом Web-сервере;
- архитектурная независимость – CGI не зависит от особенностей реализации архитектуры сервера.

Но CGI имеет также и существенные недостатки. Главная проблема заключается в затратах на выполнение CGI-приложений: поскольку на сервере для каждого очередного запроса порождается новый процесс, который завершается после его выполнения, то это приводит к невысокому быстродействию CGI-скрипта и снижает эффективность работы сервера.

API. В ответ на ограничения и недостатки спецификации CGI была разработана спецификация прикладных модулей API, встроенных в сервер. Данное расширение Web-сервера запускается как динамическая библиотека и выполняет обработку каждого вызова сервера по отдельной структуре памяти, что значительно проще, чем создание отдельного процесса для каждого клиентского запроса. Наиболее известны два API-интерфейса – NSAPI компании Netscape и ISAPI компании Microsoft. Свободно распространяемый популярный Unix-сервер Apache также имеет модуль PHP, реализующий данный интерфейс. Приложения, работающие через API, соединяются с сервером значительно быстрее, чем CGI-программы, так как API выполняется в основном процессе сервера и постоянно находится в состоянии ожидания запросов, поэтому время на запуск программы и порождения нового процесса не требуется. API-интерфейс предоставляет и большую функциональность, чем CGI – можно написать дополнительные процедуры, осуществляющие контроль доступа к файлам. Тем не менее спецификация API не имеет преимуществ CGI-интерфейса и поставщики API-модулей тоже сталкиваются с целым рядом проблем:

- «языковая» зависимость – прикладные программы могут быть написаны только на языках, поддерживаемых в данном API;
- неизолированность процесса – так как приложения выполняются в адресном пространстве сервера, то ошибочные программы могут разрушить ядро сервера или какие-либо другие приложения;
- ограниченность применения – написанные программы в соответствии с данным API могут использоваться только на данном сервере.;
- архитектурная зависимость – API-приложения зависимы от архитектуры сервера.

FastCGI. Интерфейс FastCGI сочетает в себе наилучшие аспекты спецификаций CGI и API. Взаимодействие в соответствии с FastCGI происходит сходным образом с CGI. FastCGI-приложения запускаются отдельными изолированными процессами. Отличие состоит в том, что эти процессы являются постоянно работающими и после выполнения запроса не завершаются, а ожидают новых запросов. Вместо использования переменных окружения операционной системы и стандартных потоков ввода/вывода протокол FastCGI объединяет информацию среды, стандартный ввод, вывод и сообщения об ошибках в единственное дуплексное соединение. Это позволяет FastCGI-программам выполняться на удаленных машинах, используя TCP-соединения между Web-сервером и FastCGI-модулем. Таким образом, преимущества FastCGI состоят в следующем:

- быстродействие – благодаря постоянному функционированию FastCGI-процессов обеспечивается обслуживание одним процессом многих запросов, что решает задачу и связанные с ней проблемы порождения нового процесса на отдельный клиентский запрос;
- простота применения и легкость миграции из CGI;
- «языковая» независимость – как и CGI, FastCGI-приложения могут быть написаны на любых языках программирования или командных языках;
- изолированность процессов – «неисправные» FastCGI-программы не могут разрушить ядро сервера или какие-либо другие приложения, а также получить секретную служебную информацию;
- совместимость – FastCGI поддерживается во всех открытых продуктах, включая коммерческие серверы Netscape и Microsoft, NCSA сервер и свободно распространяемый Apache;

- архитектурная независимость – FastCGI интерфейс не зависит от особенностей реализации серверной архитектуры и прикладные программы могут быть как однопоточными, так и многопоточными;
- распределенность – FastCGI обеспечивает возможность выполнять приложения удаленно, что используется для распределенной загрузки и управления внешними Web-сайтами.

Литература:

1. Р. Ахаян, А. Горев, С. Макашарипов. Эффективная работа с СУБД. С-П: ПИТЕР. 1998г.
2. С.Д. Кузнецов. Доступ к базам данных с использованием технологии WWW. “СУБД” 5, 1999г.

УДК 519216.1/2

ОСНОВЫ СПЕКТРАЛЬНОГО АНАЛИЗА В БАЗИСЕ ХАРТЛИ

В.В. Сюзев

Широкая сфера применения комплексно-экспоненциальных базисных функций в различных областях науки и техники общеизвестна [1]. Тем не менее базис этих функций обладает существенным недостатком, связанным с его комплексным характером. Хорошей альтернативой ему служит базис Хартли, образованный из тех же тригонометрических функций, что и комплексно-экспоненциальный базис, но принимающий только вещественные значения [2]. Основам спектрального анализа в базисе Хартли и посвящена данная статья, в которой наряду с известными сведениями приводятся и новые оригинальные результаты теории функций и преобразований Хартли.

1. Функции и преобразования Хартли

В 1942 г. Р.Хартли (R.Hartley) предложил для спектрального анализа систему базисных функций *{cas(kz)}*, каждая из которых являлась суммой соответствующих тригонометрических функций [2]:

$$cas(kz)=cos(kz)+sin(kz).$$

Эта система представляла собой по сути формальную базисную систему, образованную из четных *2cos(kz)* и нечетных *2sin(kz)* простых тригонометрических функций, что позволило в данном случае из ненормированных функций *cos(kz)* и *sin(kz)* получить систему нормированных функций *{cas(kz)}*.

Базис Хартли является системой вещественных периодических функций, определенных на любом интервале длительности 2π с периодом, так же равным 2π. Он является ортонормированным на том же интервале, т.к. для любых его функций

$$\frac{1}{2\pi}\int_{2\pi} \mathrm{cas}(kz)\,\mathrm{cas}(mz)dz = \begin{cases} 1, & k = m, \\ 0, & k \neq m, \end{cases}$$

и может быть использован для разложения в ряд Фурье-Хартли математических функций, определенных на интервале, кратном 2π, и удовлетворяющих на нем условию интегрируемости с квадратом.

Для представления непрерывных сигналов с конечным временным интервалом определения длительностью T базис Хартли должен быть преобразован к виду

$$\mathrm{cas}\left(\frac{2\pi}{T}kt\right) = \cos\left(\frac{2\pi}{T}kt\right) + \sin\left(\frac{2\pi}{T}kt\right)$$

или

$$\mathrm{cas}\left(2\pi ft\right) = \cos\left(2\pi ft\right) + \sin\left(2\pi ft\right),$$

где $f=k/T$ является частотой. Пара непрерывных преобразований Хартли записывается следующим образом:

$$X(k) = \frac{1}{T}\int_T x(t)cas\left(\frac{2\pi}{T}kt\right)dt,$$

$$x(t) = \sum_{k=0}^{\infty} X(k)cas\left(\frac{2\pi}{T}kt\right), \qquad (1)$$

а соответствующее им равенство Парсеваля имеет следующий вид:

$$\frac{1}{T}\int_T x^2(t)dt = \sum_{k=0}^{\infty} X^2(k).$$

Ряд Хартли (1) для сигналов, принадлежащих пространству L_p^2, обладает среднеквадратической сходимостью к этим сигналам. Спектр *X(k)* Хартли вещественных сигналов так же является вещественным.

Все родственные системы: тригонометрическая, комплексно-экспоненциальная и Хартли содержат одни и те же простые функции. Поэтому между спектрами сигнала в этих базисах существует определенная связь. Действительно имеем

$$X_X(0)=X_Ч(0),$$
$$X_X(k)=[X_Ч(k)+X_H(k)]/2, \quad k\neq 0, \tag{2}$$
$$X_X(k)=\mathrm{Re}[X_Э(k)]-I_m[X_Э(k)].$$

Здесь в виде $X_X(k)$ обозначен спектр Хартли, в виде $X_Э(k)$ - спектр в комплексно-экспоненциальном базисе, а для тригонометрического спектра используется его обозначение в виде $X_Ч(k)$ и $X_H(k)$ [3].

Обратная связь спектров базируется на свойстве симметрии тригонометрических функций. Если представить функцию $X_X(k)$ в виде суммы четной и нечетной компонент $X_{XЧ}(k)$ и $X_{XH}(k)$ соответственно, то

$$X_{XЧ}(k)=[X_X(k)+X_X(-k)]/2=\frac{1}{T}\int_T x(t)\cos(\frac{2\pi}{T}kt)dt,$$
$$X_{XH}(k)=[X_X(k)-X_X(-k)]/2=\frac{1}{T}\int_T x(t)\sin(\frac{2\pi}{T}kt)dt.$$

Тогда будут справедливы следующие соотношения:

$$X_Ч(k)=2X_{XЧ}(k),\ X_H(k)=2X_{XH}(k),$$
$$\mathrm{Re}[X_Э(k)]=X_{XЧ}(k),\ \mathrm{Im}[X_Э(k)]=-X_{XH}(k). \tag{3}$$

Соотношения (2) и (3) взаимосвязи спектров в родственных базисах могут оказаться полезными при спектральном анализе. Например, они позволяют выразить энергетический и фазовый спектры сигнала в комплексно-экспоненциальном базисе через спектр Хартли:

$$\mathrm{Re}^2[X_Э(k)]+Im^2[X_Э(k)]=X_{XH}^2(k)+X_{XH}^2(k)=$$
$$=(1/4)[X_X(k)+X_X(-k)]^2+(1/4)[X_X(k)-X_X(-k)]^2=$$
$$=[X_X^2(k)+X^2{}_X(-k)]/2.$$
$$arg[X_Э(k)]=arctg[-X_{XH}(k)/X_{XЧ}(k)]=$$
$$=arctg\{[X_X(-k)-X_X(k)]/[X_X(k)+X_X(-k)]\}.$$

Используя понятие частоты, от непрерывных преобразований Хартли нетрудно перейти к интегральным преобразованиям:

$$X(f) = \int_{-\infty}^{\infty} x(t)\,\mathrm{cas}(2\pi ft)dt,$$

$$x(t) = \int_{-\infty}^{\infty} X(f)\cos(2\pi ft)df,$$

которые могут быть использованы при представлении непрерывных сигналов, определенных на бесконечных интервалах времени. Равенство Парсеваля в этом случае приобретает также полностью интегральный вид:

$$\int_{-\infty}^{\infty} x^2(t)dt = \int_{-\infty}^{\infty} X^2(f)df.$$

Спектральная плотность *X(f)* в этих выражениях является вещественной функцией частоты.

Для интегральных преобразований так же справедливы все соотношения взаимосвязи спектров в родственных тригонометрических базисах.

Спектральное представление дискретных финитных сигналов возможно только по системе дискретных ортонормированных базисных функций Хартли $\left\{ca\,\mathrm{s}\left(\frac{2\pi}{N}ki\right)\right\}$, которые могут быть получены путем дискретизации соответствующих непрерывных функций. Процедура дискретизации в этом случае та же, что используется при дискретизации тригонометрической и комплексно-экспоненциальной систем [3], а условие ортонормированности дискретных функций Хартли на интервале *[0,N)* выглядит так:

Дискретные преобразования Хартли на этом интервале имеют следующий вид:

$$X(k) = \frac{1}{N}\sum_{i=0}^{N-1} x(i)\,\mathrm{cas}\left(\frac{2\pi}{N}ki\right),$$

$$x(i) = \sum_{k=0}^{N-1} X(k)\,\mathrm{cas}\left(\frac{2\pi}{N}ki\right), \qquad (4)$$

где первое выражение является прямым преобразованием Хартли, а второе – обратным. Равенство Парсеваля

$$\frac{1}{N}\sum_{i=0}^{N-1} x^2(i) = \sum_{k=0}^{N-1} X^2(k)$$

является математическим и физическим подтверждением эквивалентности временной и спектральной областей представления сигналов и полноты дискретной системы Хартли. Условие ортонормированности функций Хартли, пара дискретных преобразований Хартли и равенство Парсеваля на интервале *[-N/2, N/2)* имеют аналогичный вид и отличаются от их записи на интервале *[0, N)* только пределами суммирования.

Соотношения взаимосвязи спектров в дискретном варианте также справедливы, причем вид их записи полностью совпадает с соответствующими выражениями (2) и (3). Справедлива также запись энергетического и фазового спектров для дискретных комплексно-экспоненциальных функций (ДЭФ) с помощью дискретных спектров Хартли.

Для функций Хартли не выполняется свойство мультипликативности, поэтому для них в прямом виде теоремы спектров мультипликативных базисов [4] не справедливы. Однако, благодаря связи спектров сигналов в базисах ДЭФ и Хартли, формулировка этих теорем, справедливая для комплексно-экспоненциального базиса, может быть записана с использованием спектров Хартли или, как еще говорят, в терминах спектров Хартли. Покажем это на примере двух теорем – теоремы о сдвиге и теоремы о свертке.

В соответствии с теоремой о сдвиге в базисе ДЭФ частотный спектр $Y_Э(k)$ сдвинутого сигнала $x(i-i_0)$ равен модуляции аналогичного спектра $X_Э(k)$ несдвинутого сигнала $x(i)$ базисной функцией $\exp\left(-j\frac{2\pi}{N}ki_0\right)$, т.е.

$$Y_Э(k) = X_Э(k)\exp\left(-j\frac{2\pi}{N}ki_0\right).$$

Представим этот спектр в развернутом виде:

$$Y_Э(k) = \{\mathrm{Re}[X_Э(k) + j\,\mathrm{Im}[X_Э(k)]\}\cdot(\cos(\frac{2\pi}{N}ki_0) - j\sin(\frac{2\pi}{N}ki_0)) =$$

$$= \{\mathrm{Re}[\mathrm{X_Э(k)}]\cos(\frac{2\pi}{\mathrm{N}}\mathrm{ki_0}) + \mathrm{Im}[\mathrm{X_Э(k)}]\sin(\frac{2\pi}{\mathrm{N}}\mathrm{ki_0})\} +$$

$$+ j\{\mathrm{Im}[X_Э(k)]\cos(\frac{2\pi}{N}ki_0) - \mathrm{Re}[X_Э(k)]\sin(\frac{2\pi}{N}ki_0)\}.$$

Теперь учтем связь частотного спектра со спектром Хартли (см. (3)). Тогда

$$Y_Э(k) = \mathrm{Re}[Y_Э(k)] + j\,\mathrm{Im}[Y_Э(k)] = [X_{ХЧ}(k)\cos(\frac{2\pi}{N}ki_0) - X_{ХН}(k)\sin(\frac{2\pi}{N}ki_0)] +$$

$$+j[-X_{XЧ}(k)\sin(\frac{2\pi}{N}ki_0)-X_{XH}(k)\cos(\frac{2\pi}{N}ki_0)].$$

Но

$$Y_X(k)=\mathrm{Re}[Y_Э(k)]-\mathrm{Im}[Y_Э(k)]=[X_{XЧ}(k)+X_{XH}(k)]\cdot\cos(\frac{2\pi}{N}ki_0)+$$

$$+[X_{XЧ}(k)-X_{XH}(k)]\sin(\frac{2\pi}{N}ki_0)=$$

$$=X_X(k)\cos(\frac{2\pi}{N}ki_0)+X_X(-k)\sin(\frac{2\pi}{N}ki_0). \qquad (5)$$

Из этого выражения следует, что при сдвиге сигнала во времени спектр несдвинутого сигнала модулируется косинусной составляющей функции Хартли $\cos\left(\frac{2\pi}{N}ki_0\right)$, а его зеркальная копия – синусной составляющей функции Хартли $\sin\left(\frac{2\pi}{N}ki_0\right)$. Результирующий спектр Хартли сдвинутого сигнала получатся простым суммированием модулированных составляющих спектра несдвинутого сигнала. В этом состоит суть теоремы о сдвиге в базисе Хартли.

Теперь перейдем к теореме о свертке. В соответствии с ней в базисе ДЭФ спектр $Y_Э(k)$ циклической свертки двух сигналов со спектрами $X_Э(k)$ и $U_Э(k)$ с точностью до постоянного множителя N равен произведению этих спектров:

$$Y_Э(k)=N\,X_Э(k)U_Э(k).$$

Развернем эти запись, выделив действительную и мнимую части, и учтем отмеченную ранее связь спектров ДЭФ и Хартли. Тогда после несложных преобразований получим:

$$Y_Э(k)=N[(X_{XЧ}(k)U_{XЧ}(k)-X_{XH}(k)U_{XH}(k))-j(X_{XЧ}(k)U_{XH}(k)+X_{XH}(k)U_{XЧ}(k))].$$

Так как

$$X_{XЧ}(k)=\frac{X_X(k)+X_X(-k)}{2};\ X_{XH}(k)=\frac{X_X(k)-X_X(-k)}{2};$$

$$U_{XЧ}(k)=\frac{U_X(k)+U_X(-k)}{2};\ U_{XH}(k)=\frac{U_X(k)-U_X(-k)}{2},$$

то спектр ДЭФ свертки может быть преобразован к следующему виду:

$$Y_Э(k)=\frac{N}{2}[X_X(k)U_X(-k)+X_X(-k)U_X(k)]-j\frac{N}{2}[(X_X(k)U_X(k)-X_X(-k)U_X(-k)].$$

По этому выражению легко определяется искомый спектр Хартли дискретной свертки:

$$Y_X(k) = \frac{N}{2}[X_X(k)U_X(k) - X_X(-k)U_X(-k) + X_X(k)U_X(-k) + X_X(-k)U_X(k)]. \quad (6)$$

Зависимость (6) представляет собой математическое описание теоремы о циклической дискретной свертке в терминах спектра Хартли. Как видно из нее, оно значительно сложнее аналогичного описания в частотной области. И только в частном случае, когда один или оба сигнала, используемые в свертке, являются либо четными, либо нечетными, записи теоремы о свертке в базисах ДЭФ и Хартли совпадают. Это связано с тем, что в этом случае спектры Хартли также являются либо четными, либо нечетными функциями. Так, например, для четного сигнала *x(i),* для которого *x(i)=x(N-i),* спектр

$$X_X(k) = \frac{2}{N}\sum_{i=0}^{\frac{N}{2}-1} x(i)\cos\frac{2\pi}{N}ki$$

и *X(-k)=X(k)*. Поэтому

$$Y_X(k) = \frac{N}{2}\cdot X_X(k)[U_X(k) - U_X(-k) + U_X(-k) + U_X(k)] = NX_X(k)U_X(k).$$

Однозначная взаимосвязь спектров в базисах ДЭФ и Хартли позволяет все свойства, присущие частотным спектрам, формулировать и для спектров Хартли. Можно сказать, что нет таких задач, для которых справедливо использование комплексных преобразований Фурье и одновременно не может быть применено вещественное преобразование Хартли. Тем не менее вплоть до настоящего времени область практического применения преобразований Хартли остается существенно более узкой по сравнению с областью приложений преобразований Фурье в тригонометрическом и комплексно-экспоненциальном базисах.

2. Быстрые преобразования Хартли на статических интервалах времени

Поскольку базисы дискретных функций Хартли и ДЭФ являются родственными базисными системами, то для функций Хартли, несмотря на их немультипликативность, существуют различные типы быстрых преобразований (БП), подобные известным типам БП Фурье (БПФ) [2,5]. Проиллюстрируем эффективность БП Хартли на примере быстрых преобразований по основанию 2. Алгоритмы БП по другим основаниям

могут быть получены аналогичным образом. С ними можно ознакомиться также в работе [5].

Для синтеза алгоритмов БП Хартли потребуются свойства функции Хартли суммы двух аргументов. Запишем эти свойства, используя известные тригонометрические тождества:

$$\mathrm{cas}(\alpha+\beta)=\cos(\alpha)+\sin(\alpha)=\cos(\alpha)\,\mathrm{cas}(\beta)+\sin\alpha\,\mathrm{cas}(-\beta), \tag{7}$$

$$\mathrm{cas}(\alpha+\beta)=\begin{cases}\mathrm{cas}(\alpha) & при \quad \beta=2m\pi, \quad m=0,\,1,\,2,\,...,\\ -\mathrm{cas}(\alpha) & при \quad \beta=(2m+1)\pi, \quad m=0,\,1,\,2,....\end{cases} \tag{8}$$

Перепишем теперь прямое дискретное преобразование Хартли (4) без нормирующего множителя $1/N$ $(N=2^n,\ n=1,\,2,\,...)$

$$X(k)=\sum_{i=0}^{N-1}x(i)\,\mathrm{cas}\left(\frac{2\pi}{N}ki\right),\quad k=0,\,1,\,...,\,N-1 \tag{9}$$

и применим к нему известные способы прореживания сигнала [4].

Прореживание по времени. Выборка сигнала $x(i)$ на первом уровне прореживания разделяется на две промежуточные выборки $x_0(i_1)=x(2i_1)$ и $x_1(i_1)=x(2i_1+1)$. Поэтому

$$X(k)=\sum_{i_1=0}^{N/2-1}x_0(i)\,\mathrm{cas}\left(\frac{2\pi}{N}k2i_1\right)+\sum_{i_1=0}^{N/2-1}x_1(i_1)\,\mathrm{cas}\left[\left(\frac{2\pi}{N}k(2i_1+1)\right)\right]$$

или с учетом равенства (7)

$$X(k)=\sum_{i_1=0}^{N/2-1}x_0(i)\,\mathrm{cas}\left(\frac{2\pi}{N/2}ki_1\right)+\mathrm{cas}\left(\frac{2\pi}{N}k\right)\sum_{i_1=0}^{N/2-1}x_1(i_1)\cos\left(\frac{2\pi}{N/2}ki_1\right)+$$

$$+\mathrm{cas}\left(-\frac{2\pi}{N}k\right)\sum_{i_1=0}^{N/2-1}x_1(i_1)\sin\left(\frac{2\pi}{N/2}ki_1\right). \tag{10}$$

Вычислим с помощью этого уравнения первую часть спектра $X(k_1),\ k_1=0,\,1,\,...,\,N/2-1$. Она будет равна

$$X(k_1)=\sum_{i_1=0}^{N/2-1}x_0(i_1)\,\mathrm{cas}\left(\frac{2\pi}{N/2}k_1i_1\right)+\mathrm{cas}\left(\frac{2\pi}{N}k_1\right)\sum_{i_1=0}^{N/2-1}x_1(i_1)\cos\left(\frac{2\pi}{N/2}k_1i_1\right)+$$

$$+\mathrm{cas}\left(-\frac{2\pi}{N}k_1\right)\sum_{i_1=0}^{N/2-1}x_1(i_1)\sin\left(\frac{2\pi}{N/2}k_1i_1\right).$$

Но, поскольку

$$\sum_{i_1=0}^{N/2-1}x_0(i_1)\,\mathrm{cas}\left(\frac{2\pi}{N/2}k_1i_1\right)=X^{(0)}(k_1),$$

$$\sum_{i_1=0}^{N/2-1} x_1(i_1)\,\mathrm{cas}(\frac{2\pi}{N/2}k_1 i_1) = X^{(1)}(k_1)$$

($X^{(0)}(k_1),\ X^{(1)}(k_1)$ – спектры Хартли четной и нечетной промежуточной выборки соответственно), а

$$\sum_{i_1=0}^{N/2-1} x_1(i_1)\cos(\frac{2\pi}{N/2}k_1 i_1) = [X^{(1)}(k_1) + X^{(1)}(-k_1)]/2 = [X^{(1)}(k_1) + X^{(1)}(\frac{N}{2}-k_1)]/2,$$

$$\sum_{i_1=0}^{N/2-1} x_1(i_1)\sin(\frac{2\pi}{N/2}k_1 i_1) = [X^{(1)}(k_1) - X^{(1)}(-k_1)]/2 = [X^{(1)}(k_1) - X^{(1)}(\frac{N}{2}-k_1)]/2, \quad (11)$$

то после преобразований получаем

$$X(k_1) = X^{(0)}(k_1) + \cos(\frac{2\pi}{N}k_1)X^{(1)}(k_1) + \sin(\frac{2\pi}{N}k_1)X^{(1)}(\frac{N}{2}-k_1).$$

Для второй половины спектра $X(k_1+N/2),\ k_1 = 0, 1, ..., N/2-1$ из (10) имеем:

$$X(k_1+N/2) = \sum_{i_1=0}^{N/2-1} x_0(i_1)\,\mathrm{cas}(\frac{2\pi}{N/2}k_1 i_1 + 2\pi i_1) +$$

$$+\mathrm{cas}(\frac{2\pi}{N}k_1+\pi)\sum_{i_1=0}^{N/2-1} x_1(i_1)\cos(\frac{2\pi}{N/2}k_1 i_1 + 2\pi i_1) +$$

$$+\mathrm{cas}[-(\frac{2\pi}{N}k_1+\pi)]\sum_{i_1=0}^{N/2-1} x_1(i_1)\sin(\frac{2\pi}{N/2}k_1 i_1 + 2\pi i_1).$$

Учитывая в этом уравнении свойство функций Хартли (8), периодичность тригонометрических функций синуса и косинуса и соотношения (11), после преобразований получим

$$X(k_1+N/2) = X^{(0)}(k_1) - \cos(\frac{2\pi}{N}k_1)X^{(1)}(k_1) - \sin(\frac{2\pi}{N}k_1)X^{(1)}(\frac{N}{2}-k_1).$$

Объединяя уравнения для первой и второй части спектра, приходим к общему алгоритму БП Хартли по основанию 2 на первом уровне прореживания по времени:

$$X(k_1) = X^{(0)}(k_1) + [\cos(\frac{2\pi}{N}k_1)X^{(1)}(k_1) + \sin(\frac{2\pi}{N}k_1)X^{(1)}(N/2-k_1)],$$

$$X(k_1+N/2) = X^{(0)}(k_1) - [\cos(\frac{2\pi}{N}k_1)X^{(1)}(k_1) + \sin(\frac{2\pi}{N}k_1)X^{(1)}(N/2-k_1)], \quad (12)$$

$$k_1 = 0, 1, ..., N/2-1.$$

В этом алгоритме спектр исходной выборки сигнала выражается через спектры промежуточных выборок.

Поскольку $N/2$ делится на 2, то полученный алгоритм можно применить и для вычисления промежуточных спектров, введя новый уровень прореживания. На *m*-м уровне алгоритм БП Хартли будет иметь следующий вид

$$\left.\begin{aligned} X^{(\lambda_1,\lambda_2,\ldots\lambda_{m-1})}(k_m) &= X^{(\lambda_1,\lambda_2,\ldots\lambda_{m-1},0)}(k_m)+[\cos(\frac{2\pi}{2^{n-m+1}}k_m)X^{(\lambda_1,\lambda_2,\ldots,\lambda_{m-1},1)}(k_m)+ \\ &+\sin(\frac{2\pi}{2^{n-m+1}}k_m)X^{(\lambda_1,\lambda_2,\ldots,\lambda_{m-1},1)}(2^{n-m}-k_m)], \\ X^{(\lambda_1,\lambda_2,\ldots,\lambda_{m-1})}(k_m+2^{n-m}) &= X^{(\lambda_1,\lambda_2,\ldots,\lambda_{m-1},0)}(k_m)-[\cos(\frac{2\pi k_m}{2^{n-m+1}})X^{(\lambda_1,\lambda_1,\ldots,\lambda_{m-1},1)}(k_m)+ \\ &+\sin(\frac{2\pi k_m}{2^{n-m+1}})X^{(\lambda_1,\lambda_2,\ldots,\lambda_{m-1},1)}(2^{n-m}-k_m)], \\ k_m &= 0,\,1,\,\ldots,\,2^{n-m}-1;\quad \lambda_\alpha = 0,\,1;\quad \alpha = 1,\,2,\,\ldots,\,m-1, \end{aligned}\right\} \tag{13}$$

и подобен алгоритму Кули-Тьюки с прореживанием по времен [4]. На последнем $(n-1)$-м уровне прореживания при полном алгоритме БП Хартли $2^{n-m+1}=4,\ k_{n-1}=0,1$ и тригонометрические множители принимают только значения 1 или 0. Поэтому при $m=(n-1)$ алгоритм (13) упрощается

$$\begin{aligned} X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2})}(0) &= X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2},0)}(0)+X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2},1)}(0), \\ X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2})}(1) &= X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2},0)}(1)+X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2},1)}(1), \\ X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2})}(2) &= X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2},0)}(0)-X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2},1)}(0), \\ X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2})}(3) &= X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2},0)}(1)-X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-2},1)}(1). \end{aligned}$$

Спектры $X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-1})}(k_{n-1})$ промежуточных выборок в этом случае вычисляются с помощью 2-точечных дискретных преобразований Хартли, которые также имеют простейший вид:

$$\begin{aligned} X^{(\lambda_1,\lambda_2,\ldots\lambda_{n-1})}(0) &= x(2^{n-2}\lambda_{n-1}+\cdots+2\lambda_2+\lambda_1)+ \\ &+x(2^{n-1}+2^{n-2}\lambda_{n-1}+\cdots+2\lambda_2+\lambda_1), \end{aligned} \tag{14}$$

$$X^{(\lambda_1,\lambda_2,\ldots,\lambda_{n-1})}(1) = x(2^{n-2}\lambda_{n-1} + \cdots + 2\lambda_2 + \lambda_1) -$$
$$-x(2^{n-1} + 2^{n-2}\lambda_{n-1} + \cdots + 2\lambda_2 + \lambda_1).$$

Здесь учтено, что $cas(0) = 1$, а $cas(\pi) = -1$. Эти уравнения задают начальные условия для итерационного процесса вычисления спектра Хартли по БА (13) при *m = n – 1, n – 2, ... , 1*.

Оценим вычислительную сложность алгоритма БП Хартли. Подсчитаем сначала число умножений $M_Б$. На последнем шаге алгоритма умножений нет. На каждом *m*-м шаге из всех остальных шагов выполняется *N* общих умножений, 2^m из которых являются тривиальными, т.к. соответствуют нулевым и единичным значениям тригонометрических констант при индексах i_m и k_m, равных 0. Поэтому

$$M_Б = \sum_{m=1}^{n-2}(N - 2^m) = N(n - 2{,}5) + 2 = N(\log_2 N - 2{,}5) + 2.$$

Теперь оценим число сложений $A_Б$. Сложения выполняются на всех шагах, причем на каждом *m*-м шаге из $n - 2$ первых шагов выполняется по $1{,}5N - 2^m$ сложений, а на последнем шаге – $2{,}5N$ сложений. Общее число сложений будет равно

$$A_Б = \sum_{m=1}^{n-2}(1{,}5N - 2^m) + 2{,}5N = N(1{,}5n - 1) + 2 = N(1{,}5\log_2 N - 1) + 2.$$

Реальное число вычислительных операций можно получить еще меньше, если учесть тривиальные множители, равные 0 и ± 1, имеющие место и при ненулевых значениях индексов i_m и k_m. Более близкими к реальным являются оценки

$$M_Б = N(n - 3) + 4 = N(\log_2 N - 3) + 4,$$
$$A_Б = 1{,}5N(n - 1) + 2 = 1{,}5N(\log_2 N - 1) + 2.$$

Следует отметить, что все приведенные оценки $M_Б$ и $A_Б$ справедливы для вещественного входного сигнала. В случае комплексного входного сигнала они удваиваются.

Полный алгоритм БП Хартли полезно представлять в виде ориентированного сигнального графа. По сравнению с БПФ сигнальный граф БП Хартли будет содержать дополнительные узлы, зато умножения будут выполняться только на вещественные константы.

Пример 1. Записать полный алгоритм БП Хартли и построить его сигнальный граф для *N*=8.

Решение. Алгоритм будет иметь два уровня прореживания. По формулам (13), (14) получаем:

- на втором уровне $(m=2;\ \lambda_1=0,1;\ \lambda_2=0,1;\ i_2,k_2=0,1)$:

$$x_{0,0}(i_2)=\{x(0),x(4)\};\quad x_{0,1}(i_2)=\{x(2),x(6)\};$$

$$x_{1,0}(i_2)=\{x(1),x(5)\};\quad x_{1,1}(i_2)=\{x(3),x(7)\};$$

$$X^{(0,0)}(0)=x(0)+x(4);\quad X^{(0,0)}(1)=x(0)-x(4);$$

$$X^{(0,1)}(0)=x(2)+x(6);\quad X^{(0,1)}(1)=x(2)-x(6);$$

$$X^{(1,0)}(0)=x(1)+x(5);\quad X^{(1,0)}(1)=x(1)-x(5);$$

$$X^{(1,1)}(0)=x(3)+x(7);\quad X^{(1,1)}(1)=x(3)-x(7);$$

$$X^{(0)}(0)=X^{(0,0)}(0)+X^{(0,1)}(0);\quad X^{(0)}(1)=X^{(0,0)}(1)+X^{(0,1)}(1);$$

$$X^{(0)}(2)=X^{(0,0)}(0)-X^{(0,1)}(0);\quad X^{(0)}(3)=X^{(0,0)}(1)-X^{(0,1)}(1);$$

$$X^{(1)}(0)=X^{(1,0)}(0)+X^{(1,1)}(0);\quad X^{(1)}(1)=X^{(1,0)}(1)+X^{(1,1)}(1);$$

$$X^{(1)}(2)=X^{(1,0)}(0)-X^{(1,1)}(0);\quad X^{(1)}(3)=X^{(1,0)}(1)-X^{(1,1)}(1);$$

- на первом уровне (m=1; λ_1=0,1; k_1=0,1,2,3,):

$$X(0)=X^{(0)}(0)+X^{(1)}(0);\quad X(1)=X^{(0)}(1)+\frac{\sqrt{2}}{2}[X^{(1)}(1)+X^{(1)}(3)];$$

$$X(2)=X^{(0)}(2)+X^{(1)}(2);\quad X(3)=X^{(0)}(3)-\frac{\sqrt{2}}{2}[X^{(1)}(1)-X^{(1)}(3)];$$

$$X(4)=X^{(0)}(0)-X^{(1)}(0);\quad X(5)=X^{(0)}(1)-\frac{\sqrt{2}}{2}[X^{(1)}(1)+X^{(1)}(3)];$$

$$X(6)=X^{(0)}(2)-X^{(1)}(2);\quad X(7)=X^{(0)}(3)+\frac{\sqrt{2}}{2}[X^{(1)}(1)-X^{(1)}(3)].$$

Сигнальный граф этого алгоритма приведен на рисунке 1. Его структура похожа на структуру графа алгоритма БПФ Кули-Тьюки [4], хотя и содержит два дополнительных узла. На реализацию этого графа нужно затратить 26 сложений и 2 умножения на вещественные константы.

Проверка. Для проверки алгоритма вычислим с его помощью пятый спектральный коэффициент: $X(5)=X^{(0,0)}(1)+X^{(0,1)}(1)-\frac{\sqrt{2}}{2}[X^{(1,0)}(1)+X^{(1,1)}(1)]-$

$$-\frac{\sqrt{2}}{2}[X^{(1,0)}(1)-X^{(1,1)}(1)]=x(0)-x(4)+x(2)-x(6)-\frac{\sqrt{2}}{2}[x(1)-x(5)]-\frac{\sqrt{2}}{2}[x(3)-x(7)]-$$

$$-\frac{\sqrt{2}}{2}[x(1)-x(5)]+\frac{\sqrt{2}}{2}[x(3)-x(7)]= x(0)-\sqrt{2}x(1)+x(2)-x(4)+\sqrt{2}x(5)-x(6).$$

Расчет по прямому алгоритму дает следующий результат:

$$X(5)=\sum_{i=0}^{7}x(i)\,\mathrm{cas}(5\pi i/4)=$$

$$= x(0)+x(1)\,\mathrm{cas}(5\pi/4)+ x(2)\,\mathrm{cas}(5\pi/2)+x(3)\,\mathrm{cas}(15\pi/4)+$$

$$+x(4)\,\mathrm{cas}(5\pi)+ x(5)\,\mathrm{cas}(25\pi/4)+ x(6)\mathrm{cas}(30\pi/4)+$$

$$+x(7)\,\mathrm{cas}(35\pi/4)=x(0)-\sqrt{2}x(1)+x(2)-$$

$$-x(4)+\sqrt{2}x(5)-x(6).$$

Результаты совпадают.

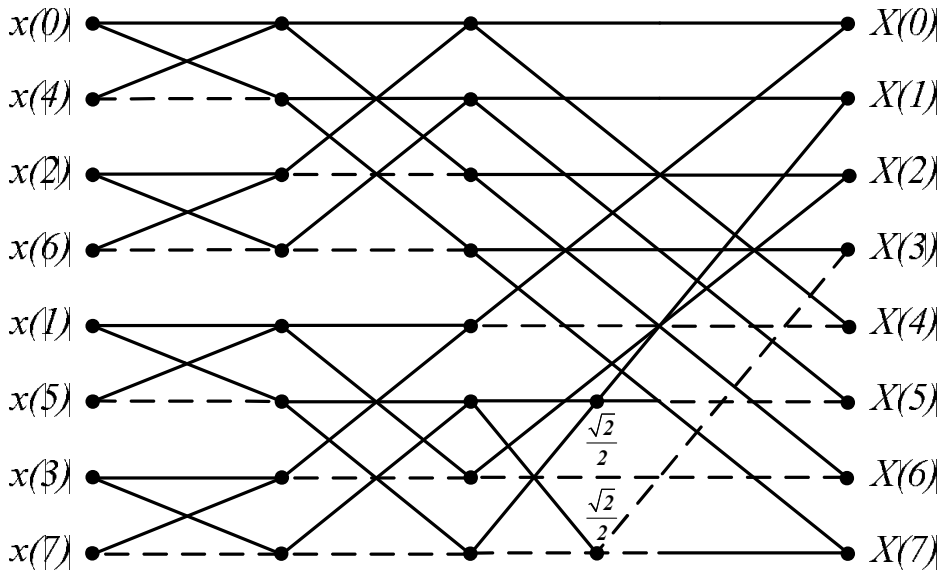


Рис. 1. Сигнальный граф полного БП Хартли по основанию 2 с прореживанием по времени для N=8

_____ . _____

Прореживание по частоте. Из выборки сигнала на первом уровне прореживания образуются две промежуточные выборки $\{x(i_1)\}$ и $\{x(i_1+N/2)\}$, а затем с помощью уравнения (9) раздельно записываются спектральные составляющие с четными и нечетными номерами. Для $k=2k_1$ получаем

$$X(2k_1)=\sum_{i_1=0}^{N/2-1}x(i_1)\,\mathrm{cas}(\frac{2\pi}{N/2}k_1i_1)+\sum_{i_1=0}^{N/2-1}x(i_1+N/2)\,\mathrm{cas}(\frac{2\pi}{N/2}k_1i_1+2\pi k_1)=$$

$$= \sum_{i_1=0}^{N/2-1} [x(i_1) + x(i_1 + N/2)] \mathrm{cas}(\frac{2\pi}{N/2} k_1 i_1) = \sum_{i_1=0}^{N/2-1} x^{(0)}(i_1) \mathrm{cas}(\frac{2\pi}{N/2} k_1 i_1), \quad (15)$$

где

$$x^{(0)}(i_1) = x(i_1) + x(i_1 + N/2), \quad (16)$$

а для $k=2k_1 + 1$ с учетом свойств функций Хартли (7) и (8) –

$$X(2k_1 + 1) = \sum_{i_1=0}^{N/2-1} [x(i_1) - x(i_1 + N/2)] \mathrm{cas}(\frac{2\pi}{N} i_1) \cos(\frac{2\pi}{N/2} k_1 i_1) +$$

$$+ \sum_{i_1=0}^{N/2-1} [x(i_1) - x(i_1 + N/2)] \mathrm{cas}(-\frac{2\pi}{N} i_1) \sin(\frac{2\pi}{N/2} k_1 i_1).$$

Используя соотношения (11), последнее уравнение можно преобразовать к виду

$$X(2k_1 + 1) = \sum_{i_1=0}^{N/2-1} [x(i_1) - x(i_1 + N/2)] \cos(\frac{2\pi}{N} i_1) \mathrm{cas}(\frac{2\pi}{N/2} k_1 i_1) +$$

$$+ \sum_{i_1=0}^{N/2-1} [x(i_1) - x(i_1 + N/2)] \sin(\frac{2\pi}{N} i_1) \mathrm{cas}[\frac{2\pi}{N/2} k_1 (N/2 - i_1)],$$

а, изменив с помощью линейной подстановки $i_1 = N/2 - i_1$ порядок суммирования во второй сумме этого выражения, его можно записать как

$$X(2k_1 + 1) = \sum_{i_1=0}^{N/2-1} x^{(1)}(i_1) \mathrm{cas}(\frac{2\pi}{N/2} k_1 i_1), \quad (17)$$

где

$$x^{(1)}(i_1) = [x(i_1) - x(i_1 + N/2)] \cos(\frac{2\pi}{N} i_1) + [x(N/2 - i_1) - x(N - i_1)] \sin(\frac{2\pi}{N} i_1) \quad (18)$$

Таким образом, в виде уравнений (15)-(18) получен алгоритм БП Хартли по основанию 2 на первом уровне прореживания по частоте. Из него следует, что полный спектр Хартли можно вычислить с помощью $N/2$-точечных ДПФ Хартли над суммой и разностями отсчетов промежуточных выборок, причем разности отсчетов должны быть предварительно умножены на соответствующие тригонометрические множители.

По способу организации вычислительного процесса этот алгоритм подобен алгоритму БПФ Кули-Тьюки с прореживанием по частоте [4]. Его можно применить и для вычисления $N/2$-точечных ДПФ Хартли, используя следующий уровень прореживания. Процедуру прореживания можно продолжать до тех пор, пока число отсчетов в промежуточных выборках не

станет равным 2. В этом случае будет достигнута максимальная глубина прореживания, равная n-1, и получен полный алгоритм БП Хартли с прореживанием по частоте.

На m-м уровне прореживания по частоте БП Хартли по аналогии с БПФ представляется следующим образом:

$$X(2^m k_m + 2^{m-2} q_{m-1} + \cdots + 2q_2 + q_1) = \sum_{i_m=0}^{2^{n-m}-1} x^{(q_1, q_2, \ldots, q_{m-1}, 0)}(i_m)\mathrm{cas}(\frac{2\pi}{2^{n-m}} k_m i_m),$$

$$X(2^m k_m + 2^{m-1} + 2^{m-2} q_{m-1} + \cdots + 2q_2 + q_1) = \sum_{i_m=0}^{2^{n-m}-1} x^{(q_1, q_2, \ldots, q_{m-1}, 1)}(i_m)\mathrm{cas}(\frac{2\pi}{2^{n-m}} k_m i_m), \quad (19)$$

$$k_m = 0, 1, \ldots, 2^{n-m} - 1; \; q_\alpha = 0, 1; \; \alpha = 1, 2, \ldots, m-1,$$

где

$$x^{(q_1, q_2, \ldots, q_{m-1}, 0)}(i_m) = x^{(q_1, q_2, \ldots, q_{m-1})}(i_m) + x^{(q_1, q_2, \ldots, q_{m-1})}(i_m + 2^{n-m}),$$

$$x^{(q_1, q_2, \ldots, q_{m-1}, 1)}(i_m) = [x^{(q_1, q_2, \ldots, q_{m-1})}(i_m) -$$

$$- x^{(q_1, q_2, \ldots, q_{m-1})}(i_m + 2^{n-m})]\cos(\frac{2\pi}{2^{n-m+1}} i_m) +$$

$$+ [x^{(q_1, q_2, \ldots, q_{m-1})}(2^{n-m} - i_m) - x^{(q_1, q_2, \ldots, q_{m-1})}(2^{n-m+1} - i_m)]\sin(\frac{2\pi i_m}{2^{n-m+1}}) \quad (20)$$

и описывает итерационный процесс с m=1,2, .., n-1 при начальных значениях в виде (16) и (18). На последнем шаге (при m=n-1) ДПФ Хартли в (19) будут 2-точечными и выполняются без умножений. Не потребуются умножения и при вычислении величин (20), поскольку тригонометрические множители в этом случае будут равны 1 либо 0.

Алгоритм БП Хартли с прореживанием по частоте будет содержать такое же число операций, как и алгоритм БП Хартли с прореживанием по времени. Поэтому для него так же справедливы все приведенные ранее оценки. Его можно проиллюстрировать с помощью сигнального графа, который в этом случае будет иметь структуру, близкую к структуре графа БПФ Кули-Тьюки с прореживанием по частоте, хотя и будет содержать дополнительные вычислительные узлы.

Пример 2. Записать алгоритм БП Хартли с прореживанием по частоте и построить его сигнальный граф для N=8.

Решение. Полный алгоритм БП Хартли в этом случае будет иметь два уровня прореживания, т.е m=1,2. В соответствии с общим алгоритмом (19), (20) получаем:

- на первом уровне прореживания (m=1; i_1=0, 1, 2, 3; q=0, 1):

$$x^{(0)}(0) = x(0) + x(4); \quad x^{(0)}(1) = x(1) + x(5);$$

$$x^{(0)}(2) = x(2) + x(6); \quad x^{(0)}(3) = x(3) + x(7);$$

$$x^{(1)}(0) = x(0) - x(4); \quad x^{(1)}(1) = \frac{\sqrt{2}}{2}\{[x(1) - x(5)] + [x(3) - x(7)]\};$$

$$x^{(1)}(2) = x(2) - x(6); \quad x^{(1)}(3) = \frac{\sqrt{2}}{2}\{[x(1) - x(5)] - [x(3) - x(7)]\};$$

– на втором уровне прореживания $(m = 2;\ i_2, k_2 = 0, 1;\ q_2 = 0, 1)$:

$$x^{(0,0)}(0) = x^{(0)}(0) + x^{(0)}(2); \quad x^{(0,0)}(1) = x^{(0)}(1) + x^{(0)}(3);$$

$$x^{(1,0)}(0) = x^{(1)}(0) + x^{(1)}(2); \quad x^{(1,0)}(1) = x^{(1)}(1) + x^{(1)}(3);$$

$$x^{(0,1)}(0) = x^{(0)}(0) - x^{(0)}(2); \quad x^{(0,1)}(1) = x^{(0)}(1) - x^{(0)}(3);$$

$$x^{(1,1)}(0) = x^{(1)}(0) - x^{(1)}(2); \quad x^{(1,1)}(1) = x^{(1)}(1) - x^{(1)}(3)$$

и результирующий спектр равен:

$$X(0) = x^{(0,0)}(0) + x^{(0,0)}(1); \quad X(4) = x^{(0,0)}(0) - x^{(0,0)}(1);$$

$$X(2) = x^{(0,1)}(0) + x^{(0,1)}(1); \quad X(6) = x^{(0,1)}(0) - x^{(0,1)}(1);$$

$$X(1) = x^{(1,0)}(0) + x^{(1,0)}(1); \quad X(5) = x^{(1,0)}(0) - x^{(1,0)}(1);$$

$$X(3) = x^{(1,1)}(0) + x^{(1,1)}(1); \quad X(7) = x^{(1,1)}(0) - x^{(1,1)}(1).$$

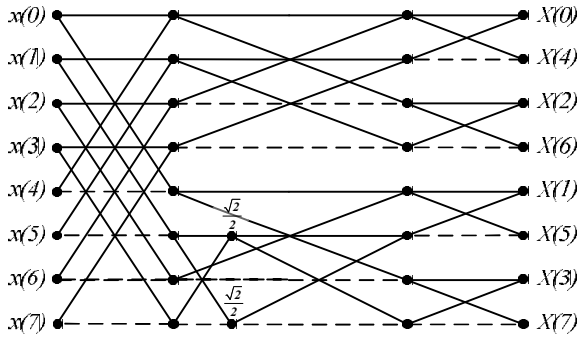


Рис. 2. Сигнальный граф полного БП Хартли
по основанию 2 с прореживанием по частоте для N=8

Сигнальный граф этого алгоритма представлен на рисунке 2. Он имеет такие же сложностные характеристики, что и граф рисунка 1.

Проверка. Её выполним для того же пятого коэффициента:

$$X(5) = x^{(1,0)}(0) - x^{(1,0)}(1) = x^{(1)}(0) + x^{(1)}(2) - x^{(1)}(1) - x^{(1)}(3) =$$

$$x(0) - x(4) + x(2) - x(6) - \frac{\sqrt{2}}{2}x(1) +$$

$$+\frac{\sqrt{2}}{2}x(5) - \frac{\sqrt{2}}{2}x(3) + \frac{\sqrt{2}}{2}x(7) + \frac{\sqrt{2}}{2}x(3) - \frac{\sqrt{2}}{2}x(7) - \frac{\sqrt{2}}{2}x(1) + \frac{\sqrt{2}}{2}x(5) =$$

$$x(0) - \sqrt{2}x(1) + x(2) -$$

$$-x(4) + \sqrt{2}x(5) - x(6).$$

Полученное значение совпадает со значением этого коэффициента, рассчитанным в предыдущем примере.

_____ . _____

Используя общую методику синтеза БПФ различного типа и учитывая специфику функций Хартли, аналогичным образом можно записать алгоритмы БП Хартли по другим основаниям, для взаимно-простых множителей и гнездового типа (типа Винограда) [5]. БП Хартли по своим вычислительным характеристикам не уступают БПФ, что только усиливает перспективность их применения для частотного анализа вещественных сигналов.

3. Преобразования Хартли на скользящих интервалах времени

Статические БП Хартли очень близки по описанию к статическим БПФ, что объясняется, как уже отмечалось, родственностью систем функций Хартли и ДЭФ. Это позволяет существенно упростить процедуру синтеза скользящих БП Хартли за счет использования уже разработанных скользящих БПФ. Применим такой подход к синтезу скользящих БП Хартли по основанию 2 с прореживанием по времени.

Сравнение статических БПФ Кули-Тьюки [4] и БП Хартли (13) показывает, что алгоритм Хартли можно получить из алгоритма Кули-Тьюки, если в последнем на каждой итерации одно умножение промежуточного спектра нечетной подвыборки на комплексную константу $W_{2^{n-m+1}}^{-k_m}$ заменить на два умножения одной половины этого спектра на вещественную константу $\cos(2\pi k_m / 2^{n-m+1})$, а другой – на вещественную константу $\sin(2\pi k_m / 2^{n-m+1})$. В остальном процедура вычислений остается

прежней. Очевидно, что то же самое можно проделать и применительно к скользящему БПФ Кули-Тьюки с прореживанием по времени [4], получив в результате следующий алгоритм скользящего БП Хартли по основанию 2 с μ уровнями прореживания по времени:

$$\left.\begin{aligned}
X_j^{(m-1)}(k_m) &= X_j^{(m)}(k_m) + [\cos(\pi k_m / 2^{n-m+1}) X_{j-2^{m-1}}^{(m)}(k_m) + \\
&\quad + \sin(2\pi k_m / 2^{n-m+1}) X_{j-2^{m-1}}^{(m)}(2^{n-m} - k_m)], \\
X_j^{(m-1)}(k_m + 2^{n-m}) &= X_j^{(m)}(k_m) - [\cos(2\pi k_m / 2^{n-m+1} X_{j-2^{m-1}}^{(m)}(k_m) + \\
&\quad + \sin(2\pi k_m / 2^{n-m+1}) X_{j-2^{m-1}}^{(m)}(2^{n-m} - k_m)], \\
k_m &= 0, 1, ..., 2^{n-m} - 1; \quad m = 1, 2, ..., \mu; \\
X_j^{(\mu)}(k_\mu) &= \sum_{i_\mu=0}^{2^{n-\mu}-1} x(j - 2^\mu i_\mu) cas(2\pi k_\mu i_\mu / 2^{n-\mu}).
\end{aligned}\right\} \quad (21)$$

При $\mu = n-1$ скользящий алгоритм (21) становится полным.

Для полного алгоритма на последнем (n-1)-м уровне прореживания тригонометрические константы принимают только значения 1 и 0, поэтому при $m = n-1$ алгоритм (21) упрощается

$$X_j^{(n-2)}(0) = X_j^{(n-1)}(0) + X_{j-2^{n-2}}^{(n-1)}(0),$$

$$X_j^{(n-2)}(1) = X_j^{(n-1)}(1) + X_{j-2^{n-2}}^{(n-1)}(1),$$

$$X_j^{(n-2)}(2) = X_j^{(n-1)}(0) - X_{j-2^{n-2}}^{(n-1)}(0),$$

$$X_j^{(n-2)}(3) = X_j^{(n-1)}(1) - X_{j-2^{n-2}}^{(n-1)}(1).$$

Спектры $X_j^{(n-1)}(k_{n-1})$ в этом случае вычисляются с помощью 2-точечных дискретных преобразований Хартли:

$$X_j^{(n-1)}(0) = x(j) + x(j - 2^{n-1}),$$

$$X_j^{(n-1)}(1) = x(j) - x(j - 2^{n-1}).$$

Эти уравнения определяют начальные данные для итерационного процесса вычисления скользящего спектра Хартли по БА (21) при $m = n-1,\ n-2, ..., 1$. Его результатом будет спектр $X_j^{(0)}(k)$, являющийся искомым спектром $X_j(k)$ входного сигнала к j-му моменту текущему времени.

Проведем оценку сложности скользящего алгоритма (21). На последней итерации умножения не выполняются, а выполняется только 6 сложений, 2 из которых используются при определении начальных данных. На каждой m-й из всех последующих итераций алгоритма выполняются $2\cdot 2^{n-m}$ общих *умножений* и $3\cdot 2^{n-m}$ сложений. Однако для двух значений индекса k_m, равных 0 и 2^{n-m-1}, тригонометрические функции косинуса и синуса будут равны 1 или 0 и поэтому в этом случае умножения станут тривиальными, а число слагаемых в уравнениях (21) сократиться до двух. По этой причине число нетривиальных операций на m-й итерации будет равно $2(2^{n-m}-2)$ умножений и $(3\cdot 2^{n-m}-2)$ сложений. Отсюда следует и общее число нетривиальных операций в самом алгоритме:

$$M_c=\sum_{m=1}^{n-2}2(2^{n-m}-2)=2(N-2n)=2(N-2\log_2 N),$$

$$A_c=\sum_{m=1}^{n-2}(3\cdot 2^{n-m}-2)+6=3N-2(n+1)=3N-2(\log_2 N+1). \qquad (22)$$

Алгоритм еще можно улучшить за счет дополнительного уменьшения числа умножений, если учесть, что тригонометрические множители при $k_m=2^{n-m-2}$ и $k_m=3\cdot 2^{n-m-2}$ по модулю равны между собой и их можно вынести за скобки, обеспечив тем самым выполнение двух умножений вместо четырех. В этом случае на каждой m-й итерации из $(n-2)$-х в алгоритме (21) будет выполняться $(2^{n-m+1}-6)$ умножений, а их общее число будет равно

$$M_c=2[N-(3n-2)]=2[N-(3\log_2 N-2)]. \qquad (23)$$

Число дополнительных данных, которые необходимо хранить в памяти для реализации скользящего преобразования Хартли, такое же, как в скользящем алгоритме БПФ Кули-Тьюки по основанию 2 с прореживанием по времени. Пример 3. Записать полный алгоритм скользящего БП Хартли для *N*=8.

Решение. Алгоритм (21) при *n*=3 будет иметь всего 2 итерации.

Итерация 1 (*m*=1, *k*=0, 1, 2, 3):

$$X_j(0)=X_j^{(1)}(0)+X_{j-1}^{(1)}(0);\ X_j(1)=X_j^{(1)}(1)+\frac{\sqrt{2}}{2}[X_{j-1}^{(1)}(1)+X_{j-1}^{(1)}(3)];$$

$$X_j(2)=X_j^{(1)}(2)+X_{j-1}^{(1)}(2);\ X_j(3)=X_j^{(1)}(3)-\frac{\sqrt{2}}{2}[X_{j-1}^{(1)}(1)-X_{j-1}^{(1)}(3)];$$

$$X_j(4) = X_j^{(1)}(0) - X_{j-1}^{(1)}(0);\ \ X_j(5) = X_j^{(1)}(1) - \frac{\sqrt{2}}{2}[X_{j-1}^{(1)}(1) + X_{j-1}^{(1)}(3)];$$

$$X_j(6) = X_j^{(1)}(2) - X_{j-1}^{(1)}(2);\ \ X_j(7) = X_j^{(1)}(3) + \frac{\sqrt{2}}{2}[X_{j-1}^{(1)}(1) - X_{j-1}^{(1)}(3)].$$

Итерация 2 ($m = 2,\ k_2 = 0, 1$):

$$X_j^{(1)}(0) = X_j^{(2)}(0) + X_{j-2}^{(2)}(0);\ \ X_j^{(1)}(1) = X_j^{(2)}(1) + X_{j-2}^{(2)}(1);$$

$$X_j^{(1)}(2) = X_j^{(2)}(0) - X_{j-2}^{(2)}(0);\ \ X_j^{(1)}(3) = X_j^{(2)}(1) - X_{j-2}^{(2)}(1),$$

где

$$X_j^{(2)}(0) = x(j) + x(j-4);\ \ X_j^{(2)}(1) = x(j) - x(j-4).$$

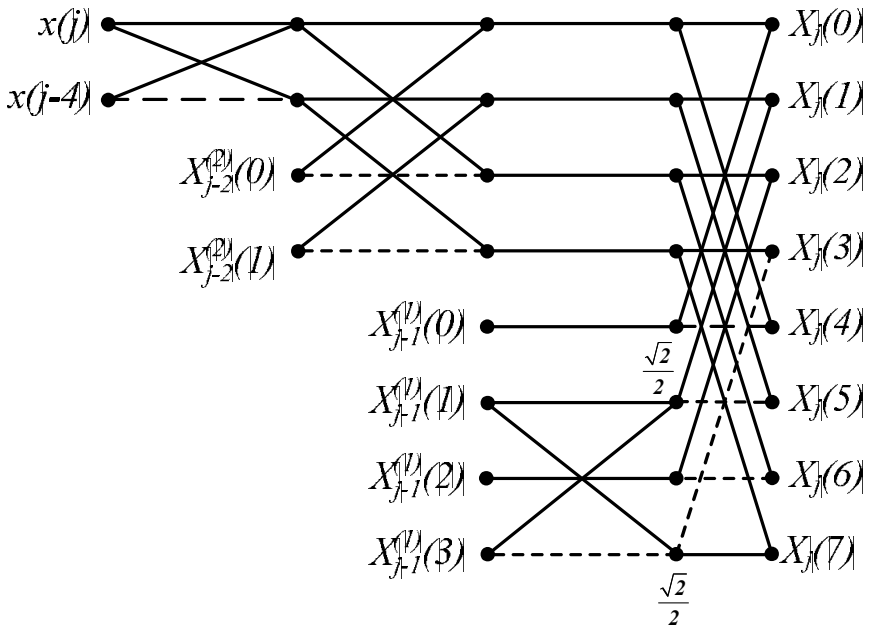


Рис. 3. Сигнальный граф полного скользящего преобразования Хартли по основанию 2 с прореживанием по времени для N=8

Сигнальный граф этого алгоритма представлен на рисунке 3. Алгоритм содержит 2 умножения на вещественные константы $\sqrt{2}/2$ и 16 сложений, что соответствует оценкам (22) и (23) при N=8.

Сигнальные графы скользящих преобразований Хартли по форме отличаются от графов БПФ Кули-Тьюки и содержат дополнительные узлы. Зато все выполняемые в них операции вещественные.

Приведенные результаты, включающие свойства базисных функций и методы синтеза быстрых преобразований, составляют теоретическую основу спектрального анализа в базисе Хартли. Его применение особенно перспективно при решении спектральными методами различных задач обработки действительных сигналов.

Настоящая статья подготовлена по результатам НИР в рамках реализации ФЦП «Научные и научно-педагогические кадры инновационной России на 2009-2013 годы (Государственный контракт №П1264 от 27.08.2009г.).

Литература:

1. Залманзон Л.А. Преобразования Фурье, Уолша, Хаара и их применение в управлении, связи и других областях. – М.: Наука, 1989. – 496с.

2. Брейсуэл Р. Пребразование Хартли: Пер. с англ. – М.: Мир, 1990. – 175с.

3. Трахтман А.М. Введение в обобщенную спектральную теорию сигналов. – М.: Сов. радио, 1972. – 345с.

4. Трахтман А.М., Трахтман В.А. Основы теории дискретных сигналов на конечных интервалах. – М.: Сов. радио, 1975. – 208с.

5. Власенко В.А., Лаппа Ю.М., Ярославский Л.П. Методы синтеза быстрых алгоритмов свертки и спектрального анализа сигналов. – М.: Наука, 1990. – 180с.

УДК 004.35

ПРОЕКТИРОВАНИЕ USB-УСТРОЙСТВ

Д.А. Овчаренко, В.Я. Хартов

Устройства, подключаемые к персональному компьютеру и передающие информацию по шине USB, широко применимы и популярны в настоящее время. Шина USB уже вытеснила последовательный интерфейс с портом COM и все больше заменяет параллельный LPT.

Интерфейс USB удобен для пользователя («горячее подключение», plug-n-play, компактность, универсальность), но требует от разработчика программного и аппаратного обеспечения глубоких знаний для управления устройствами на физическом уровне. В докладе кратко обсуждаются спецификации шины USB, которые необходимо знать при разработке нового устройства и рассматриваются методы возможного ускорения и облегчения разработки USB-устройств.

USB – последовательный интерфейс передачи данных для среднескоростных и низкоскоростных периферийных устройств вычислительной техники. Шина USB имеет три режима передачи данных: низкоскоростной (LS, Low-speed) – до 1,5 Мбит/с, полноскоростной (FS, Full-speed) – до 12 Мбит/с, высокоскоростной (HS, High-speed, для USB 2.0) – до 480 Мбит/с. Однако, на практике достичь пиковой пропускной способности 60 Мбайт/с не удается, реально скорость передачи данных вдвое меньше. Это объясняется тем, что устройства инерционны и между запросом на передачу данных и передачей возникают задержки.

Для подключения периферийных устройств к шине USB используется четырёхпроводный кабель, при этом две линии в дифференциальном включении используются для приёма и передачи данных и две линии для питания периферийного устройства (+5В) и общий (GND). Передача сигналов по шине осуществляется с использованием метода NRZI – кодирование без возвращения к нулю и с инверсией. В полноскоростном USB-кабеле логической единице соответствует напряжение +5В, в низкоскоростном – 0В.

С работой шины связаны понятия «главное устройство» – хост, он же USB контроллер, который обычно встроен в микросхему южного моста на материнской плате ПЭВМ, и «периферийные устройства».

На логическом уровне устройство USB поддерживает транзакции приема и передачи данных. Пакет каждой транзакции содержит в себе номер конечной точки на устройстве.

При подключении устройства драйверы в ядре ОС читают с устройства список конечных точек из нулевой конечной точки и создают управляющие структуры данных для общения с каждой конечной точкой устройства. Совокупность конечной точки и структур данных в ядре ОС называется каналом.

USB поддерживает четыре типа передачи данных:

- *управляющие передачи* – передачи «вопрос – ответ», необходимые для настройки*;*
- *потоковые* – гарантирующие доставку каждого пакета, приостановку при переполнении и опустошении буфера, но не гарантирующие заданной скорости*;*

- *передачи по прерываниям* – короткие сообщения без подтверждения, гарантирующие своевременную доставку*;*
- *изохронные передачи* – потоковые передачи информации без подтверждения, гарантирующие передачу заданного числа пакетов на один период шины. Используется для аудио и видеоинформации, информации реального времени.

Время шины делится на периоды, в начале каждого периода контроллер передает всей шине маркер начала периода. Далее в течение периода передаются пакеты прерываний, потом изохронные в требуемом количестве, в оставшееся время периода передаются управляющие пакеты и в последнюю очередь потоковые.

Все USB устройства принимают запросы от хоста и отвечают на них через основной канал сообщений. Запросы выполняются при помощи управляющих посылок. Запрос и его параметры передаются устройству в конфигурационном пакете, который имеет размер 8 байт. Тип запроса имеет длину 1 байт и содержит всю необходимую информацию по флагам.

Существуют запросы, определяемые разработчиком, и стандартные запросы USB:

GET_STATUS – возвращает состояние (вкл/выкл) устройства;

CLEAR_FEATURE/SET_FEATURE – сбросить/установить функциональные группы реализации устройства;

SET_ADDRESS – передает устройству его адрес;

GET_DESCRIPTOR/SET_DESCRIPTOR – возвращает/устанавливает дескриптор устройства и все дескрипторы конечных точек;

GET_COFIGURATION/SET_CONFIGURATION – возвращает/устанавливает конфигурацию устройства;

GET_INTERFACE/SET_INTERFACE – возвращает/устанавливает информацию об интерфейсе; SYNC_FRAME – сообщение о кадре синхронизации конечной точки.

Разработка произвольного устройства USB в общем случае сводится к выбору микросхемы, поддерживающей USB (например – 32-разрядный микроконтроллер) и написанию драйвера для операционной системы, который будет с этим микроконтроллером взаимодействовать. При этом

написание драйвера – процесс сложный и требует от программиста, как знания операционной системы, так и аппаратной части.

Несмотря на сложность разработки существует способ, позволяющий реализовать разные USB-устройства, для которых не существенна высокая скорость передачи данных (ниже 64 Кбайт/с). Это использование класса USB-устройств HID (Human Interface Device).

Поскольку, начиная с Windows 98, ОС имеют встроенные HID-Class драйверы, отпадает необходимость в трудоемкой собственной разработке драйвера для нового устройства. Также драйверы и библиотеки работы с HID имеются и в Linux.

При разработке HID устройства программисту уже не нужно программировать микроконтроллер на работу с драйвером и создавать сам драйвер устройства. В задачи проектировщика входит конфигурирование устройства на работу в нужном режиме HID, и в моменты, когда необходима передача (при нажатии кнопок, событии, цикле таймера), выполнять вызов процедур передачи и получения данных. Сами данные помещаются в структуру данных, называемую Report (репорт), который на персональном компьютере считывается с заданным программистом интервалом. На ЭВМ программисту нужно написать программу, которая бы считывала все HID-устройства, потом из них выделяла нужное (например, по VendorID и DeviceID с использованием функций GetManufacturerString и GetProductString), получала его Report и разбирала его по структуре, заданной стандартом и дополненной самим разработчиком.

При этом на компьютере программисту требуется только вызывать функции работы с HID устройствами (GetReport, ReadFile, RawInput) из стандартных или сторонних библиотек (SetupApi и Hid), которые во многом похожи на функции работы с файлами.

Пример последовательности вызовов функций для взаимодействия с устройством:

CreateFile – открытие HID-устройство (создание дескриптора) и проверка, в каком режиме оно работает (чтение, чтение и запись);

HidD_GetAttributes – получение атрибутов устройства (VendorID, DeviceID и др.);

HidD_GetPreparsedData – получение блока данных для обработки HID репорта;

HidP_GetCaps – получение информации об использовании устройства и используемой страницы, среди которых важным параметром является длина буфера репорта;

HidD_GetInputReport – получение репорта для его последующей обработки;

HidD_FreePreparsedData – освобождение выделенной памяти.

Помимо обращений через Report с HID-устройством можно взаимодействовать через RawInput («сырой ввод»). Приложение получает от зарегистрированного устройства данные в необработанном виде в сообщениях WM_INPUT, все время получая от него информацию. Таким образом можно реализовать программы, отслеживающие показания датчиков, или подключить к компьютеру несколько мышей и раздельно получать данные от них. Однако, некоторые антивирусные программы фиксируют использование функции RawInput как «клавиатурный перехватчик».

В настоящее время идет внедрение стандарта USB 3.0, на сегодня уже 10% новых материнских плат поддерживают спецификацию USB 3.0. Разъёмы и кабели обновлённого стандарта физически и функционально совместимы с USB 2.0. К кабелю версии 2.0 в USB 3.0 добавлены еще четыре линии связи. Спецификация USB 3.0 предусматривает максимальную скорость передачи информации до 4,8 Гбит/с, нагрузочная способность увеличена с 500 мА до 900 мА. Стандарт USB развивается, его поддержка, разработка устройств с использованием USB несомненно будет актуальна в обозримом будущем.

В заключение можно отметить, что разработка несложного USB-устройства может быть проведена в короткие сроки с использованием HID-класса устройств. Класс реализуемых устройств не ограничен исключительно взаимодействием с человеком: помимо клавиатур, указателей, джойстиков, регуляторов это могут быть и датчики, считыватели, индикаторы и нестандартные устройства. В качестве примера возможного использования HID-устройтсва, не отвечающего за интерфейс с пользователем, можно представить коммутатор распределенного по домовладению аудиосигнала, который будет представлять собой микроконтроллер, подключаемый по шине USB к PC, соединенный с электрически управляемыми переключателями аудиосигнала. Микроконтроллер, получив от PC HID-

сообщение, выдает низкий или выскоий уровень напряжения на указанный выход, подключенный к переключателю, тот в свою очередь, управляемый воздействием, переключает канал, либо выключает его, тем самым управляя источником звукового сигнала конкретного динамика или комнаты. Если вместо переключателей подключить управляемые резисторы, то можно регулировать громкость. При этом нет необходимости реализовывать устройство физического взаимодействия с пользователем, все управление можно производить с PC.

Литература:

1. В.Я. Хартов. Микропроцессорные системы. Москва. Академия. 2010г.
2. Павел Агуров. Практика программирования USB. Санкт-Перебург, БХВ-Петербург, 2006г.
3. Тревор Мартин. Микроконтроллеры ARM7. Семейство LPC2000 компании Philips. Вводный курс. 2006г.
4. Спецификации стандарта USB.
5. Спецификации микроконтроллера SAM7.
6. Документация msdn.
7. Маркетинговый анализ с использованием Яндекс.Маркет.

УДК 004.822

ФОРМАЛИЗАЦИЯ И СРАВНЕНИЕ УЧЕБНЫХ ПРОГРАММ НА ОСНОВЕ ОНТОЛОГИЧЕСКОГО ПОДХОДА

Е.А.Черникова

Вступление

В настоящее время активно развиваются разработка и внедрение информационных систем поддержки управления университетом, в частности особое внимание уделяется разработке подсистем, осуществляющих поддержку процессов обучения и управление учебным процессом. Одной из важнейших задач является создание подсистем, позволяющих автоматизировать разработку и сопровождение учебных планов (УП) и программ учебных дисциплин (ПУД). Целью данной работы является

автоматизация сравнения учебных планов ВУЗов на основе их формализации в виде онтологий и последующего выравнивания.

Структура учебных планов и программ учебных дисциплин

В ВУЗах Российской Федерации программы учебных дисциплин и учебные планы составляются на основании Федерального Государственного образовательного стандарта высшего профессионального образования (ФГОС ВПО). ФГОС ВПО определяет структуру и основную обязательную программу учебных дисциплин и учебных планов по направлению подготовки специалиста, бакалавра или магистра по определенной специальности. Помимо этого ФГОСом ВПО как в ПУД, так и в УП, предполагается наличие вариативной части, определяемой каждым ВУЗом самостоятельно. Таким образом, характерной особенностью учебных планов и программ учебных дисциплин являются четкая структурированность и глубина изложения материала учебного курса. В различных разделах документов определяются цели и задачи дисциплины или учебной программы, основное содержание, а также знания, умения и навыки, получаемые в процессе обучения.

В рамках данной работы программы учебных дисциплин и учебные планы формализуются в виде онтологий. Данный подход позволяет отразить не только структурную, но и семантическую составляющую этих документов, что является необходимым для решения задачи их сравнения.

Онтология образовательной программы

Онтология образовательной программы строится на основе трех взаимосвязанных онтологий более низкого уровня: онтологии учебного плана, онтологии программы учебной дисциплины и онтологии предполагаемого результата обучения. Каждая из онтологий состоит из классов, их свойств, отношений между классами, индивидов. Для заполнения части классов онтологии образовательной программы на основе онтологий нижнего уровня применяются правила на языке SWRL. Таким образом, онтология образовательной программы задается четверкой вида O=(C,R,I,RL), где C – классы, R – свойства и отношения между классами, I – индивиды, RL – правила на языке SWRL.

Онтология учебного плана в качестве классов содержит основные характеризующие его понятия, такие как название, степень, получаемая студентом в результате его освоения, направление подготовки, трудоемкость, а также список кодов дисциплин с указанием обязательности изучения каждой из них. Кроме того, онтология УП содержит два подкласса предполагаемых результатов обучения (ПРО). С точки зрения естественного языка, предполагаемый результат обучения – это высказывание, характеризующие знания, умения и навыки, приобретаемые учащимися в результате выполнения учебных заданий [1]. Первый из подклассов ПРО онтологии УП содержит результаты обучения высокого уровня абстракции, указанные непосредственно в учебном плане. Во второй подкласс индивиды поднимаются с уровня входящих в учебный план дисциплин при помощи обработки ризонером SWRL-правила вида:

Учебный план(?x) ∧ Дисциплина(?y) ∧ Предполагаемый Результат Обучения(?z) ∧ включает Дисциплину(?x, ?y) ∧ имеетДисциплинаПредполагаемый Результат Обучения (?y, ?z) → имеетУчебныйПланПредполагаемый Результат Обучения (?x, ?z)

Онтология программы учебной дисциплины в качестве классов состоит из основных концептов ПУД, таких как название, предметная область, ссылки на предварительно изучаемые дисциплины, ключевые понятия, а также предполагаемые результаты обучения.

Онтология предполагаемого результата обучения

В данной работе схожесть предполагаемых результатов обучения студента дисциплинам, входящим в определенные учебные планы, является основным критерием похожести учебных планов. Данный подход мотивирован тем, что предполагаемые результаты обучения описывают не только изучаемые понятия и объекты предметной области дисциплины, но и действия, которые студент должен уметь выполнять над объектами предметной области каждой из дисциплин с указанием уровня овладения данным знанием, умением или навыком.

На основе анализа ФГОС [2], а также учебных планов и программ учебных дисциплин факультета «Информатика и системы управления» МГТУ им. Н.Э. Баумана было выявлено, что предполагаемый результат обучения состоит из глагола, выражающего действие над объектом

предметной области, объекта действия, степени овладения данным действием, а также уточняющего предложения. Обязательным является наличие первых двух составляющих. На основе выше сказанного была составлена онтология предполагаемого результата обучения, которая состоит из следующих классов: глагол-действие, объект действия, степень овладения действием, а также ряда классов для представления возможных уточняющих предложений. Основными классами, используемыми для выравнивания, являются первые три.

Семантическая мера для сравнения предполагаемых результатов обучения

Основной задачей при выравнивании онтологий является определение меры схожести между их сущностями. В нашем случае таковой является мера схожести между предполагаемыми результатами обучения, рассчитываемая как нормализованная взвешенная сумма мер схожести между глаголами-действиями, объектами действия и степенями овладения материалом.

$$sim_{\text{ПРО1,ПРО2}} = \frac{w_1 \times sim_{\text{гл-д1,гл-д2}} \mid w_2 \times sim_{\text{об-д1,об-д2}} \mid w_3 \times sim_{\text{ст-овл1,ст-овл2}}}{\sum_{i=1}^{3} w_i} \quad (1).$$

Алгоритм выравнивания онтологий учебных планов

Выравнивание онтологий учебных планов заключается в нахождении пар наиболее схожих предполагаемых результатов обучения дисциплинам, составляющим эти образовательные программы. Нахождение меры схожести между двумя онтологиями учебных планов заключается в агрегации максимальных значений мер схожести ПРО. Для этой цели применяется «жадная стратегия» [3]. Онтологии учебных планов выравниваются попарно.

Алгоритм выравнивания онтологий учебных планов заключается в следующем:

1. Вычисление по формуле (см. формула 1) мер схожести между каждой парой предполагаемых результатов обучения сравниваемых учебных планов.

2. Заполнение матрицы найденными значениями мер схожести ПРО. Матрица мер схожести по вертикали содержит ПРО из одного учебного плана, по горизонтали – из второго. Элементами матрицы являются значения мер схожести ПРО.

3. Нахождение и запоминание значения элемента матрицы, имеющего максимальное значение.

4. Вычеркивание строки и столбца матрицы, в которых найдено максимальное значение, что означает, что находящиеся в них элементы не принимают участия в дальнейших итерациях.

5. Переход на шаг 3, если в матрице осталась хотя бы одна пара строка-столбец.

6. Вычисление меры схожести учебных планов по следующей формуле

$$sim_{УП1,УП2} = \frac{\sum[\max(sim]_{ПРО1,ПРО2})}{n} \qquad (2),$$

где: в числителе содержится сумма значений, полученных на шаге 3,

n – количество мер схожести, учитываемых в сравнении учебных планов.

Заключение

В данной статье описано использование онтологического подхода для представления и сравнения учебных планов ВУЗов.

Для решения поставленной задачи разработана методика представления учебных планов ВУЗов и программ учебных дисциплин в виде онтологий. На основе анализа УП и ПУД показано, что сравнение должно проводиться на основе предполагаемых результатов обучения студента. Построена онтологическая модель ПРО. Разработан алгоритм выравнивания онтологий учебных планов, в рамках которого введена семантическая мера схожести предполагаемых результатов обучения дисциплинам.

Литература:

1. Phillips L. The Continuing Education Guide: The CEU and Other Professional Development Criteria. Kendall/Hunt Publishing Company, 1994.

2. Федеральный государственный образовательный стандарт высшего профессионального образования по направлению подготовки «230100 Информатика и вычислительная техника» (квалификация (степень) «бакалавр»). Утвержден приказом Министерства образования и науки Российской Федерации от 9 ноября 2009 года №553.

3. Ehrig. M. Ontology Alignment: Bridging the Semantic Gap. Springer Science-i-Business Media, 2007, стр. 73.

УДК 681.3.06

ОЦЕНКА НАДЕЖНОСТИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Д.А. Ошуркевич

Надёжность программного обеспечения (ПО) является одной из важнейших его характеристик, и хотя абсолютная надёжность современного программного обеспечения, по-видимому, недостижима, до сих пор не существует общепринятой меры надёжности компьютерных программ.

Программа считается правильной (корректной), если она не содержит ошибок. Было выполнено довольно много работ, в которых предлагались различные методы оценки числа оставшихся в программе ошибок по результатам тестирования, в том числе метод «засорения» известными ошибками, однако количество ошибок в программе не имеет никакого отношения к её надёжности.

К динамическим моделям надежности ПО относятся, например, модели La Padula. По этой модели выполнение последовательности тестов производится в т этапов. Каждый этап заканчивается внесением изменений (исправлений) в ПО. Возрастающая функция надежности базируется на числе ошибок, обнаруженных в ходе каждого тестового прогона.

Надежность ПО в течение i-го этапа:

$R(t) = R(\infty) - A / i$, $i = 1,2,3,\ldots,$

где A – параметр роста;

$R(i) \to R(\infty)$ при i → ∞. Т.е R(∞) – предельная надежность ПО.

Эти неизвестные величины вычисляются при решении следующих уравнений:

$$\sum_{i=1}^{m}\{\frac{S_i - m_i}{S_i} - R(\infty) + A/i\} = 0,$$

$$\sum_{i=1}^{m}\{(\frac{S_i - m_i}{S_i} - R(\infty) + A/i)(\frac{1}{i})\} = 0,$$

где S_i – число тестов;

m_i – число отказов во время i-го этапа:

m – число этапов;

$i = 1,2, ...,m.$

Определяемый по этой модели показатель есть надежность ПО на i-м этапе:

$R(i) = R(\infty) - A/i$, $i = m+1, m+2 ...$

Преимущество модели заключается в том, что она является прогнозной и, основываясь на данных, полученных в ходе тестирования, дает возможность предсказать вероятность безотказной работы программы на последующих этапах ее выполнения.

Преимущество модели заключается в том, что она является прогнозной и, основываясь на данных, полученных в ходе тестирования, даёт возможность предсказать вероятность безотказной работы программы на последующих этапах ее выполнения.

Статистические модели надежности ПО принципиально отличаются от динамических прежде всего тем, что в них не учитывается время появления ошибок в процессе тестирования. Использование модели Миллса предполагает необходимость перед началом тестирования искусственно вносить в программу («засорять») некоторое количество известных ошибок. Ошибки вносятся случайным образом и фиксируются в протоколе искусственных ошибок. Специалист, проводящий тестирование, не знает ни количества, ни характера внесенных ошибок до момента оценки показателей надежности по модели Миллса. Предполагается, что все ошибки (как естественные, так и искусственно внесенные) имеют равную вероятность быть найденными в процессе тестирования.

Тестируя программу в течение некоторого времени, собирается статистика об ошибках. В момент оценки надежности по протоколу искусственных ошибок все ошибки делятся на собственные и искусственные. Соотношение:

$$N = \frac{S \cdot n}{V},$$

дает возможность оценить N – первоначальное число ошибок в программе. В данном соотношении, которое называется формулой Миллса, S – количество искусственно внесенных ошибок, n – число найденных собственных ошибок, V – число обнаруженных к моменту оценки искусственных ошибок. Например, если в программу внесено 50 ошибок и к некоторому моменту тестирования обнаружено 25 собственных и 5 внесенных ошибок, то по формуле Миллса делается предположение, что первоначально в программе было 250 ошибок.

Вторая часть модели связана с проверкой гипотезы от N. Предположим, что в программе имеется K собственных ошибок и внесем в нее еще S ошибок. В процессе тестирования были обнаружены все S внесенных ошибок и n собственных ошибок.

Тогда по формуле Миллса мы предполагаем, что первоначально в программе было $N = n$ ошибок. Вероятность, с которой можно высказать такое предположение, возможно рассчитать по следующему соотношению:

$$C = \begin{cases} 1, \text{ если } n > K, \\ \dfrac{S}{S+K+1}, \text{ если } n \le K. \end{cases} \quad (1)$$

Например, если утверждается, что в программе нет ошибок ($K = 0$), и при внесении в программу 10 ошибок все они в процессе тестирования обнаружены, но при этом не выявлено ни одной собственной, то $C = 0{,}9$. То есть с вероятностью 0,9 можно утверждать, что в программе нет ошибок. Но если в процессе тестирования была обнаружена одна собственная ошибка, то $C = 1$, так как $n > K$, и наше предположение о том, что в программе нет ошибок, на 100% не подтвердилось.

Таким образом, величина C является мерой доверия к модели и показывает вероятность того, насколько правильно найдено значение N. Эти два связанных между собой по смыслу соотношения образуют полезную модель ошибок: первое предсказывает возможное число первоначально имевшихся в программе ошибок, а второе используется для установления доверительного уровня прогноза. Однако формула (1) для расчета C не может быть использована в случае, когда не обнаружены все искусственно рассеянные ошибки. Для этого случая, когда оценка надежности

производится до момента обнаружения всех 5 рассеянных ошибок, величина *C* рассчитывается по модифицированной формуле (2):

$$C = \begin{cases} 1, \text{ если } n > K, \\ \binom{S}{V-1} / \binom{S+K+1}{V+K}, \text{ если } n \le K, \end{cases} \quad (2)$$

где числитель и знаменатель формулы при $n < K$ являются биномиальными коэффициентами вида:

$$(\frac{a}{b}) = \frac{a!}{b!(a-b)!}.$$

В действительности модель Миллса можно использовать для оценки *N* после каждой найденной ошибки. Предлагается во время всего периода тестирования отмечать на графике число найденных ошибок и, текущие значения для *N*. Достоинством модели являются простота применяемого математического аппарата, наглядность и возможность использования в процессе тестирования.

Однако она не лишена и ряда недостатков, самые существенные из которых - это необходимость внесения искусственных ошибок (этот процесс плохо формализуем) и достаточно вольное допущение величины *K*, которое основывается исключительно на интуиции и опыте человека, проводящего оценку, т.е. допускает большое влияние субъективного фактора.

Эмпирические модели в основном базируются на анализе структурных особенностей программного средства (или программы). Как указывалось ранее, эмпирические модели часто не дают конечных результатов показателей надёжности, однако их использование на этапе проектирования ПО полезно для прогнозирования требующихся ресурсов тестирования, уточнения плановых сроков завершения проекта и т.д.

Для разработки программной реализации моделей надежности ПО были выбраны модели, использование которых наиболее полно описывает ПО на всех стадиях его разработки. Для этого были использованы, как статистические, так и динамические модели. Была разработана программа для подсчета различных оценочных данных кода на различных этапах разработки ПО. Она также выводит по запросу пользователя статистику использования моделей, средние значения и сохраняет полученные данные в файл.

Помимо описанных выше моделей, существует много других. Охватить весь ряд моделей достаточно сложно и не имеет смысла.

В действительности с помощью таких моделей реально спрогнозировать вероятность вычисления всех ошибок на разных стадиях создания ПО. Каждая модель имеет свои недостатки и преимущества.

Так, в динамических моделях учитываются появления ошибок и вероятность их появлений, поэтому в динамических моделях удобно вносить корректировки в программу на текущем этапе тестирования. Статистические модели принципиально отличаются от динамических, так как в них тестирование основывается только на статистических данных. Эмпирические модели часто не дают окончательного представления о показателях надёжности ПО, однако использование их на этапе проектирования ПС полезно для прогнозирования требующихся ресурсов тестирования, уточнения плановых сроков завершения проекта и т.д.

При разработке и тестировании ПО не обязательно на соответствующих этапах использовать модели в том порядке, который был приведён выше. Тем более, что помимо описанных моделей существует множество других. Выбранные для программной реализации модели наиболее полно описывают цикл разработки ПО, а также помогают оценить количество ошибок в уже готовых программных продуктах.

Литература:

1.Липаев В.В.. Тестирование программ. М.: Радио и связь, 1986. - 234 с.

2.Фокс Дж. Программное обеспечение и его разработка. М., 1985.

3.Штрик А.А., Осовецкий Л.Г., Мессих И.Г. Структурное проектирование надёжных программ встроенных ЭВМ. – Л.: Машиностроение. Ленингр. отд-ние, 1989. - 296 с.

+ ДИСККККК

УДК 681.3.07

ИСКЛЮЧЕНИЕ ВНУТРЕННИХ ПОДЗАГОЛОВКОВ И ИЗБАВЛЕНИЕ ОТ СЛОЖНЫХ АТРИБУТОВ ПРИ ПРЕОБРАЗОВАНИИ НЕРЕЛЯЦИОННЫХ ТАБЛИЦ К РЕЛЯЦИОННОМУ ВИДУ

Мин Тхет Тин, А.В. Брешенков

К настоящему времени во всех отраслях человеческой деятельности накопился большой объем информации, представленной в табличной форме. Примерами такого рода информации служат справочники, прайс-листы, ведомости и многое другое. При этом данная информация может быть представлена разнообразными способами: на бумаге, в HTML формате, в форматах электронных таблиц, в формате текстовых редакторов, в формате текстовых процессоров и в других формах. Как показывает опыт работы на предприятиях различного профиля, в большинстве случаев имеется настоятельная необходимость получения возможности использования высокоэффективных средств современных баз данных для обработки такого рода информации. В связи с этим возникает проблема преобразования информации табличного вида к реляционному виду, то есть к виду, приемлемому для ее использования в реляционных системах управления базами данных.

Реляционными таблицами называют таблицы, обладающие следующими свойствами:

- для каждого столбца таблицы должно выполняться требование однотипности элементов;
- каждая строка таблицы должна быть уникальной;
- порядок строк и столбцов таблицы может быть произвольным;
- каждая ячейка таблицы не должна содержать в себе других ячеек;
- в таблице не должно быть подзаголовков[1].

В работе [2] рассматриваются задачи автоматизированного проектирования реляционных баз данных на основе использования существующей информации табличного вида. В ней выполнен анализ и решение следующих задач:

- приведение информации табличного вида к реляционному виду;
- назначение первичных ключей в заполненных реляционных таблицах;
- нормализация заполненных реляционных таблиц;
- формирование связей между заполненными реляционными таблицами.

В настоящей работе рассмотрены вопросы, относящиеся к первой задаче, причем только в части последнего требования к реляционным таблицам (в таблице не должно быть подзаголовков). При этом анализируются вопросы, которые не нашли своего отражения в работе [2].

В нереляционных таблицах могут встречаться подзаголовки трех типов: внешние подзаголовки, внутренние подзаголовки и подзаголовки-столбцы. В реляционных таблицах подзаголовки недопустимы. Рассмотрим примеры подзаголовков различных типов.

Внешние подзаголовки. Такого рода подзаголовки часто встречаются в таблицах-справочниках. В качестве примера в таблице 1 приведем фрагмент каталога электрооборудования автомобилей.

Таблица 1

Указатели				Системы зажигания			
Габаритные указатели		Поворотные указатели		Контактные		Бесконтактные	
№	Марка	№	Марка	№	Марка	№	Марка

Как видно из структуры таблицы она имеет заголовки трех уровней - основной заголовок, подзаголовок первого уровня и подзаголовок второго уровня. В реляционных таблицах допустимы заголовки только одного уровня. Для информации представленной в табличной форме для общего случая справедливы следующие утверждения:

T={З1,З2,…,ЗК,….,ЗN}, ЗК={П1ЗК,П2ЗК,…,ПLЗК,….,ПFЗК}

ПLЗК={П1ПLЗК, П2ПLЗК,…., ПQПLЗК,…., ПРПLЗК},

где T – схема таблицы;

ЗК – схема К-того заголовка таблицы Т;

ПLЗК – L-ый подзаголовок первого уровня ЗК- го заголовка таблицы Т;

ПQПLЗК – Q-ой подзаголовок второго уровня ПLЗК-го подзаголовка первого уровня ЗК-го заголовки таблицы Т (1).

Следует обратить внимание на то, что число уровней подзаголовков может быть существенно б*о*льшим, но правила формирования их псевдонимов аналогичны приведенным. Здесь же из соображений ограничения работы принято ограничиться тремя уровнями заголовков и подзаголовков. Как показывает анализ доступных документов различных предметных областей, обычно задействуют два уровня подзаголовков. Это видно из примера приведенного во фрагменте таблицы крепежных деталей (таблица 2).

Таблица 2

Болты				Шайбы				Винты								Штифты				Гайки				Шплинты	
6-и гранные		Цилиндрические с внутренним 6-й гранником		Плоские		Гравер		с чечевичной головкой		с полусферической головкой		с потайной головкой		с цилиндрической головкой		Простые		Конусообразные		Основные		Коронки			
N	T	N	T	N	T	N	T	N	T	N	T	N	T	N	T	N	T	N	T	N	T	N	T	N	T

Для этой таблицы могут быть задействованы заголовки более высокого уровня. Их схема следующая:

Нормаль = {гайки, болты, винты, шайбы}.

Фиксирующие детали = {штифты, шплинты, клинья}.

В обозначениях столбцов (1) типа ПQПLЗК заложены индексы, с помощью которых могут быть построены циклы, содержащиеся в алгоритмах исключения внешних заголовков. Так например, для основного цикла могут быть задействованы индексы K и N . Для 1-го внутреннего цикла индексы L и K . Для 2-ого внутреннего индексы Q и P. Чтобы исключить сложные заголовки и привести таблицу к реляционному виду, необходимо организовать сканирование подзаголовков самого нижнего уровня и для каждого подзаголовка осуществить сбор всей относящейся к нему информации. Затем собранную информацию необходимо использовать в качестве неделимого (атомарного) заголовка.

Например, 1-й полученный таким образом заголовок таблицы 2 будет выглядеть следующим образом: N шестигранные болты или болты шестигранные N. Второй заголовок может быть таким: болты шестигранные тип. Далеко не всегда заголовки, полученные таким образом, могут быть воспринимаемы потенциальным пользователем базы данных. Более того, их длина может превышать максимально допустимую длину атрибутов используемых инструментальных средств. В связи с этим процедура формирования атомарных столбцов должна быть не автоматической, а автоматизированной. То есть пользователь средств исключения внешних подзаголовков должен иметь возможность вмешаться в процесс формирования заголовков столбцов с целью присвоения атомарным

столбцам приемлемых имен. Один из возможных алгоритмов исключения внешних подзаголовков приведен в работе [3].

Следует обратить внимание на то, что исходная информация может быть представлена различными способами: на бумаге, в виде текстовых файлов, в формате электронных таблиц и др. Удобнее всего для разработчика средств преобразования было бы использование в качестве исходного единый формат данных. Например, в качестве основного - использование формата Excel, а все другие представления данных преобразовывать в данный формат. Но, к сожалению, не всегда это возможно. Например, табличные данные. представленные на бумаге, далеко не всегда удается отсканировать в формат электронных таблиц.

Поэтому необходимо предусмотреть реализацию алгоритма не только для информации табличного вида, представленной в формате .xls, но и в формате .txt.

Внутренние подзаголовки. Подзаголовки такого рода часто встречаются в прайс-листах, в отчетах по покупке и продаже товаров, то есть в тех случаях, когда структуры нескольких таблиц совпадают, но их данные относятся к различным группам. Группировка может, осуществляется, например, по датам, категориям товаров, регионам. В качестве примера рассмотрит список проданных за день товаров магазина «Соки - воды» (таблица 3).

Таблица 3

Название	Обьем	Цена	Количество
Газированная вода			
Тархун	1 л	30р	7
Байкал	2 л	60р	5
Колокольчик	1,5 л	40р	10
Соки			
Вишневый	1 л	45р	3
Ананасовый	1 л	45р	15
Яблочный	1 л	45р	8
Морсы			
Клюквенный	1,5 л	60р	7
Малиновый	1,5 л	60р	7
Рябиновый	1 л	45р	7
…			

Как видно из примера, таблица легко воспринимается визуально. Однако в таком виде она неприменима в составе баз данных. Такая таблица с небольшим числом строк без особых усилий может быть преобразована в реляционную таблицу вручную. Но в реальных таблицах может быть тысячи или более строк. В этом случае их преобразование не очевидно. Необходимы специальные автоматизированные средства.

В случае если таблица 3 будет импортирована в какую-либо систему управления базами данных она примет вид (таблица 4):

Таблица 4

Название	Обьем	Цена	Количество
Газированная вода			
Тархун	1 л	30р	7
Байкал	2 л	60р	5
Колокольчик	1,5 л	40р	10
Соки			
Вишневый	1 л	45р	3
Ананасовый	1 л	45р	15
Яблочный	1 л	45р	8
Морсы			
Клюквенный	1,5 л	60р	7
Малиновый	1,5 л	60р	7
Рябиновый	1л	45р	7

Как видно из таблицы 4 смысл импортированной таблицы утрачен. В частности из таблицы следует, что напиток Газированная вода не имеет объема, цены и не продавался. Хотя на самом деле все обстоит по-другому. Необходимо избавиться от противоречий такого рода. Для этого:

- Нужно выявить внутренние подзаголовки.
- Исключить соответствующие строки.
- Сформировать реляционную таблицу или таблицы таким образом, чтобы сохранить смысл исходных таблиц.

Информация табличного вида нередко представлена таким образом, что в области заголовков имеют место заголовки, которые включают в себя несколько позиций. Например, какой-либо заголовок может быть представлен следующим образом: континент, часть света, страна. Таблица такого рода не является реляционной и преобразование ее к реляционному виду задача нетривиальная. Более того, проблема преобразования может существенно

усложниться в связи с тем, что такого рода заголовки влекут за собой необходимость использования сложных заголовков. Например (таблица 5):

Таблица 5

Континент, часть света, страна		Количество крупных городов	Количество крупных рек
Европа			
Восток	Запад		
Россия		33	26
	Испания	8	14

И в первом и втором случаях нарушена первая нормальная форма – атрибуты реляционных таблиц должны быть неделимы. Кроме того, такого рода подзаголовки могут встречаться внутри таблицы. Пример такого рода таблицы – таблица 6.

Таблица 6

Континент, часть света, страна		Количество крупных городов	Количество крупных рек
Европа			
Восток	Запад		
Россия		33	26
	Испания	8	14
Азия			
Восток	Запад		
Мьянма		14	4
	Индия	26	8

Такую таблицу невозможно обрабатывать с помощью языка запросов.

В связи с этим эту таблицу оправданно представить в виде 2-х связанных реляционных таблиц: «Части света. Континенты» и «Страны».

Но в этой таблице существуют сложные атрибуты, поэтому надо избавляться от сложных атрибутов, избавляться от подзаголовков, которые попали в значения атрибутов. В таблице 7 показаны эти изменения.

Таблица 7

Континент, часть света, страна		Количество крупных городов	Количество крупных рек
Восточная Европа	Западная Европа		
Россия		33	26
	Испания	8	14
Восточная Азия	Западная Азия		
Мьянма		14	4
	Индия	26	8

Предлагается следующая последовательность действий. Формируется новый столбец с номерами частей света, континентов. Сканируется преобразованная таблица, очередной части света, континенту присваивается номер и этот номер распространяется на страны. Но эта таблица не очень удачна для 2-х связанных реляционных таблиц: «Части света. Континенты» и «Страны». Поэтому предлагается следующий вариант (таблица 8).

Таблица 8

Континент, часть света, страна	Количество крупных городов	Количество крупных рек
Восточная Европа		
Россия	33	26
Западная Европа		
Испания	8	14
Восточная Азия		
Мьянма	14	4
Западная Азия		
Индия	26	8

Результат формирование нового столбца с номерами части света, континентов и заполнением столбца приведен в таблице 9.

Таблица 9

№	Континент, часть света, страна	Количество крупных городов	Количество крупных рек
1	Восточная Европа		
1	Россия	33	26
2	Западная Европа		
2	Испания	8	14
3	Восточная Азия		
3	Мьянма	14	4
4	Западная Азия		
4	Индия	26	8

Результат формирования новой таблицы «Части света. Континенты» и исключения записей с «частью света, континентов» из исходной таблицы приведены соответственно в таблице 10. В таблице 11 - Страны.

Таблица 10

№	Континент, часть света
1	Восточная Европа
2	Западная Европа
3	Восточная Азия
4	Западная Азия

Таблица 11

№	страна	Количество крупных городов	Количество крупных рек
1	Россия	33	26
2	Испания	8	14
3	Мьянма	14	4
4	Индия	26	8

Для таблиц данного вида таблиц можно строить реляционные запросы.

Затем из записей с «частью света, континентов» формируется новая таблица, соответствующие записи из исходной преобразованной таблицы удаляются. После этих преобразований будут сформированы 2-е таблицы, связанные между собой связью типа 1: ∞.

Литература:

1. Дейт К., Дж. Введение в системы баз данных. 8-е изд.: Пер. с англ.- М.: Вильямс, 2005. - 1328 с.

2. Балдин А.В., Брешенков А.В. Анализ проблемы проектирования реляционных баз данных на основе использования существующей информации табличного вида. Вестник Московского государственного технического университета – 2007. – №4. – С. 43-52.

3. Брешенков А.В., Гудзенко Д.Ю., Казаков Г.И. Проектирование реляционных баз данных на основе информации табличного типа. Учебное пособие- М.: Изд-во МГТУ им. Н.Э. Баумана, 2009 - 150 с.

Сборник трудов кафедры ИУ6
МГТУ им. Н.Э. Баумана

СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

редактор сборника *В.В. Сюзев*
ответственные за выпуск *Б.И. Ващенко*
компьютерная верстка *В.Г. Перепелицына*

Подписано в печать 29.04.2011. Формат 60x84/16.
Усл. печ. л. 11,25. Тираж 100 экз. Заказ

105005, Москва, 2-я Бауманская ул., д. 5
НИИ РЛ МГТУ им. Н.Э. Баумана