ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

11(195) 2012

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с ноября 1995 г.

УЧРЕДИТЕЛЬ
Издательство "Новые технологии"

СОДЕРЖАНИЕ

Информация о журнале доступна по сети Internet по адресу **http://novtex.ru/IT.**
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

УДК 004.942

Н. А. Князев, аспирант[1], инженер-практикант[2],
e-mail: nickolai.a.knyazev@intel.com,
К. К. Малинаускас[2],
канд. физ.-мат. наук, науч. сотр.,
e-mail: kostas.malinauskas@intel.com
[1] ИПМ им. М. В. Келдыша РАН,
[2] ЗАО "Интел А/О"

Алгоритмы поиска критических путей в задаче статического временного анализа СБИС

Рассматриваются алгоритмы поиска критических путей сигналов, используемые в статическом временном анализе (СВА) синхронных цифровых схем. Для содержащих миллионы транзисторов СБИС алгоритмы СВА могут работать несколько суток, что существенно осложняет разработку и отладку схем. Поэтому в статье особое внимание уделяется возможности параллелизации методов и инкрементальным методам СВА. Рассмотрены известные подходы к параллелизации СВА и методы повторного анализа схемы после ее редактирования. Подходы сравниваются с точки зрения применимости к различным схемам, оценивается эффективность параллелизации.

Ключевые слова: *параллельные алгоритмы, инкрементальные алгоритмы, поиск критических путей в графе, статический временной анализ*

Введение

Статический временной анализ (СВА) — распространенный подход для оценки быстродействия синхронных интегральных цифровых схем. В таких схемах полное моделирование работы на физическом уровне слишком трудоемко, и поэтому СВА получил широкое распространение на практике.

Различают статический и динамический временной анализы схем [1]. При динамическом анализе проводится полное моделирование переходных процессов при различных возможных вариантах переключений входных сигналов. Динамический анализ позволяет получать наиболее точные оценки, однако время его работы слишком велико, и на практике применяют статический анализ. В статическом временном анализе задержки вычисляются путем упрощенного моделирования отдельных элементов схемы и их соединений с использованием сценариев худшего случая распространения сигналов. В качестве результата СВА принято рассматривать набор критических путей во временном графе [2, 3], определяющих быстродействие схемы.

В статическом временном анализе схема представляется в виде взвешенного направленного графа. Для расчета задержки на уровне отдельных логических элементов, как правило, используют метод Элмора [4] или асимптотические методы [3, 5, 6]. На уровне схемы работает алгоритм анализа взвешенного графа. В статье предлагается обзор алгоритмов поиска критических путей на временном графе, используемых в СВА. Эти алгоритмы несколько отличаются от классических алгоритмов поиска путей наибольшей (наименьшей) длины (Дейкстры, Беллмана—Форда и др.), известных в теории графов, поскольку задача СВА имеет свою специфику [7—9].

Современные СБИС содержат миллионы логических элементов, а время их анализа может занимать более суток [10]. В такой ситуации важную роль в СВА начинают играть инкрементальные и параллельные алгоритмы. Инкрементальные алгоритмы анализируют схему после ее редактирования и используют данные анализа исходной схемы для ускорения СВА. Существенно ускорить СВА может также параллелизация методов. К сожалению, при создании многих существующих методов эта возможность не учитывалась; в данном обзоре параллелизации уделено особое внимание.

Одной из проблем СВА является выявление "ложных путей", т. е. путей, которые не реализуемы ни в одном из сценариев нормальной работы схемы. Эта проблема подробно рассмотрена в работах [11—13], однако предлагаемые алгоритмы редко используются в коммерческих САПР из-за большой сложности задачи [12]. Другая проблема — это проблема выбора k критических путей [11, 14, 15]. Большинство алгоритмов предполагает вывод только критических (худших по своим временным характеристикам) путей в графе. Однако инженерам часто требуется знать некоторое число околокритических путей в данном узле. В работе [16] представлен также полиномиальный алгоритм выделения множества путей в схеме с задержками, превышающими некоторое заранее заданное пороговое значение.

Данный обзор посвящен базовым алгоритмам поиска критических путей в задаче статического временного анализа, включая инкрементальные и параллельные.

Основные понятия

Сигналы распространяются по схеме через логические элементы и межсоединения. На прохождение каждого участка схемы сигнал затрачивает время, называемое задержкой. В зависимости от электрического напряжения в данном узле состояние сигнала может соответствовать нижнему ("0") и верхнему ("1") уровню. Для постановки задачи статического временного анализа введем несколько определений.

Событие — изменение уровня сигнала в данном узле схемы.

Тип события — как правило, различают нисходящие (соответствуют изменению уровня сигнала с 1 на 0) и восходящие (изменение с 0 на 1) события.

Временной граф цифровой схемы — направленный граф, где в качестве вершин выступают узлы схемы (входы и выходы элементов), а направленные дуги соответствуют причинно-следственным связям между возможными событиями в соседних узлах. Каждой дуге сопоставляется вес, равный времени распространения события вдоль нее. Для простоты в дальнейшем будем считать, что данное время не зависит от типа события и длительности его фронта.

Задержка события — время наступления события относительного некоторого начала отсчета.

Для СВА различают анализ "сверху" и анализ "снизу" [1, 17]. При анализе "сверху" анализируются наибольшие, а при анализе "снизу" — наименьшие задержки распространения событий. По умолчанию рассматривается анализ "сверху" (все аналогичные выводы можно также повторить и для анализа "снизу").

Путь — последовательность событий, где одно событие порождает следующее в соседнем узле. При этом задержка следующего события складывается из задержки предыдущего события и задержки внутри элемента или межсоединения. *Задержкой пути, заканчивающегося данным событием,* называется задержка этого события.

Критический путь — путь с наихудшей задержкой события данного типа в данном узле схемы. Для простоты в дальнейшем при вычислении критических путей будем игнорировать тип распространяемых событий и обозначим задержку худшего события в каждом узле V_i как $LAT(V_i)$ (*Latest Arrival Time*).

Запас — разность между фактической задержкой события (пути) и задержкой, предельно допустимой для корректной работы схемы. В случае анализа "сверху" предельно допустимые задержки определяются тактовой частотой схемы. Отрицательный запас соответствует нарушению корректной работы схемы [11].

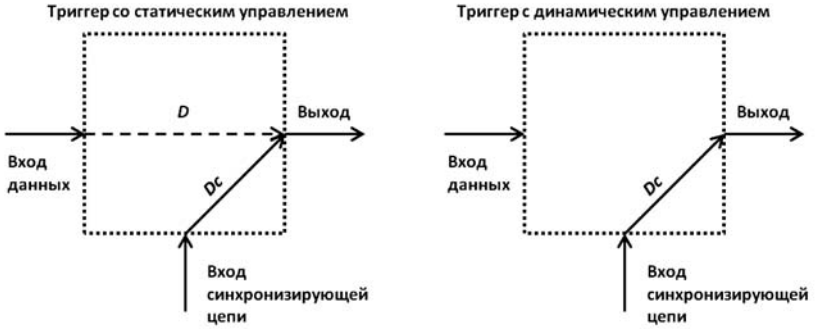
Отрицательный путь — путь, имеющий отрицательный запас.

Синхронный триггер (далее просто *триггер*) — логический элемент, имеющий два входа (данных и синхронизации) и один выход.

Триггер с динамическим управлением (*flip-flop, с управлением по фронту*) — триггер, в котором состояние сигнала со входа на выход передается в момент возникновения управляющего события синхронизирующей цепи (восходящего или нисходящего фронта). Поскольку выходное событие порождается событием синхронизации, то во временном графе присутствует дуга от входа синхронизации к выходу.

Триггер со статическим управлением (*latch, или с управлением по уровню*) — триггер, в котором условием передачи сигнала служит состояние входа синхронизации (0 или 1), определяющее "прозрачность" триггера. Триггер прозрачен (открыт, *transparent*) между открывающим и закрывающим событиями синхронизирующей цепи, в остальное время триггер непрозрачен (закрыт). Обозначим задержки событий на входе и выходе триггера как *Arr* и *Dep* соответственно. Задержки открывающего и закрывающего событий синхронизирующей цепи обозначим как *Clk_Open* и *Clk_Close*. Обозначим время распространения события со входа данных на выход как *D*, а время распространения события со входа синхронизации на выход — как *Dc* (рис. 1).

Если событие на входе данных наступило между открывающим и закрывающим событиями, т. е. триггер прозрачен ($Clk\_Open \leqslant Arr \leqslant Clk\_Close$), то сигнал передается на выход немедленно. Если событие на входе данных произошло, когда триггер непрозрачен, то сигнал задерживается и передается на выход одновременно с очередным открывающим событием. Поэтому во временном графе присутствуют две дуги: от входа данных к выходу и от входа синхронизации к выходу триггера. Временная диаграмма для триггера со статическим управлением изображена на рис. 2.



Триггер со статическим управлением: событие на выходе порождается событием на входе данных, если триггер прозрачен, и событием на входе синхронизирующей цепи в обратном случае.

Триггер с динамическим управлением: событие на выходе порождается событием на входе синхронизирующей цепи.

Рис. 1. Временной граф для триггера со статическим управлением и триггера с динамическим управлением

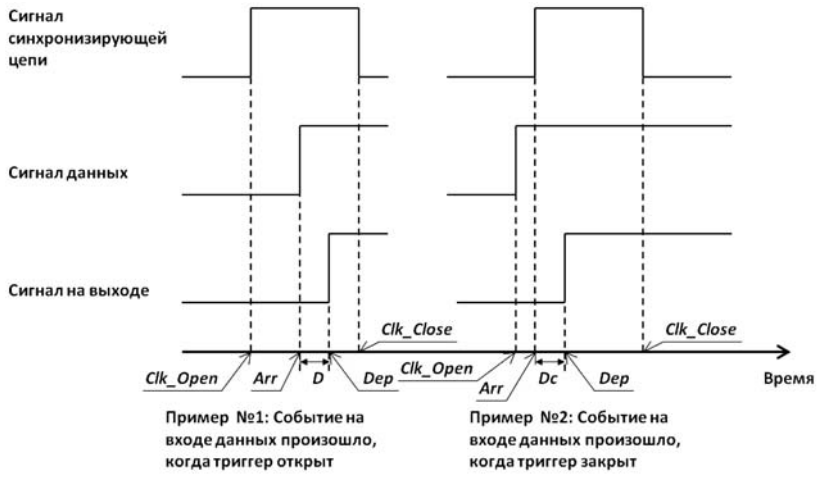


Рис. 2. Временная диаграмма работы триггера со статическим управлением

В общем случае временной граф содержит циклы. Введем определения различных типов циклов в графе.

Комбинационный цикл — цикл, состоящий только из устройств комбинационной логики и, возможно, триггеров с динамическим управлением. Такие циклы обычно являются ошибкой проектирования схемы и не требуют СВА.

Последовательный цикл — цикл, содержащий хотя бы один триггер со статическим управлением, такие циклы могут являться корректными с точки зрения логики схемы и требуют СВА.

Критический цикл — замкнутый критический путь.

В общем случае результат СВА — это набор всех путей и их запасов. Число путей может экспоненциально зависеть от размера схемы, поэтому, как правило, рассматривают анализ "худшего случая" (*worst-case analysis*) [1, 11], где в качестве результата выступает набор худших (критических) путей. Данный подход подробно описан в [18]. В качестве входных данных для СВА выступает набор событий на входах схемы, при распространении событий используется *прореживание*: в каждом узле схемы выбираются события с наихудшей (наибольшей или наименьшей) задержкой, и дальше по схеме распространяются только эти события.

При вычислении задержек внутри элемента в общем случае необходимо учитывать следующие характеристики:

- задержки событий на всех входах элемента;
- длительность фронтов переключений (время между началом и окончанием изменения уровня). Длительность фронта переключения входного события влияет на задержку внутри элемента и межсоединения. Стоит отметить, что событие с "пологим" фронтом переключения и малой задержкой на выходе схемы может "накопить" задержку больше, чем событие с большей задержкой, но с "крутым" фронтом переключения [19];
- тип событий. Некоторые пути в элементе могут быть доступны только для событий определенного типа;
- различные параметры и условия работы схемы (температура, создающие помехи соседние элементы и т. д.).

Традиционные подходы к статическому временному анализу

Статический временной анализ ациклических схем. Обычно в правильно спроектированной комбинационной схеме отсутствуют циклы. Временной граф последовательной схемы, где все триггеры управляются динамически, также не содержит циклов. Анализ ациклического графа существенно упрощает тот факт, что можно использовать топологический порядок обработки узлов — от входов к выходам.

В работе [11] приводится обобщенный алгоритм СВА для ациклического графа, который можно представить так:

```
Вход: список начальных элементов.
Выход: худшие задержки LAT во всех узлах схемы.
1. Проинициализировать LAT в каждом узле
   схемы значением −∞.
2. Для каждого элемента g схемы из списка
   (в порядке от входов к выходам) выполнить:
   2.1. Распространить события со входов
        элементов на выходы, вычислить LAT
        на выходах, выполняя прореживание.
   2.2. Распространить события с выходов
        элементов на входы следующих элемен-
        тов, вычислить LAT на входах, выпол-
        няя прореживание.
```

Когда определены худшие задержки во всех узлах схемы, можно восстановить критические пути, вычислить их запасы и определить пути с нарушениями.

Параллельные алгоритмы для комбинационных схем представлены в работах [20, 21] и рассмотрены ниже.

Статический временной анализ схем с циклами, триггер-граф. В последовательных схемах, содержащих триггеры со статическим управлением, могут встречаться циклы. В случае присутствия критических циклов задача поиска критических путей становится неопределенной: задержки в цикле могут накапливаться сколь угодно много, приводя, в конечном счете, к нарушениям при любой тактовой частоте. На практике достаточно идентифицировать такие циклы и прервать СВА или продолжить анализ с разрывом цикла (запрещая повторное распространение события с большей задержкой, см., например, [22]). Предполагается, что инженер в дальнейшем проанализирует критические циклы

и в случае необходимости скорректирует схему. В СВА учитывается также тот факт, что обычно в любой корректной схеме в каждом цикле присутствует хотя бы один триггер со статическим управлением.

В некоторых работах предлагается рассматривать вместо полного временного графа так называемый триггер-граф, где вершинами являются триггеры, а вес ребер между триггерами $delay(V_i, V_j)$ равен суммарной задержке во временном графе между ними. Сложность построения триггер-графа для временного графа G, имеющей N триггеров, равна $O(N|G|)$, где $|G|$ — число вершин в графе G [22, 23]. Пример триггер-графа изображен на рис. 3.

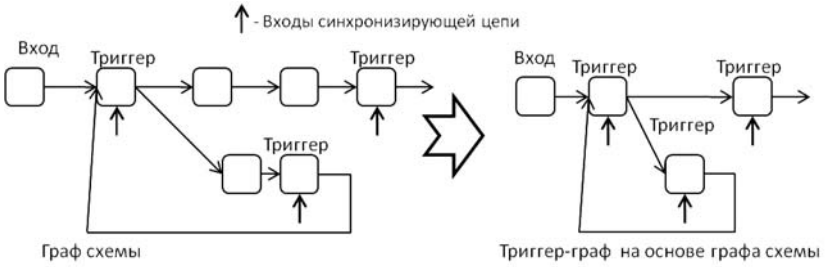


Рис. 3. Пример триггер-графа

Фазовый сдвиг [22]. Задержки всех распространяемых в СВА событий отсчитываются относительно синхронизирующих событий "идеального" генератора тактовых импульсов. В представленных ниже алгоритмах задержка события, распространяемого со входа данных на выход прозрачного триггера V_i, корректируется, изменяя точку отсчета на точку отсчета закрывающего события (Clk_Close_i) в цепи синхронизации данного триггера: $Arr_i + D_i - \Delta_{ji}$, где Δ_{ji} — оператор фазового сдвига между триггером V_i и триггером V_j — источником события Arr_i (подробнее см. [22]).

Простой алгоритм на триггер-графе. В работе [22] предлагается простейший алгоритм расчета триггер-графа, основанный на алгоритме Форда—Беллмана [24]:

Вход: временной граф схемы.
Выход: список нарушений.
1. Для всех выходов триггеров V_i установить $ArrPrev_i = Arr_i = Dep_i = -\infty$.
2. Для каждого триггера вычислить худшую задержку на выходе: $Dep_i = \max(ArrPrev_i + D_i - \Delta_{ji}, Clk\_Open_i + Dc_i)$.
3. Для каждого триггера вычислить худшую задержку на входе: $Arr_i = \max\{Dep_j + delay(v_j, v_i)|$ по всем входящим ребрам $(v_i, v_j)\}$.
4. Если хотя бы для одного триггера $ArrPrev_i \neq Arr_i$ и число итераций меньше числа триггеров, то установить $ArrPrev_i = Arr_i$ и перейти на п. 2.
5. Вычислить запасы и вывести нарушения.

Прерывание по числу итераций на шаге 2 используется для предотвращения зацикливания алгоритма в случае наличия критических циклов. Нарушения, соответствующие отрицательным путям и циклам, не различаются и выводятся вместе.

Корректировка задержек. Рассмотрим случай, когда на триггере со статическим управлением задержка события на входе больше задержки закрывающего события ($Arr > Clk\_Close$), т. е. нарушена корректная работа схемы. В этом случае иногда считают, что событие на входе наступило одновременно с закрывающим событием. Это делается для того, чтобы не прерывать распространение "опоздавшего" события, а продолжить распространение в предположении, что нарушение исправлено. Таким образом, задержка события на выходе элемента корректируется: $Dep = \min(\max(Arr, Clk\_Open), Clk\_Close)$.

В работе [22] корректировка задержек представляется добавлением между триггером и следующими за ним элементами еще одного элемента, так называемой min-вершины, этот элемент передает событие с задержкой, не большей Clk_Close (см. рис. 4). В остальном используется простой алгоритм на триггер-графе, описанный выше.

Рис. 4. Пример min-вершины

Алгоритм распространения от триггеров. В работе [22] предлагается алгоритм с обнаружением циклов на триггер-графе с помощью построения путей, начиная от триггеров. На i-м шаге алгоритма для каждого триггера выбирается худший путь длины $i - 1$ (длина пути равна числу триггеров, входящих в этот путь), заканчивающийся на данном триггере. Если данный путь имеет задержку большую, чем открывающий сигнал ($Arr > Clk\_Open$, т. е. триггер прозрачен), то строится путь длины i путем дальнейшего распространения события через триггер на его выход и до следующих триггеров. Алгоритм имеет вид:

Вход: граф схемы.
Выход: список критических путей и циклов, список нарушений.
1. Всем триггерам ставим в соответствие путь нулевой длины, начинающийся и заканчивающийся в этом триггере, устанавливаем текущую длину путей $i = 1$.
2. Выполнять следующие действия, пока есть путь длины $i - 1$:

2.1. На входе каждого триггера L найти худшее событие, порождаемое путем длины $i - 1$, заканчивающимся в триггере, соединенным с входом данного.
2.2. Если данное событие имеет задержку, большую Clk_Open, выполнить:
2.2.1. Если этот путь не содержит триггер L ранее, то расширить путь до длины i, включив триггер L в путь. В противном случае сообщить об обнаруженном цикле.
3. Вычислить запасы и вывести пути с нарушениями и циклы.

Сравнение алгоритмов на графах с циклами. В работе [22] приводятся результаты реализации алгоритмов на основе триггер-графа, сравнивается работа алгоритмов на одинаковых схемах с несколькими тысячами триггеров. Самым быстрым является алгоритм распространения от триггеров, он работает около 5...6 с. Простейший алгоритм с корректировкой задержек работает около 9...14 с. При этом без использования корректировки задержек выполнение простого алгоритма на триггер-графе занимает несколько часов. Отметим, что алгоритм распространения от триггеров потенциально может быть реализован в параллельном варианте, так как поиск худших путей заданной длины можно выполнять параллельно. Однако авторы вопрос параллелизации не рассматривают.

Инкрементальные алгоритмы

Инкрементальные алгоритмы решают задачу пересчета задержек в схеме после изменения (добавления или исключения) нескольких элементов с использованием уже имеющейся информации о критических путях. Данные алгоритмы призваны существенно экономить время, не пересчитывая задержки полностью.

Назовем множество элементов в направленном графе, достижимых из данного, *выходным конусом элемента*. Объединение выходных конусов изменившихся элементов назовем *конусом изменений*. В работе [25] доказывается, что задержки после изменения нескольких элементов изменятся только в конусе изменений, и поэтому расчет задержек можно сократить только до элементов, входящих в конус изменений.

Упрощенный инкрементальный алгоритм. На практике, если граф изменился незначительно, часто допускают, что структура критических путей не изменилась. В этом случае выполнять повторно СВА всей схемы нет необходимости. Для отражения этого изменения достаточно вычислить заново задержки во всех критических путях, проходящие через измененный элемент. Инкрементальный перерасчет конуса занимает существенно меньше времени, чем СВА всей схемы. Простой инкрементальный алгоритм, по сути, представляет собой простой обход графа критических путей. Обход графа поиском в ширину хорошо распараллеливается [21]. Авторами в работе [20] предлагается реализация простого инкрементального алгоритма, с помощью которой на 16 потоках удалось достичь 13-кратного ускорения. Сложность этого алгоритма линейна по отношению к числу событий в конусе, при использовании прореживания эта сложность равна $O(N)$, где N — число элементов в конусе изменений.

Точный инкрементальный алгоритм. Авторы [25] отдельно рассматривают случай увеличения и уменьшения задержек. Для общего случая предлагается смешанный алгоритм.

Случай увеличения задержек. Когда задержки событий на всех измененных элементах увеличились, доказывается, что остальные задержки также могут только увеличиться и только внутри конуса изменений. Для данного случая авторы приводят следующий алгоритм:

Вход: множество ребер, увеличивших свой вес E^+.
Выход: позднее время прибытия сигнала для каждой вершины графа схемы.
1. Произвольно пронумеровать вершины графа схемы внутри конуса (сопоставляем каждому узлу V_i число $bfs(V_i)$); назовем ребра положительными, если они ведут из вершины с меньшим номером в вершину с большим номером, иначе отрицательными.
2. Установить счетчик $m = 0$, добавить в очередь Q^0 все вершины на изменившихся ребрах графа схемы.
3. Пока очередь Q^m не пуста, выполнить:
3.1. Извлечь вершину V_i из начала очереди, для всех ребер (V_i, V_j), исходящих из этой вершины, выполнить:
a) если $bfs(V_i) < bfs(V_j)$ (ребро положительное) и $LAT(V_i) + delay(V_i, V_j) > LAT(V_j)$, обновить значение $LAT(V_j)$ и граф худших путей, добавить узел V_j в конец очереди Q^m.
b) если $bfs(V_i) > bfs(V_j)$ (ребро отрицательное) и V_j отсутствует в очереди Q^{m+1}, то добавить узел в начало очереди Q^{m+1}.
4. Если очередь Q^m пуста, то увеличить счетчик m на 1, если очередь Q^{m+1} пуста, то закончить алгоритм, иначе перейти на шаг 2.

Авторы алгоритма утверждают, что обычно до завершения алгоритма значение счетчика не достигает 10. В противном случае имеет смысл осуществить проверку на циклы в графе критических путей (достаточно осуществить поиск обратным проходом).

Случай уменьшения задержек. Если в результате изменений веса ребер уменьшились, аналогично с предыдущим алгоритмом доказывается, что задержка в конусе изменений могла только уменьшиться.

Выбросим из временного графа ребра, которые не входят ни в один критический путь. Авторы доказали, что значения задержек в дереве критических путей вне конуса изменений не изменяются (так как к ним уже идет другой критический путь, а в данном случае задержка могла только уменьшиться). Алгоритм, предлагаемый для данного случая, не решает задачу полностью, а уменьшает задержки внутри конуса изменений и готовит данные для описанного в следующем разделе алгоритма общего случая.

Назовем ребра (V_i, V_j), где V_j принадлежит графу критических путей, а V_i не принадлежит, граничными ребрами. Алгоритм для случая уменьшения задержек будет выглядеть следующим образом:

Вход: множество ребер, уменьшивших свой вес E^-.
Выход: обновленные задержки на вершинах графа схемы, список ребер *Out*.

1. Отсортировать ребра конуса дерева худших путей в соответствии с топологическим порядком.
2. Для каждого ребра (V_i, V_j) в топологическом порядке, если узел V_j не отмечен как посещенный:
   2.1. Создать очередь Q и положить в нее V_j.
   2.2. Пока очередь Q не пустая:
      2.2.1. Извлечь узел $\underline{V}_1$ из очереди и отметить его как посещенный, пусть в дереве худших путей в этот узел входит ребро ($\underline{V}_k$, $\underline{V}_1$).
      2.2.1. Пересчитать задержку сигнала в узле $\underline{V}_1$ как $LAT(\underline{V}_1) = LAT(\underline{V}_k) + delay(\underline{V}_1, \underline{V}_k)$.
      2.2.1. Для каждого входящего граничного ребра ($\underline{V}_m$, $\underline{V}_1$): если $LAT(\underline{V}_1) < LAT(\underline{V}_m) + delay(\underline{V}_m, \underline{V}_k)$, то обновить худшую задержку и граф худших путей.
      2.2.1. Для всех ребер ($\underline{V}_1$, $\underline{V}_k$) на графе худших путей добавить в очередь узел $\underline{V}_k$, если он еще не посещен.
3. Для всех ребер ($\underline{V}_i$, $\underline{V}_j$), не входящих в граф худших путей, но входящих в конус изменений, если $LAT(V_j) < LAT(V_j) + delay(V_i, V_j)$, то добавить ($V_i$, V_j) в список *Out* (этот список используется в алгоритме для общего случая, описанном ниже).

Общий случай. Для общего случая, когда среди измененных ребер графа есть ребра и увеличившие, и уменьшившие свой вес, предлагается общий алгоритм:

Вход: набор изменившихся ребер.
Выход: обновленные задержки на вершинах графа схемы.

1. В список E^- добавить все ребра, уменьшившие свой вес. В список E^+ добавить все ребра, увеличившие свой вес.
2. Выполнить алгоритм для случая уменьшения задержки: на вход алгоритму передать E^-, считая оставшиеся веса неизменными и сформировать список *Out*. Очистить список E^-.
3. Выполнить алгоритм для случая увеличения задержек, подавая на вход объединение списков E^+ и *Out*. Очистить список E^+.
4. Для каждого найденного на шаге 3 цикла разорвать его ребро (установить задержку на этом ребре $-\infty$) и добавить это ребро в E^-.
5. Если E^- пуст, закончить алгоритм, иначе перейти на шаг 2.

Авторами доказывается, что сложность каждой итерации алгоритма не превосходит $O(N)$, где N — число элементов в конусе изменений. Таким образом, сложность всего алгоритма равна $O(N^*N_c)$, где N_c — число циклов в схеме.

Сравнение инкрементальных алгоритмов. Упрощенный инкрементальный алгоритм рассматривает случай, когда структура критических путей в схеме не изменилась, что не всегда исполняется; таким образом, простой алгоритм не всегда вычисляет правильно худшие задержки, однако хорошо распараллеливается и масштабируется и используется на усмотрение пользователя. Точный инкрементальный алгоритм выдает правильный результат СВА. К сожалению, в нем используется линейный порядок вершин, что делает параллелизацию невозможной. Однако есть работы по замене линейного порядка частичным [21], что может позволить извлечь ограниченную пользу от использования многопоточности.

Подходы к параллелизации СВА

Синхронная параллелизация с использованием ранжирования. Многие современные методы обхода временного графа в СВА предусматривают анализ вершин в заранее определенном порядке (основанном на топологическом). Наличие топологического порядка гарантирует, что перед анализом элемента события на входах элемента будут рассчитаны, однако ограничивает применимость методов ациклическими графами. В [26] предлагается для параллелизации использовать ранжирование вершин графа по максимальной удаленности от входных элементов. Это позволяет независимо и параллель-

Рис. 5. Пример параллелизации с ранжированием

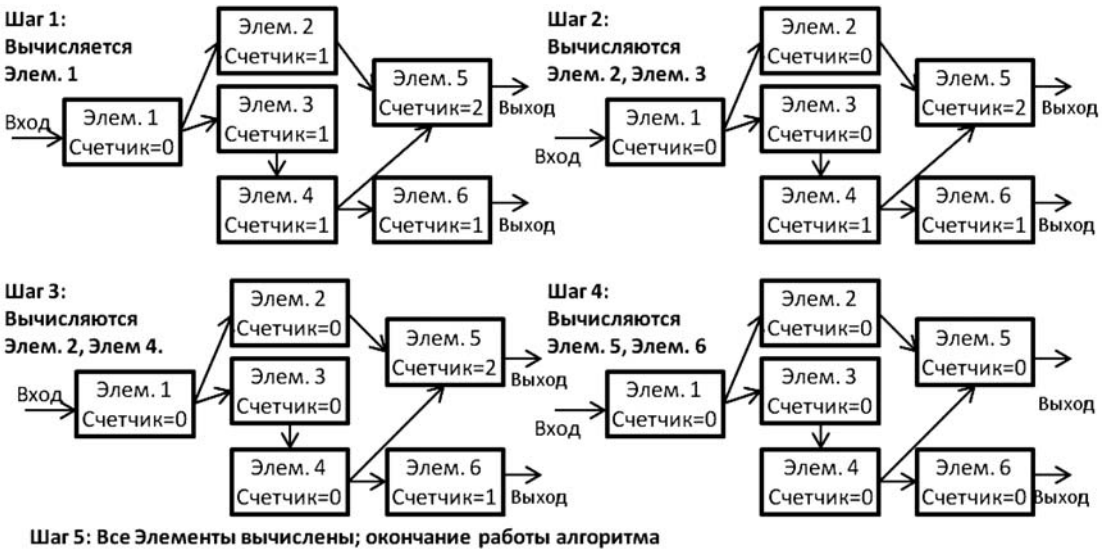


Рис. 6. Пример применения асинхронного подхода

но рассчитывать элементы на одном уровне с синхронизацией при переходе на следующий уровень.

Пример такого подхода показан на рис. 5. Недостатком подхода является то, что, если на моделирование элемента 2 будет тратиться больше времени, чем на моделирование элемента 3, то пока моделируется элемент 2, остальные вычислительные мощности будут простаивать. Пример ранжирования вершин по уровням приведен на рис. 5.

В [21, 26] представлена реализация данного подхода. В [26] исследуется работа алгоритма СВА, использующего ранжирование вершин на многопроцессорных системах. Авторы отмечают, что они сталкиваются с проблемой малого числа элементов на одном уровне даже на "широких" графах. В случае если на одном уровне оказывается малое число вершин, то загрузка процессоров осуществляется неравномерно, ускорение уменьшается. Однако авторам удалось достичь 100-кратного ускорения на 1024 процессорах для схемы, содержащей 3 млн элементов.

В [27] предлагается параллельный алгоритм СВА без использования прореживания событий. Однако предложенный алгоритм использует эвристику и не является точным.

Асинхронная параллелизация. В отличие от описанного выше "синхронного" подхода в [20] предлагается "асинхронная" параллелизация, когда элемент начинает моделироваться, как только для него готовы данные. Для обеспечения асинхронной параллелизации каждому элементу ставится в соответствие счетчик, соответствующий числу входов, на которых еще не готовы события. Изначально счетчики всех элементов, кроме начальных, устанавливаются равными числу входов. В процессе СВА значение счетчиков уменьшается. Когда значение счетчика равно 0, элемент готов к анализу. Данный подход дает возможность параллельно анализировать элементы со значением счетчиков, равным нулю. Алгоритм состоит из двух этапов (установка счетчиков и собственно временной анализ), каждый из которых можно выполнять в параллельном режиме. Основой каждого из этапов является обход графа, похожий на "поиск в ширину". Недостатком алгоритма является его применимость только для ациклических графов. Пример работы алгоритма приведен на рис. 6.

Реализация данного алгоритма предложена автором в [20]. В итоге удалось достичь более высокого максимального ускорения по сравнению с алгоритмами синхронной параллелизации, однако исследование ограниченно лишь многоядерными системами.

Сравнение подходов к параллелизации СВА. Стоит отметить, что данные подходы эффективно применимы только для "широких" временных графов

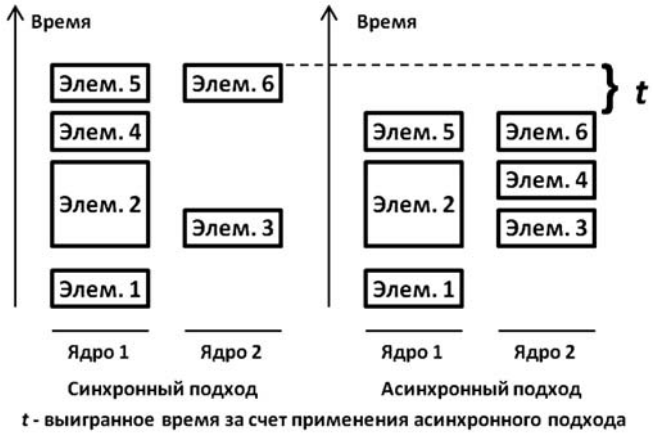


Рис. 7. Сравнение синхронного и асинхронного подходов

(где длина пути от входов до выходов значительно меньше числа вершин), иначе авторы [26] отмечают малую загрузку процессоров.

Несколько улучшить загрузку можно, используя асинхронный подход [20], за счет лучшей ее балансировки (рис. 7).

Выводы

Статический временной анализ — давно исследуемая область применения математических методов в САПР. В частности, разработаны эффективные алгоритмы для комбинационных и последовательных схем, в том числе содержащих циклы. На содержащих миллионы элементов СБИС традиционные алгоритмы СВА работают непозволительно долго. В некоторых случаях можно применять инкрементальные методы, однако и они не всегда работают достаточно быстро или дают неточный результат. Параллелизация могла бы существенно ускорить время работы. Однако когда разрабатывались алгоритмы временного анализа, многопоточного исполнения обычно не предполагалось. Применение методов синхронной и асинхронной параллелизации для описанных алгоритмов может существенно сократить время анализа цифровых схем. Эффективная параллелизация традиционных и инкрементальных алгоритмов СВА и их масштабирование для работы со сверхбольшими интегральными схемами являются на сегодняшний день открытой проблемой статического анализа цифровых схем.

Список литературы

1. **Hassoun S., Sasao T.** Logic synthesis and verification // Kluwer international series in engineering and computer science, 2002. Vol. 654. P. 373—401.

2. **Hitchcock B., Smith G. L., Cheng D. D.** Timing analysis of computer hardware // IBM Journal of Research and Development Journal. 1982. N 26. P. 100—105.

3. **Kirkpatrick T. I., Clark N.** PERT as an aid to logic design // IBM Journal of Research and Development Journal., 1966. N 10. P. 135—141.

4. **Elmore W. C.** The transient response of damped linear networks with particular regard to wideband amplifiers // Oak Ridge, Tenn.: Atomic Energy Commission. 1948. N 19.9. P. 569—571.

5. **Pillage L. T., Rohrer.** Asymptotic Waveform Evaluation for Timing Analysis // IEEE Transactions of CAD. 1990. N 9. P. 352—366.

6. **Pillage L. T., Ratzlaff C. L.** RICE rapid interconnect circuit evaluation // Proc. 28th ACM/IEEE Design Automation Conference. 1994. P. 763—776.

7. **Shenoy N. V., Brayton K., Sangiovanni-Vincentelli A. L.** Resynthesis of multi-phase pipelines // In Proc. Des. Automa. Conf. 1993. P. 490—496.

8. **Lockyear B., Ebeling C.** Optimal retiming of multi-phase level-clocked circuits // Proc. Adv. Res. VLSI Parallel Systems. 1992. P. 1249—1264.

9. **Ishii A., Leiserson C. E., Papaefthymiou M. C.** Optimizing two phase level-clocked circuitry // Adv. Res. VLSI Parallel Syst. 1992. P. 148—199.

10. **Scheffer L., Lavagno L.** EDA for IC implementation, circuit design, and process technology. New-York: CRC Press. 2006. Vol. 2. 608 p.

11. **Гаврилов С., Стемпковский А., Глебов А.** Методы логического и логико-временного анализа цифровых СБИС. М.: Наука, 2007. 224 с.

12. **Соловьев Р. А., Глебов А. Л., Гаврилов С. В.** Статический временной анализ с обнаружением ложных проводящих путей на основе логических импликаций // Всероссийская научно-техническая конференция "Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС)". Сб. тр. ИППМ РАН. 2006. С. 78—84.

13. **Матросова А. Ю., Кудин Д. В., Николаева Е. А.** Обнаружение ложных путей в комбинационной схеме // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2011. № 2. С. 99—107.

14. **Ganley J. L., Cohoon J. P.** Routing a multi-terminal critical net: Steiner tree construction in the presence of obstacles // Proc. IEEE Int. Symp. on Circuits and Systems. 1994. P. 113—116.

15. **Yen S., Du D., Ghanta S.** Efficient Algorithms for Extracting the K Most Critical Paths in Timing Analysis // Proc. Design Automation Conference (DAC). 1989. P. 649—654.

16. **Ahn K., Sahni S.** NP-hard module rotation problems // IEEE Transactions on Computers. 1993. N 42 (12). P. 1506—1510.

17. **Норенков И. П.** Основы автоматизированного проектирования: Учебник для вузов. М.: Изд. МГТУ им. Н. Э. Баумана, 2002. 336 с.

18. **Ирбенек В.** Временная верификация и оптимизация размещения компонентов предельных по быстродействию ЭВМ / Дис. на соиск. уч. ст. д-ра техн. наук, Институт микропроцессорных вычислительных систем РАН, 2001. 188 с.

19. **Blaauw S., Zolotov D., Sundareswaran V.** Slope Propagation in Static Timing Analysis // Proc. of the 2000 IEEE/ACM international conference on CAD. 2000. P. 338—343.

20. **Князев Н. А.** Статический временной анализ синхронных цифровых схем на многоядерных ЭВМ // Научный сервис в сети Интернет: экзафлопсное будущее: Тр. Междунар. суперкомпьютерной конф. (19—24 сентября 2011, г. Новороссийск). 2011. С. 617—623.

21. **Schultze P., Korf E.** Large-Scale Parallel Breadth-First Search // Proc. AAAI'05 Proceedings of the 20th national conference on Artificial intelligence, 2005. Vol. 3. P. 1380—1385.

22. **Burks T. M., Sakallah K. A., Mudge T. N.** Critical Paths in Circuits with Level-Sensitive Latches // IEEE Transactions on Very Large Scale Integration (VLSI) systems. 1995. N 3. P. 273—291.

23. **Lee J., Tang D. T., Wong C. K.** Timing Analysis Algorithm for Circuits with Level-Sensitive Latches // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 1996. N 15. P. 535—543.

24. **Bellman R.** On a routing problem // Quarterly of Applied Mathematics, 1958. N 16 (1). P. 87—90.

25. **Jin-fuw L., Tang D. T.** An Algorithm for Incremental Timing Analysis // Proc. of the 32nd annual ACM/IEEE Design Automation Conference. 1995. P. 696—701.

26. **Holder A., Christopher D. C., Kalafala K.** Prototype for a Large-Scale Static Timing Analyzer running on an IBM Blue Gene // Proc. IPDPS Workshops. 2010. P. 1—8.

27. **Li-Ren C., Liu D., Hsi-Chuan D.** An Efficient Parallel Critical Path Algorithm // Proc. of the 28th ACM/IEEE Design Automation Conference. 1991. P. 535—540.

28. **Chadha J., Bhasker.** Static Timing Analysis for Nanometer Designs, A Practical Approach. New-York: Springer, 2009. 572 p.

29. **Dijkstra E.** A note on two problems in connection on graphs // Numerische Mathematic. 1959. N 1. C. 269—271.

30. **Hadley S. W., Mark B. L., Vanelli A.** An efficient eigenvector approach for finding netlist partitions // Trans. on Computer-Aided Design. 1992. N 11. P. 885—892.

31. **Tsay R., Lin I.** Robin Hood: A System Timing. Verifier for Multi-Phase Level Sensitive Clock Designs // Proc. Fifth Annual IEEE International IBM. 1992. P. 516—519.

УДК 629.73.533.6

Е. С. Чувашева[1], аспирант,
e-mail: chuvashevalena@gmail.com,
С. Н. Чувашев[2], д-р физ.-мат. наук, проф.,
e-mail: snchuv@mail.ru,
И. Г. Зорина[3], канд. техн. наук, доц.,
e-mail: zorina.ig@gmail.com
[1] Институт философии РАН
[2] "МАТИ" — РГТУ имени К. Э. Циолковского
[3] МГТУ им. Н. Э. Баумана

Комплексная математическая модель для концептуального проектирования высокоскоростных летательных аппаратов

Разработана комплексная модель, описывающая разнородные физические процессы в узлах высокоскоростных летательных аппаратов. Учитывается динамика полета, внешняя и внутренняя (двигатель) аэродинамика, распределение сил давления и трения, аэродинамический нагрев поверхности, распространение тепловых потоков по узлам аппарата, законы управления, в том числе ручное в ходе полета, и др. Разработанная прикладная программа предназначена для проверки технических решений при концептуальном проектировании.

Ключевые слова: *высокоскоростные летательные аппараты, теплообмен, динамика полета, комплексные модели*

Введение

В настоящее время в различных странах интенсивно разрабатываются высокоскоростные летательные аппараты (ЛА), предназначенные для движения в верхней атмосфере со скоростями, превышающими скорость звука в 4...7 и более раз. В связи со сложностью и дороговизной наземных экспериментов большую роль в их разработке играет математическое моделирование. Наряду с работами по подробному моделированию отдельных узлов и рабочих процессов развивается направление комплексного моделирования, в котором рассматриваются сразу ряд аспектов и несколько узлов (см., например, [1—5]). Его актуальность связана с большой степенью новизны высокоскоростных ЛА и сложности интуитивного нахождения оптимального концептуального облика ЛА. Для более обоснованного решения этой задачи необходимо описать в единой комплексной математической модели различные аспекты функционирования и взаимодействия подсистем ЛА и взаимодействия ЛА с другими объектами при выполнении заданных миссий.

Представленный в данной работе программный комплекс реализует физическую и математическую комплексную модель ЛА, которая одновременно описывает, по-видимому, существенно больше аспектов, чем в любой из указанных выше работ.

Физическая и математическая постановка

Динамика полета подробно описывается на основе решения системы уравнений для трехмерного движения твердого тела. Моделируется полет гиперзвукового ЛА в атмосфере в системе координат X, Y, Z, связанной с Землей (ось OZ направлена вверх). Решается система уравнений для общего случая трехмерного движения ЛА:

$$M\partial w_x/\partial t = A_x;$$

$$M\partial w_y/\partial t = A_y;$$

$$M\partial w_z/\partial t = A_z - Mg;$$

$$\partial X/\partial t = w_x;$$

$$\partial Y/\partial t = w_y;$$

$$\partial Z/\partial t = w_z;$$

$$\partial M/\partial t = -m_1;$$

$$A_z = A_\tau \sin\beta_w + A_n \cos\theta \cos\beta_w;$$

$$A_X = A_\tau \cos\beta_w \cos\varphi - A_n \cos\theta \cos\beta_w \cos\varphi + A_n \sin\theta \sin\varphi;$$

$$A_Y = A_\tau \cos\beta_w \sin\varphi - A_n \cos\theta \sin\beta_w \sin\varphi - A_n \sin\theta \cos\varphi;$$

$$A_\tau = P\cos\alpha - F_L,\ A_n = F_Z + P\sin\alpha;$$

$$F_L = C_L S\rho w^2/2,\ F_Z = C_Z S\rho w^2/2,\ w^2 = w_x^2 + w_y^2 + w_z^2,$$

где M — масса аппарата, равная сумме массы конструкции и массы топлива; g — ускорение свободного падения; вектор скорости w составляет угол β_w с горизонтом, а его проекция на ось OXY — угол φ; θ — угол крена; α — угол атаки; S — площадь поперечного сечения ЛА; ρ — плотность воздуха, зависящая от Z в соответствии с [6]; w — модуль скорости воздуха; C_L и C_Z — коэффициенты аэродинамического сопротивления и подъемной силы; сила тяги $P = (m_1' + m_a')w_e - m_a' w$, где m_1' — расход горючего, m_a' — расход воздуха, w_e — скорость потока, выходящего из двигателя.

В данной реализации методики возможно задание геометрии ЛА, составленной из характерного набора поверхностей — плоские, конические, цилиндрические, тороидальные. В частности, в данном ЛА нижняя часть носовой поверхности выбрана конической, а верхняя — тороидальной, образующиеся при их пересечении острые кромки

закруглены с радиусом 1 см. Воздухозаборник состоит из участков трех соосных конусов, расположенных друг за другом. Обечайка двигателя и верхняя часть корпуса над двигателем — участки цилиндрических поверхностей. Также имеются крылья, расположенные в районе двигателя.

Для гиперзвукового обтекания носовых частей ЛА как частей затупленного тела применялись полуэмпирические соотношения [7]. Локальный тепловой поток

$$q = \alpha(T_r - T_W);$$

$$T_r = T_\delta[1 + r\mathrm{M}_\delta^2(\gamma(T^*) - 1)/2],$$

здесь при ламинарном погранслое локальный коэффициент теплоотдачи

$$\alpha = \alpha_l = 0{,}325 C_p(T^*)\rho(T^*)w_\delta \mathrm{Re}_x^{-1/2}\,\mathrm{Pr}^{-2/3};$$

$$\mathrm{Re}_x = \rho_\delta(T^*)w_\delta x/\mu(T^*),$$

при турбулентном погранслое

$$\alpha = \alpha_t = 0{,}029 C_p(T^*)\rho(T^*)w_\delta \mathrm{Re}_x^{-1/5}\,\mathrm{Pr}^{-2/3},$$

T_r — температура восстановления; r — коэффициент восстановления температуры, $r = 0{,}9$; M_δ — число Маха в потоке воздуха на границе погранслоя; Re_x — число Рейнольдса; Pr — число Прандтля; w_δ — скорость воздуха на границе погранслоя; x — координата вдоль поверхности; $C_p(T^*)$ — теплоемкость; $\rho(T^*)$ — плотность; $\mu(T^*)$ — динамическая вязкость; γ — показатель адиабаты в потоке воздуха, вычисляемые при определяющей температуре $T^* = (T_\delta + T_W)/2 + 0{,}22(T_r - T_\delta)$; T_δ — температура воздуха на границе погранслоя; T_W — температура стенки. Для гиперзвукового течения в этих формулах можно считать, что для заостренного тела (или за пределами влияния затупления) T_δ, w_δ, M_δ связаны с параметрами невозмущенного потока T_∞, M_∞, w_∞ соотношениями на косой ударной волне под углом β к потоку

$$p_\delta/p_\infty = 2\gamma/(\gamma + 1)\,\mathrm{M}_\infty^2\sin^2\beta - (\gamma - 1)/(\gamma + 1);$$

$$w_\delta = w_\infty\cos\omega/\cos(\beta - \omega);$$

$$T_\delta/T_\infty = [2\gamma/(\gamma + 1)\,\mathrm{M}_\infty^2\sin^2\beta - (\gamma - 1)/(\gamma + 1)] \times$$

$$\times \{(\gamma - 1)/(\gamma + 1) + 2/[(\gamma + 1)\,\mathrm{M}_\infty^2\sin^2\beta]\};$$

$$\mathrm{tg}\omega =$$

$$= \mathrm{ctg}\beta(\mathrm{M}_\infty^2\sin^2\beta - 1)/\{1 + \mathrm{M}_\infty^2[(\gamma + 1)/2 - \sin^2\beta]\},$$

здесь ω — угол поворота потока. Для определения мощности аэродинамического нагрева вычислялось число Стентона $\mathrm{St}_h = \alpha_{\Delta h}/(\rho_\delta w_\delta)$; $\alpha_{\Delta h} = q/\Delta h$ — коэффициент теплоотдачи, определенный для разницы энтальпий; $\Delta h = h_W^* - h_W$; h_W^* — энтальпия среды у стенки при изоэнтропическом торможении потока; $h_W^* = C_p T_W + r\,w_\delta^2/2$, здесь C_p — теплоемкость при постоянном давлении; $h_W = C_p T_W$; $\mathrm{St}_h = \mathrm{Pr}^{-0{,}6}c_f/2$, где c_f — коэффициент трения (сила трения на единицу площади поверхности равна $c_f\rho_\delta w_\delta^2/2$). Для ламинарного потока локальное трение и теплоотдача в случае толщины погранслоя $\delta = 0$ при координате по потоку $L = 0$ [8].

$$c_f = 0{,}664/(\mathrm{Re}_L)^{1/2};\ \mathrm{Nu} = 0{,}332 f_l(\mathrm{Pr})(\mathrm{Re}_L)^{1/2};$$

$$f_l(\mathrm{Pr}) \approx 0{,}9,$$

здесь Re_L — число Рейнольдса, определенное при температуре T^*. Для турбулентного погранслоя [8]

$$c_f = B_1\mathrm{Re}_x^{-\mu};$$

$$\mu = m/(m + 1);$$

$$m = 2n/(n + 1);$$

$$B_1 = B^{m/(1+m)}(2/(1 + m))^{m/(1+m)};$$

$$B = 2((\delta^{**})^n/A)^{2/(1+n)};\ \delta^{**} = n/[(1 + n)(1 + 2n)],$$
$$n = 1/7,\ A = 8{,}74.$$

Ламинарно-турбулентный переход происходит при [9]

$$\log\mathrm{Re}_{x\delta} = \log\mathrm{Re}_{cr} + C_\mathrm{M}\mathrm{M}_\delta,$$

здесь $\mathrm{Re}_{x\delta}$ — локальное число Рейнольдса в пристеночном потоке; M_δ — локальное число Маха в пристеночном потоке; Re_{cr}, C_M — эмпирические константы, на фюзеляже и крыльях без скольжения $C_\mathrm{M} = 0{,}2$, $\log\mathrm{Re}_{cr} = 5{,}5$.

Радиационный тепловой поток с поверхности

$$q_{rad} = \chi\sigma_{SB}T_W^4.$$

Здесь $\chi \approx 1$ — степень черноты; σ_{SB} — постоянная Стефана—Больцмана.

При принятой конструкции ЛА воздух испытывает в воздухозаборнике сжатие и нагрев в четырех последовательных косых ударных волнах: в трех при взаимодействии с трехступенчатым профилем нижней части ЛА и в одной — при взаимодействии

с нижней стенкой двигателя. Характеристики потока за ударными волнами находятся из уравнений [10]:

$$\mathrm{tg}(v_i - \omega_i) = \{(\gamma - 1)/(\gamma + 1) + 2/[(\gamma + 1)\mathrm{M}_i^2 \sin^2(v_i)]\}\mathrm{tg}(v_i),\ i = 0...3;$$

$$p_{i+1}/p_i = [2\gamma \mathrm{M}_i^2 \sin^2(v_i) - (\gamma - 1)]/(\gamma + 1);$$

$$\rho_{i+1}/\rho_i = [\mathrm{M}_i^2 \sin(v_i)(\gamma + 1)]/[2 + (\gamma - 1)\mathrm{M}_i^2 \sin^2(v_i)];$$

$$w_{i+1}^2/w_i^2 = 1 - \sin^2(v_i) + \sin^2(v_i)\{2/[(\gamma + 1) \times \times | \mathrm{M}_i^2 \sin^2(v_i)] + (\gamma - 1)/(\gamma + 1)\}^2;$$

$$T_{i+1}/T_i = [2\gamma \mathrm{M}_i^2 \sin^2(v_i) - (\gamma - 1)]\{2/[\mathrm{M}_i^2 \sin^2(v_i)] + (\gamma - 1)\}/(\gamma + 1)^2,$$

здесь ω_i — угол наклона стенки к i-му потоку при i-м повороте; v_i — угол наклона ударной волны к i-му потоку, индекс "0" соответствует невозмущенному потоку; p, ρ, w, T — давление, плотность, скорость и температура воздуха. Расчеты тепловых потоков проводились по приведенным выше соотношениям для турбулентного погранслоя.

Рассчитывался нестационарный нагрев внутренних структур аппарата, например, с "внешним слоем" оборудования и приборов b_1, на который падает излучение со стенки W_1, и "внутренними частями" d, заэкранированными этим слоем b_1 (рис. 1). Считалось, что стенка W_1 с теплоемкостью C_{pW_1}, плотностью ρ_{W_1} и толщиной δ_{W_1} прогревается аэродинамическим потоком q_{Σ_1} до температуры T_{W_1},

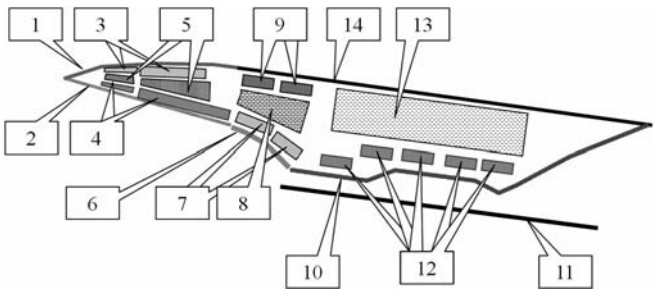


Рис. 1. Структура ЛА (к расчету тепловых процессов при полете): *1, 2* — верхняя и нижняя стенки передней части ЛА; *3, 4* — узлы, примыкающие к верхней и нижней стенкам передней части ЛА; *5* — узлы в глубине передней части ЛА; *6* — стенка воздухозаборника; *7* — узлы, примыкающие к стенке воздухозаборника; *8* — узлы в глубине над воздухозаборником; *9* — узлы, примыкающие к верхней стенке основной части фюзеляжа ЛА; *10, 11* — верхняя и нижняя стенки двигателя; *12* — узлы, примыкающие к верхней стенке двигателя; *13* — бак с основным горючим; *14* — верхняя стенка основной части фюзеляжа ЛА

охлаждается в обе стороны излучением с мощностью $(1 + \eta_{Wb_1})\sigma_{SB} T_{W_1}^4$, где $\eta_{Wb_1} = 0...1$ учитывает изоляцию. Учтены кондуктивные потоки теплоты на характерное расстояние Δ с характерной теплопроводностью $\lambda = \Delta\Sigma(F_i\lambda_i/\Delta_i)/\Sigma(F_i\lambda_i)$; F_i, λ_i, Δ_i — соответствующие площади поперечного сечения, коэффициенты теплопроводности и расстояния передачи теплового потока по тепловым мостикам (креплениям, изоляции, трубам и т. п.). Соответствующие соотношения, описывающие средние тепловые потоки от двух противоположных стенок W_1 и W_2, два "внешних слоя" узлов b_1 и b_2 и "внутренние части" d имеют вид:

$$\delta_{W_1} C_{pW_1} \rho_{W_1} \partial T_{W_1}/\partial t = q_{\Sigma_1} - q_{m'1} - (1 + \eta_{Wb_1})\sigma_{SB} T_{W_1}^4 + \eta_{Wb_1}\sigma_{SB} T_{b_1}^4 - \lambda_{Wb_1}(T_{W_1} - T_{b_1})/\Delta_{Wb_1};$$

$$\delta_{b_1} C_{pb_1} \rho_{b_1} \partial T_{b_1}/\partial t = \lambda_{Wb_1}(T_{W_1} - T_{b_1})/\Delta_{Wb_1} + \eta_{Wb_1}\sigma_{SB} T_{W_1}^4 - (1 + \eta_{Wb_1})\sigma_{SB} T_{b_1}^4 + \sigma_{SB} T_d^4 - \lambda_{db_1}(T_{b_1} - T_d)/\Delta_{db_1};$$

$$\delta_d C_{pd} \rho_d \partial T_d/\partial t = \lambda_{db_1}(T_{b_1} - T_d)/\Delta_{db_1} + \lambda_{db_2}(T_{b_2} - T_d)/\Delta_{db_2} + \sigma_{SB} T_{b_1}^4 - 2\sigma_{SB} T_d^4 + \sigma_{SB} T_{b_2}^4;$$

$$\delta_{b_2} C_{pb_2} \rho_{b_2} \partial T_{b_2}/\partial t = \lambda_{Wb_2}(T_{W_2} - T_{b_2})/\Delta_{Wb_2} + \eta_{Wb_2}\sigma_{SB} T_{W_2}^4 - (1 + \eta_{Wb_2})\sigma_{SB} T_{b_2}^4 + \sigma_{SB} T_d^4 - \lambda_{db_2}(T_{b_2} - T_d)/\Delta_{db_2};$$

$$\delta_{W_2} C_{pW_2} \rho_{W_2} \partial T_{W_2}/\partial t = q_{\Sigma_2} - q_{m'2} - (1 + \eta_{Wb_2})\sigma_{SB} T_{W_2}^4 + \eta_{Wb_2}\sigma_{SB} T_{b_2}^4 - \lambda_{Wb_2}(T_{W_2} - T_{b_2})/\Delta_{Wb_2}.$$

Предусмотрена возможность активного охлаждения стенок двигателя. Считается, что горючее поступает в стенку, где оно нагревается и испаряется при температуре T_W с удельным поглощением энергии $\Delta\varepsilon_W$, пары нагреваются и могут проходить химические превращения при температуре T_{ch} с удельным поглощением энергии $\Delta\varepsilon_{ch}$. Тепловой поток со стенки $q_{m'1} = \varepsilon_f(T_{W_1})\, m'_{f_1}/F_1$, здесь $\varepsilon_f(T_{W_1})$ — удельная энергия топлива при температуре T_{W_1}; m'_{f_1} — его расход, при расчетах он определяется как часть η_{W_1} нестационарного расхода, обеспечивающего стехиометрический состав в камере

сгорания; F_1 — площадь стенки W_1. Например, для гептана разложение на цеолите завершается (превышает 90 %) при температуре около $T^* = 950...1000$ K, при этом на нагрев и испарение тратится $\Delta\varepsilon_{ph} = 2{,}32 \cdot 10^6$ Дж/кг, а на химическую реакцию — $\Delta\varepsilon_{ch} = 3{,}60 \cdot 10^6$ Дж/кг (желательно при этом поддерживать давление выше $p^* \approx 2{,}2$ МПа).

При описании процессов в двигателе применялось канальное приближение с учетом малости поперечных градиентов параметров ядра течения и с погранслоями, соответствующими начальному и основному участкам течения в кольцевом канале с переменными диаметрами, с поступлением массы топлива и энерговыделением, распределенным в заданном объеме камеры сгорания. Приток массы и энергии определялся по расходу горючего; в рассматриваемых вариантах он был пропорционален расходу воздуха через двигатель, чтобы был постоянен коэффициент избытка воздуха; возможны и другие закономерности инжекции горючего, в том числе ручное управление. Коэффициенты теплоотдачи α_d и α_u для нижней и верхних стенок двигателя, имеющего форму сектора кольцевого зазора, определяется выражениями [8]

$$\alpha_d = (1 - 0{,}45/(2{,}4 + \mathrm{Pr})(D_d/D_u)^{0,6})f_{Td}f_E\lambda\mathrm{Nu}_e/D;$$

$$\alpha_u = (1 - 0{,}45/(2{,}4 + \mathrm{Pr})(D_d/D_u)^{0,16/\mathrm{Pr}''})f_{Td}f_E\lambda\mathrm{Nu}_e/D.$$

Здесь $\mathrm{Pr}'' = \mathrm{Pr}^{0,15}$;

$$\mathrm{Nu}_e = \zeta\mathrm{Re}_e\mathrm{Pr}/(40\zeta^{1/2}(\mathrm{Pr}^{2/3} - 1) + 8);$$

$$\zeta = 1/(0{,}78\log\mathrm{Re}_e - 1{,}64);$$

$$f_{Td} = 4/((\psi_d - 1{,}64(\psi_d - 1)\zeta^{1/2})^{1/2} + 1);$$

$$f_{Tu} = 4/((\psi_u - 1{,}64(\psi_u - 1)\zeta^{1/2})^{1/2} + 1);$$

$$f_E = (0{,}6\,\mathrm{Re}_e^{0,25}\,D/l)^{0,2},$$

где λ — коэффициент теплопроводности газа (по формуле Эйкена [11]); Nu_e — число Нуссельта; D — средний диаметр двигателя; f_{Td} и f_{Tu} — температурные факторы для нижней и верхней стенки; f_E — множитель, описывающий теплообмен на входном участке; D_d и D_u — диаметр нижней и верхней стенок двигателя; Re_e — число Рейнольдса для потока в двигателе; ψ_d — отношение между температурой нижней стенки и температурой в двигателе; ψ_u — то же для верхней стенки; l — расстояние по оси OZ от исследуемой точки до входа в двигатель.

В разработанном программном комплексе возможно задание различных управляющих алгоритмов или ручное управление углом атаки при полете. В представленных расчетах применялся закон управления, при котором полет происходит с постоянным углом атаки (6°). При различных высотах, но при примерном постоянстве скорости полета это соответствует сохранению геометрии ударно-волновой структуры перед двигателем, т. е. безо всяких дополнительных средств управления структурой ее легко сделать оптимальной для работы прямоточного двигателя, что позволяет обеспечить высокую тяговую эффективность без усложнения конструкции.

3. Характерные результаты вычислений

Некоторые характерные результаты вычислений на примере выполнения задачи доставки груза фиксированной массы (300 кг) на максимальное расстояние представлены на рис. 2—6 для следующего набора параметров: масса конструкции ЛА 506 кг, масса топлива при запуске 1288 кг, длина ЛА 6,1 м, диаметр его цилиндрической части 0,9 м, коэффициент избытка воздуха 1,5, аэродинамические коэффициенты для продольной и поперечной составляющих аэродинамической силы примерно постоянны и равны соответственно 0,13 и 0,44, горючее — гептан. Считалось, что $\delta_W = 0{,}5$ см, $\rho_W C_{pW} = 5{,}5 \cdot 10^6$ Дж/(м$^3 \cdot$ K), $\rho_b C_{pb} = \rho_d C_{pd} = \rho_W C_{pW}\xi_з$, $\xi_з = 0{,}5$, в носовой части и области воздухозаборника $\delta_d = 3$ см, в носовой части $\delta_b = 10$ см, в области воздухозаборника $\delta_b = 20$ см,

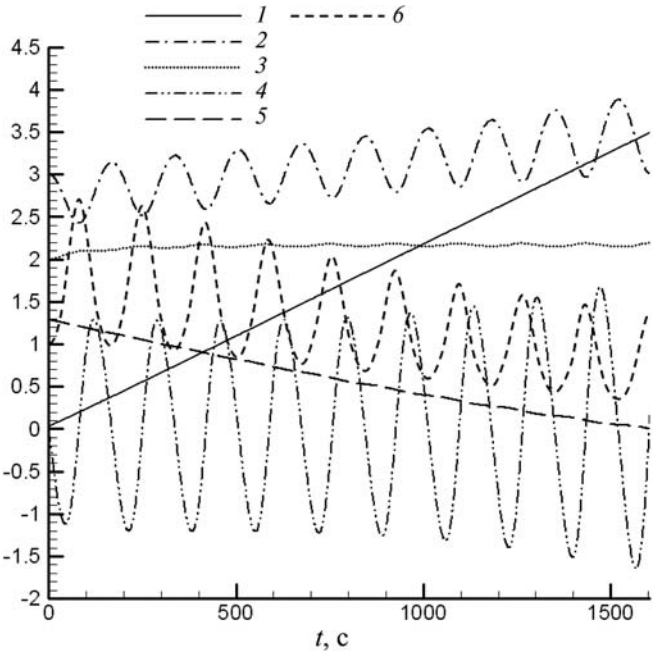


Рис. 2. Временные зависимости параметров полета:
1 — удаление L от точки старта прямоточного воздушно-реактивного двигателя, в 10^6 м; *2* — высота полета Z, в 10^4 м; *3*, *4* — горизонтальная и вертикальная составляющие скорости, в 10^3 м/с; *5* — масса топлива, в 10^3 кг; *6* — максимальное давление в двигателе, в 10^6 Па

над двигателем $\delta_d = 10$ см, теплоемкость топлива $2{,}5 \cdot 10^3$ Дж/(кг · К), плотность топлива $0{,}8 \cdot 10^3$ кг/м^3, $\lambda_{db}/\Delta_{db} = 0{,}052$ Вт/(м^2 · К), $\varepsilon_f(T_{W_1}) = \Delta\varepsilon_{ph}\, T_{W_1}/T^* + \Delta\varepsilon_{ch}/\{1 + \exp[(T_{ch} - T_{W_1})/\Delta T_{ch}]\}$, где $T_{ch} \approx 900$ К, $\Delta T_{ch} \approx 50$ К. Двигатель ЛА включался на высоте 30 км при скорости 2 км/с. В данном варианте ЛА осуществляется охлаждение двигателя с помощью вышеописанной эндотермической реакции разложения горючего. В охлаждении двигателя участвует 90 % горючего, остальное горючее используется в охлаждении окошек, критических точек и пр. Для снижения теплового потока, приходящего к баку с горючим от двигателя, между баком и узлами над двигателем расположен слой фольги с коэффициентом отражения 97 %.

Видно, что полет происходит по периодической траектории. Имеются колебания высоты и других кинематических характеристик. Дальность полета до момента, когда заканчивается топливо, составляет приблизительно 3,5 тыс. км. Горизонтальная составляющая скорости в начале полета незначительно возрастает, а затем практически не меняется. Давление в двигателе достигает 2 МПа, что предъявляет жесткие требования к жаропрочности конструкции.

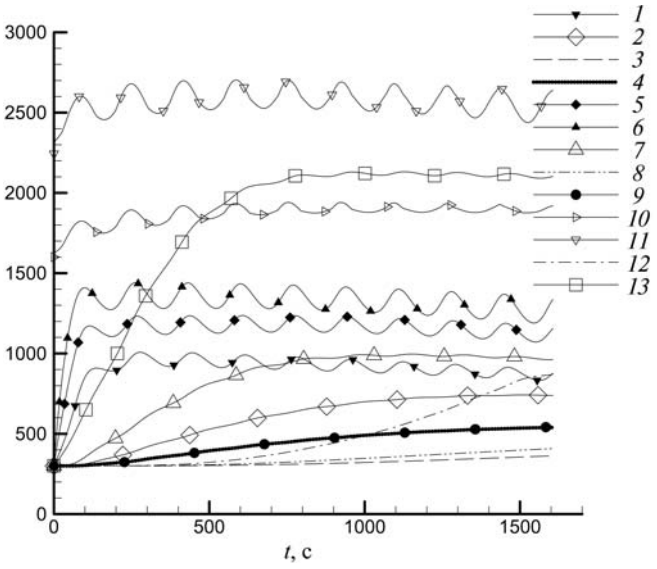


Рис. 6. Временные зависимости изменения температуры узлов ЛА (в К):
1 — средняя температура нижней стенки передней части ЛА; *2* — средняя температура узлов, примыкающих к нижней стенке передней части ЛА; *3* — средняя температура узлов, находящихся в центре корпуса в передней части; *4* — средняя температура узлов, примыкающих к верхней стенке передней части ЛА; *5* — средняя температура воздухозаборника; *6* — максимальная температура воздухозаборника; *7* — средняя температура узлов, примыкающих к воздухозаборнику; *8* — средняя температура узлов, находящихся в центре корпуса в области воздухозаборника; *9* — средняя температура узлов, примыкающих к верхней стенке в области воздухозаборника; *10* — средняя температура в двигателе; *11* — максимальная температура в двигателе; *12* — средняя температура бака; *13* — средняя температура приборов над баком

Распределения температуры, давления и силы трения по поверхностям ЛА приведены на рис. 3—5 (см. вторую сторону обложки) для момента времени 1096 с от начала работы прямоточного двигателя ЛА.

На рис. 6 представлены временные зависимости температур ряда узлов ЛА. Максимальная температура — в критическом сечении двигателя, в данном случае она достигает 2700 К. Средняя температура стенок значительно ниже — не доходит до 2000 К. Стенка воздухозаборника и нижняя стенка передней части ЛА нагреваются до 1000...1400 К. Температуры ниже 500 К сохраняются в течение всего полета только у узлов, расположенных в глубине передней части и в глубине над воздухозаборниками, а также у верхней стенки фюзеляжа и в узлах, примыкающих к ней. Примерно в середине полета бак с топливом нагревается до 600 К, что может привести к кипению горючего и/или повышению давления. На основе вышесказанного можно заключить, что при проектировании узлов ЛА, предназначенных для выполнения подобных полетов, надо или учитывать повышенные температуры утих узлов, или изолировать и/или охлаждать их, или располагать их в глубине аппарата.

Таким образом, представленная комплексная модель описывает различные аспекты функционирования узлов высокоскоростного летательного аппарата с учетом их нелинейного взаимного влияния.

Список литературы

1. **Tran K.** One Dimensional Analysis Program for Scramjet and Ramjet Flowpaths. Blacksburg: Virginia Polytechnic Institute and State University. 2010. 111 p.

2. **Doolan C. A.** Quasi-One-Dimensional Mixing and Combustion Code for Trajectory Optimization and Design Studies // 15th AIAA International Space Planes and Hypersonic Systems and Technologies Conference. Dayton: AIAA. 1995. 25 p.

3. **Weissman D.** Representation of two-dimensional hypersonic inlet flows for one-dimensional scramjet cycle analysis. 28th Aerospace Sciences Meeting. Reno, NV, AIAA, 1990. 50 p.

4. **Jolley P. R., Whitmore S. A.** Aerodynamic and Propulsion Assisted Maneuvering for Orbital Transfer Vehicles // 5th Responsive Space Conference. Los Angeles: AIAA. 2007. 40 p.

5. **Фомин В. М., Аульченко С. М., Звегинцев В. И.** Полет гиперзвукового летательного аппарата с прямоточным воздушно-реактивным двигателем по рикошетирующей траектории // Прикладная механика и техническая физика. 2010. Т. 4 (51). С. 85—94.

6. **ГОСТ Р 4401—95.** Стандартная атмосфера. 1995.

7. **Аржаников Н. С., Садекова Г. С.** Аэродинамика больших скоростей. М.: Высшая школа. 1965. 562 с.

8. **Кутателадзе С. С.** Теплопередача и гидродинамическое сопротивление. М.: Энергоатомиздат. 1990. 367 с.

9. **Ouinn R. D., Gong L.** A Method for Calculating Transient Surface Temperatures and Surface Heating Rates for High-Speed Aircraft. Edwards, California: Dryden Flight Research Center (NASA/TP-2000-209034). 2000. 35 p.

10. **Абрамович Г. Н.** Прикладная газовая динамика. Ч. 1. М.: Наука. 1991. 602 с.

11. **Григорьев И. С., Мейлихов Е. З.** Физические величины: Справочник. М.: Энергоатомиздат. 1991. 1232 с.

УДК 004.942

Ю. О. Викторов, аспирант, инженер-практикант,
e-mail: yuriy.viktorov@intel.com,
А. Н. Готманов, инженер-исследователь,
e-mail: alexander.gotmanov@intel.com
Intel Corporation

Проблемы качества обслуживания при проектировании коммуникационных фабрик в системах на кристалле

Рассматривается проектирование систем на кристалле, удовлетворяющих заданным требованиям качества обслуживания. Описываются проблемы, возникающие при анализе показателей производительности. Для анализа качества обслуживания предлагается использовать высокоуровневое моделирование в среде xMAS с последующей формальной верификацией.

Ключевые слова: *качество обслуживания, системы на кристалле, сети на кристалле, коммуникационные фабрики, сеть межсоединений*

Введение

Традиционно устройства внутри микропроцессора коммутировались с помощью общей шины. С течением времени росла степень интеграции, что сопровождалось увеличением числа блоков внутри микропроцессора [1]. При числе блоков порядка десятков и сотен шинная архитектура соединений становится непригодной вследствие ограниченной пропускной способности и масштабируемости [2].

Современные системы межсоединений, архитектуры которых в сильной степени зависят от области применения [3], объединяют под общим названием *коммуникационных фабрик* (*communication fabrics*). При этом для северного кластера микросхем потребительской электроники характерна централизованная структура, являющаяся дальнейшим развитием идей общей шины в сторону конвейеризации и параллельной обработки запросов. Южный кластер традиционно имеет древовидную иерархическую структуру [4]. Для серверов применяют распределенные архитектуры типа "сеть на кристалле" (*Network on Chip*, *NoC*), такие как кольца и решетки [5].

Ресурсы в системах на кристалле (размеры очередей, разрядности шин и т. д.), как правило, ограничены, а производительность и корректность работы системы сильно зависят от эффективности их разделения конкурирующими процессами. Поэтому при проектировании коммуникационных фабрик необходимо учитывать минимальный уровень обслуживания, который система должна гарантировать своим агентам. Совокупность таких гарантий называется *качеством обслуживания*. В литературе существуют различные формулировки этого понятия, привязанные к разным предметным областям [6, 7]. В наиболее общем виде качество обслуживания (*Quality of Service*, *QoS*) — это способность системы обеспечивать своим агентам определенные измеряемые гарантии по обработке потоков данных, касающиеся производительности, надежности, безопасности и т. д. *Потоком данных* мы будем называть совокупность сообщений, обладающих каким-либо общим признаком, например, сообщения от одного источника.

Существует большое разнообразие метрик для описания уровня качества обслуживания. Некоторые метрики качества обслуживания тесно связаны друг с другом или могут быть выражены друг через друга. Например, нулевая вероятность потери данных эквивалентна требованию гарантии доставки [7]. Для каждой конкретной цели (с учетом свойств предметной области), задач и предполагаемых методов их решения выбирается определенный базис метрик.

В данной статье мы уделяем основное внимание гарантиям производительности. Можно выделить ряд показателей производительности, которые представляют интерес независимо от используемой архитектуры [8]:

- форма потока — объем данных, переданных потоком, начиная с некоторого момента времени t_0. Для описания формы потока часто используют показатели средней плотности (*rate*, *bandwidth*) и величины всплеска (*burst*) [9, 10];
- задержка передачи (*latency*) — интервал времени между отправлением данных источником и их получением в точке назначения;
- вариация задержки (*jitter*) — разница между максимальным и минимальным значениями задержки в потоке данных.

В этой статье мы рассмотрим проблемы, связанные с проектированием и анализом гарантий производительности в системах на кристалле. Начав с простого примера в разделе 1, мы проведем обзор результатов в области качества обслуживания (раздел 2). Раздел 3 посвящен трудностям проектирования коммуникационных фабрик, а раздел 4 — применяемым на практике методам анализа их свойств. В разделе 5 мы предлагаем собственный подход к проверке гарантий производительности в системах на кристалле с использованием среды высокоуровневого моделирования xMAS и средств верификации моделей. В заключении дан план дальнейшей работы.

1. Гарантии производительности на примере

Поясним важность выполнения требований качества обслуживания на примере системы, состоящей из видеоконтроллера и оперативной памяти, соединенных через коммуникационную фабрику (рис. 1) [11].

Видеоконтроллер выполняет построчное отображение изображения. Соединение с монитором требует постоянной скорости передачи данных, изменять которую видеоконтроллер не должен. Видеоконтроллер имеет внутренний буфер небольшого объема (существенно меньшего, чем размер одного кадра изображения). В случае опустошения буфера при отрисовке кадра часть пикселей изображения будет иметь случайный цвет. Поэтому при корректной работе системы буфер никогда не должен опустошаться.

Запросы видеоконтроллера на чтение из памяти попадают в коммуникационную фабрику. На обработку запроса требуется некоторое время, после чего данные поступают в буфер видеоконтроллера. Заполнение буфера зависит от пропускной способности, задержки и ее вариации для потока запросов и ответов, т. е. от уровня качества обслуживания. Например, можно потребовать, чтобы ответ на очередной запрос поступал не позднее, чем через время T после его отправки. Правильный подбор емкости буфера и T позволяет гарантировать корректность работы.

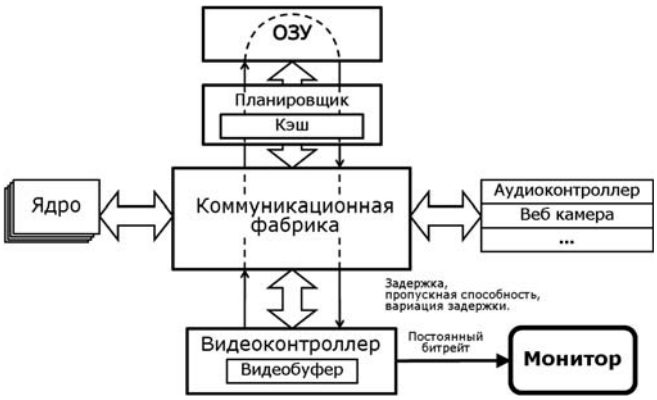


Рис. 1. Функционирование видеоконтроллера в составе системы на кристалле

Ситуация осложняется тем, что на практике взаимодействие с памятью осуществляется через планировщик, который может пакетировать запросы или задерживать их выполнение (например, для сокращения энергопотребления), а также содержит кэш, время обращения к которому на несколько порядков меньше времени обращения к памяти. Также запросы видеоконтроллера конкурируют с запросами других агентов, таких как процессорные ядра, аудиоконтроллер и т. п. [12].

2. Обзор результатов

Сети на кристалле (*NoC*) являются прямым аналогом вычислительных сетей, которые исторически возникли намного раньше. Вопросы их проектирования, в том числе и в области качества обслуживания, подробно исследованы в литературе и отражены в технических стандартах [13—15]. Существуют методы анализа и обеспечения требований к пропускной способности, задержке и ее вариации, надежности и т. д. [9, 16, 17]. Широко используются различные схемы резервирования для достижения компромисса между качеством обслуживания и утилизацией ресурсов [15].

В глобальных сетях поток данных может проходить через участки, построенные по различным технологиям, различающимся способами коммутации, разделения среды, маршрутизации и т. д. [17]. Например, при звонке с IP-телефона на сотовый телефон голосовой трафик может сначала проходить через локальную вычислительную сеть с коммутацией пакетов, затем через сеть ATM с коммутацией соединений и, наконец, через мобильную сеть UMTS [15]. Чтобы обеспечить выполнение требований качества обслуживания на всем пути от источника к получателю, необходимо учитывать отличия механизмов качества обслуживания в этих сетях.

Существенным отличием вычислительных сетей от сетей на кристалле является масштаб времени [18]. Для глобальных сетей время обработки пакетов измеряется микросекундами или миллисекундами, тогда как для сетей на кристалле время обработки и доставки пакетов измеряется наносекундами при более высокой пропускной способности [3]. Это накладывает дополнительные ограничения на используемые алгоритмы маршрутизации и арбитража. Для вычислительной сети анализ одного пакета данных может предполагать выполнение целой подпрограммы, в то время как в сетях на кристалле боль-

шинство решений должно приниматься за 1—2 периода тактового сигнала или комбинационно. Для буферизации трафика в сетях на кристалле используется регистровая память небольшого объема, и широко применяются сложные схемы разделения ресурсов с блокировкой каналов. Таким образом, решение вопросов качества обслуживания в коммуникационных фабриках требует как минимум существенных усилий по адаптации существующих подходов, а на практике — разработки новых [19, 20—22].

За последнее десятилетие были предложены подходы к проектированию коммуникационных фабрик, призванные обеспечить гарантии качества обслуживания "по построению". Их можно условно разделить на две категории [23]. Подходы на основе дифференциации трафика (*traffic differentiation*) решают проблему конкурирующих запросов введением уровней приоритета, соответствующих разным классам трафика [24]. Однако для сложной системы затруднительно определить необходимое число уровней приоритетов и избежать проблем инверсии приоритета и истощения ресурсов (*starvation*).

Передача без конкуренции (*contention-free transmisson*) основывается на той или иной схеме резервирования ресурсов перед началом передачи [25]. Используя статическое резервирование, несложно добиться определенных гарантий производительности. Такой подход позволяет оптимизировать архитектуру под конкретное приложение и пригоден для встраиваемых систем, но приводит к низкой утилизации ресурсов в системах на кристалле широкого спектра применения [26]. При динамическом резервировании ресурсов задержки на стадии резервирования по-прежнему трудно оценить.

3. Проблемы проектирования коммуникационных фабрик

Коммуникационная фабрика — это сложное распределенное устройство, решающее задачу взаимодействия агентов на кристалле и обеспечивающее эффективное разделение ресурсов. Для него характерен высокий уровень параллелизма и конвейеризации, с буферизацией данных и одновременной обработкой множества транзакций, проводимых разными устройствами и находящихся на разных стадиях исполнения [27].

Некорректное распределенное взаимодействие агентов и фабрики может приводить к нарушению целостности памяти, возникновению тупиковых ситуаций, истощению ресурсов и другим проблемам, которые не могут быть обнаружены при анализе частей системы по отдельности. Это затрудняет обнаружение и локализацию ошибок [28] и усугубляется тем, что при проектировании сложной системы (например, микропроцессора) большинство инженеров не имеют полного представления о работе системы в целом.

На практике требования качества обслуживания далеко не всегда достаточно формализованы. Проектные спецификации часто составляют, не используя ясных метрик, что затрудняет их анализ [7]. Принят подход описания требований качества обслуживания на основе примеров, когда вместо формального описания требований в спецификации приводится ряд типовых сценариев работы системы и желаемое поведение. Формализация такого описания требует значительных усилий по его уточнению.

Проектирование коммуникационных фабрик осложняется большим числом второстепенных факторов. Например, обработка конкурирующих запросов и динамическое управление энергопотреблением может приводить к неравномерной утилизации ресурсов и большому разбросу значений задержки [29]. Грубая оценка по наихудшему случаю часто оказывается чрезмерно консервативной и недостижимой на практике, а получение точных оценок связано с большой вычислительной сложностью. Существующие методы и средства САПР не обеспечивают своевременного обнаружения ошибок. Проблемы проявляются на поздних стадиях и не всегда могут быть эффективно исправлены, что приводит к удорожанию проекта и срыву сроков выхода продукта на рынок.

4. Методы анализа производительности

Для анализа производительности в системах на кристалле используют имитационное моделирование и формальные методы.

Имитационное моделирование можно применять к моделям различных уровней сложности и детализации. Симуляция подробного RTL-описания коммуникационной фабрики и ее окружения дает наиболее точные результаты, но возможна только на поздних стадиях разработки и оказывается чрезмерно сложной [26]. Современные RTL-симуляторы позволяют исследовать поведение такой модели лишь на ограниченном отрезке времени. При этом сложность проектируемых систем растет быстрее, чем производительность средств имитационного моделирования [30]. На практике для анализа производительности используют модели высокого уровня (обычно реализованные на C++ или SystemC) [31].

Принципиальным ограничением имитационного моделирования является возможность исследовать лишь малый набор поведений, что позволяет выявить только статистически значимые ошибки и не дает достаточной уверенности в корректной ра-

боте системы. Например, тупиковые состояния и проблемы истощения ресурсов часто возникают в нетипичных для данной системы условиях, поэтому их можно обнаружить только для систем крайне малого размера путем полного перебора поведений.

Для исчерпывающего анализа качества обслуживания используют формальные подходы, такие как аналитические методы, формальная верификация (верификация моделей) и доказательство теорем. *Аналитические методы* работают с сильно обобщенной моделью системы и позволяют получать приближенные оценки метрик качества обслуживания в виде замкнутых формул [32]. Например, в сетевом исчислении (*network calculus*) используются линейные модели трафика и производительности узлов, при этом показатели качества обслуживания, такие как задержка и заполнение буфера (*backlog*), могут быть выражены простыми зависимостями от параметров сети [10]. Однако этот подход применим только для сетей определенного вида. Например, обычно полагают, что все очереди в сети имеют достаточный объем, исключающий потери данных и возникновение блокировки каналов (*backpressure*). В коммуникационных фабриках такие предположения часто не выполняются [9].

Формальная верификация работает с абстрактной математической моделью системы, семантически близкой к недетерминированному конечному автомату. Примером такой модели может служить синхронное описание на языке SMV [33] или асинхронная программа на языке Murphi [34]. Описание модели дополняется спецификацией ее свойств на языке темпоральной логики [35]. Простейшим примером такого свойства является инвариант, т. е. утверждение, ограничивающие множество допустимых состояний системы. Для каждого типа моделей существуют алгоритмы проверки спецификаций, основанные на переборе состояний и поведений системы в той или иной форме. Заметим, что быструю сходимость алгоритмов можно гарантировать далеко не всегда, так как она зависит от характеристик модели, сложности спецификации и способа организации перебора. Однако современные достижения в области задач SAT и SMT [36] существенно расширили применимость методов формальной верификации.

Метод доказательства теорем требует описания поведения системы в некоторой формальной логике [37] или в виде особого вида программы [38]. Гарантии качества обслуживания могут быть сформулированы в виде основных теорем, доказательство которых проходит в полуавтоматическом режиме и обычно требует большого числа вспомогательных утверждений (лемм), вводимых вручную. Высокая трудоемкость часто делает доказательство теорем нерентабельным, особенно на ранних этапах проектирования.

Для некоторых видов сетей могут применяться специализированные языки моделирования и средства анализа. Например, для оценки производительности эластичных систем [39] используют маркированные графы; сети на кристалле "Æthereal [40] моделируют с помощью диаграмм потоков данных. Однако такие методы неприменимы для коммуникационных фабрик общего вида.

5. Верификация качества обслуживания на модели

Среда моделирования xMAS (*eXecutable Microarchitecture Specification*) была специально разработана для создания математических моделей микроархитектур, описывающих в абстрактной форме размеры и расположение буферов, точки арбитража, задержки вычислений и другие свойства системы, влияющие на уровень качества обслуживания [41]. Модели xMAS конструируются из небольшого числа параметризованных стандартных блоков (примитивов) (рис. 2), соединенных каналами в синхронную сеть передачи пакетов. Примитивы выполняют простые операции над пакетами данных. Например, queue (очередь) выполняет буферизацию пакетов, сохраняя порядок их следования, а примитивы join и fork играют роль барьеров, синхронизируя передачи на входных и выходных каналах.

Для верификации моделей xMAS используются как специализированные алгоритмы, так и средства верификации моделей общего назначения. Модель xMAS всегда может быть автоматически представлена в виде эквивалентного описания на языке Verilog, к которому применимы методы символьной верификации, такие как ограниченная вери-

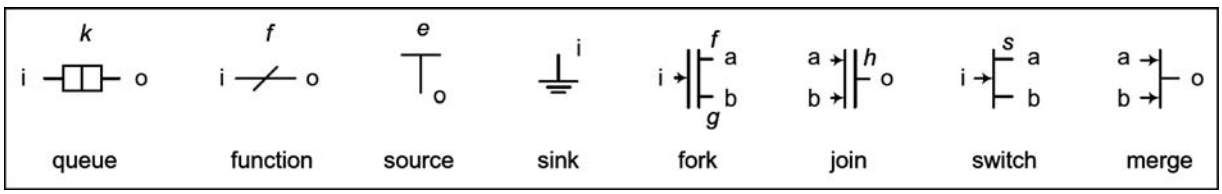


Рис. 2. Символы основных примитивов для моделирования в среде xMAS. Символы, выделенные курсивом (*k*, *f*, *e*, *g*, *h* и *s*), обозначают параметры, символы (i, o, a, и b) указывают имена соединений

фикация (*bounded model checking*) [42, 43], интерполяция [44], *k*-индукция [36] и т. д. Во многих случаях масштабируемость средств общего назначения можно существенно повысить, используя структурный анализ модели для автоматического порождения вспомогательных лемм [45].

Модели xMAS позволяют описывать микроархитектуру коммуникационных фабрик в простой и наглядной форме. Они успешно применялись для обнаружения тупиковых ситуаций (*deadlocks*) и формального доказательства их отсутствия [46]. Такой анализ может быть проведен без учета особенностей алгоритмов арбитража и интенсивности потоков данных. По этой причине базовый примитив арбитра (*merge*) реализует простейшую справедливую политику выбора (например, round-robin [47]), а моменты порождения и поглощения запросов выбираются примитивами source и sink случайно.

Моделирование качества обслуживания требует учета большего числа деталей и расширения набора примитивов xMAS. Мы разработали несколько специфических реализаций примитива merge для таких политик арбитража, как polling, age-based [47], а также некоторых политик, используемых в промышленных системах на кристалле. Для порождения трафика с заданной средней плотностью и величиной всплеска к набору примитивов xMAS был добавлен примитив shaper (формирователь потока). Shaper блокирует входной канал, когда интенсивность входящих пакетов превышает пороговое значение.

Простейшей задачей верификации качества обслуживания является анализ свойств арбитров. Мы построили ряд моделей, содержащих арбитр и очереди входящих и исходящих запросов, с ограничениями на интенсивность порождения и обработки пакетов. В качестве показателя производительности было выбрано максимальное время ожидания запроса на входе арбитра. Ограничение времени ожидания выражается предложением темпоральной логики, которое может быть заменено эквивалентным ему инвариантом при добавлении к модели специального счетчика задержки. Для получения начальных оценок использовались имитационное моделирование и ограниченная верификация. Затем для ряда случаев удалось получить индуктивные доказательства корректности найденных оценок.

Основной проблемой применения стандартных средств верификации моделей к задачам качества обслуживания является низкая масштабируемость. Например, для получения консервативной оценки задержки L в наихудшем случае можно воспользоваться k-индукцией. Однако сходимость метода можно гарантировать только при числе шагов k, близком к оценке L. Значение L может достигать сотен и тысяч циклов работы системы, что находится далеко за пределами возможностей существующих средств формальной верификации логических сетей. Мы планируем разработать методы верификации задержки в моделях xMAS, производительность которых будет слабо зависеть от L.

Заключение

Проблема анализа качества обслуживания для коммуникационных фабрик и сетей на кристалле представляет большой практический интерес и относительно мало изучена. В силу современных тенденций развития индустрии (увеличение числа транзисторов на единицу площади, многоядерность, интеграция все большего числа узлов системы в процессорный кристалл) важность этой задачи в дальнейшем будет только возрастать. В настоящее время ведутся активные исследования архитектуры промышленных коммуникационных фабрик и разрабатываются алгоритмы надежной оценки параметров качества обслуживания.

Мы планируем получить практические методы верификации качества обслуживания на основе абстрактных микроархитектурных моделей xMAS с применением формальных или полуформальных алгоритмов. На начальном этапе будет проведено сравнение различных механизмов качества обслуживания и подходов, применяемых для анализа их свойств. Затем мы разработаем специализированные алгоритмы оценки таких свойств моделей xMAS, как задержка передачи пакета и пропускная способность канала. Разработанные алгоритмы будут применены для исследования характеристик качества обслуживания промышленных коммуникационных фабрик компании Intel.

Список литературы

1. **International** technology roadmap for semiconductors, Semiconductor Industry Association. 2003. URL: http://www.itrs.net/Links/2003ITRS/ExecSum2003.pdf (01.03.2012).

2. **Garver S., Crepps B.** The new era of Tera-scale computing. URL: http://software.intel.com/en-us/articles/the-new-era-of-tera-scale-computing (11.10.2011).

3. **Marescaux T., Briek'el B.** et al. Dynamic Time-Slot Allocation for QoS Enabled Networks on Chip // Proc. Embedded Systems for Real-Time Multimedia 3rd workshop. 2005. P. 47—51.

4. **PCI-Express** base specification rev. 1.1, PCI-SIG, 28. March 2005. URL: http://www.pcisig.com/specifications/pciexpress/base (01.03.2012).

5. **Bjerregaard T., Mahadevan S.** A survey of research and practices of network-on-chip // ACM Computing Surveys. 2006. 38 (1). P. 1—51.

6. **Internetworking** Technology Handbook, CISCO Systems. URL: http://docwiki.dsco.com/wiki/Internetworking_Technology_Handbook (25.12.2011).

7. **Wang G., Wang C.** et al. Quality of Service (QoS) Contract Specification, Establishment, and Monitoring for Service Level Agreement // Proc. EDOCW'06. 2006.

8. **Salminen E., Srinivasan K., Lu Z.** OCP-IP Network-on-chip benchmarking workgroup Dec 2010. URL: http://www.ocpip.org/ (25.12.2011).

9. **Bakhouya M., Suboh S.** et al. Analytical Modeling and Evaluation of On-Chip Interconnects Using Network Calculus // Proc. NoCS 3rd ACM/IEEE International Symposium. 2009. P. 74—79.

10. **Le Boudec J. Y., Thiran P.** Network Calculus, A Theory of Deterministic Queuing Systems for the Internet. Springer-Verlang. 2001.

11. **Викторов Ю.** Анализ и оптимизация качества обслуживания в системах на кристалле // XXXVII Гагаринские чтения. 2011. Т. 4. С. 42—44.

12. **Varatkar G., Marculescu R.** Traffic analysis for on-chip networks design of multimedia applications // Proc. ACM/IEEE Design Automation Conference, June 2002. P. 510—517.

13. **ISO/IEC** International Standard 13236: Information Technology — Quality of Service: Framework, First Edition, International Organization for Standartization, Dec 1998. URL: http://webstore.iec.ch/preview/info_isoiecl3236 %7Bedl.0 %7Den.pdf (01.03.2012).

14. **ISO/IEC** Technical Report 13243: Information Technology — Quality of Service: Guide to Methods and Mechanisms, First Edition, International Organization for Standartization, Nov 1999. URL: http://webstore.iec.ch/preview/infoisoiecl3243 %7Bedl.0 %7Den.pdf (01.03.2012).

15. **Traffic** Management Specification, version 4.1 AF-TM-0121.000 // The ATM Forum, Technical Committee, Mar 1999. URL: http://www.broadband-forum.org/ftp/pub/approved-specs/af-tm-0121.000.pdf (01.03.2012).

16. **Wang G., Wang C.** et al. Service Level Management using QoS Monitoring, Diagnostics, and Adaptation for Networked Enterprise Systems // Proc. IEEE EDOC'05. 2005. P. 239—248.

17. **Wu J., Foster G.** A QoS Mapping Algorithm for ETE QoS Provisioning // SPIE. 2002. Vol. 4911. P. 256—264.

18. **Varshavsky V.** Time, Timing and Clock in Massively Parallel Computing System // Proc. International Conference Massively Parallel Computing Systems, April 6—9. 1998.

19. **Qian Y., Lu Z., Dou W.** Analysis of communication delay bounds for networks on chip // Proc. ASPDAC'09, Jan. 2009. P. 7—12.

20. **Cohen I., Rottenstreich P., Keslassy I.** Statistical Approach to NoC Design // Proc. Second ACM/IEEE Int. Symposium on NoC. 2008. P. 171—180.

21. **Shi Z., Burns A.** Real-Time Communication Analysis for On-Chip Networks with Wormhole Switching // Proc. Second ACM/IEEE Int. Symposium on NoC, 2008. P. 161—170.

22. **Rahmati P., Murali S.** et al. A Method for Calculating Hard QoS Guarantees for Networks-on-Chip // Proc. ICCAD'09, Nov. 2009. P. 579—586.

23. **Jantsch A., Lu Z.** Networks on Chip: Theory and Practice, chapter Resource allocation for quality of service on-chip communication. Taylor & Francis Group LLC — CRC Press, 2009.

24. **Bolotin E., Cidon I., Ginosar R., Kolodny A.** QNoC: QoS architecture and design process for network on chip // The Journal of Systems Architecture. December 2003. P. 105—128.

25. **Lu Z., Jantsch A.** TDM virtual-circuit configuration for network-on-chip // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2008. 16 (8). P. 1021—1034.

26. **Kishinevsky M., Gotmanov A., Viktorov Y.** Challenges in Verifying Communication Fabrics // Proc. ITP'11. August 2011. P. 18—21.

27. **Dally W. J., Towles B.** Route packets, not wires: On-chip interconnection networks // Proc. Design Automation Conference. 2001. P. 684—689.

28. **Ogras U., Hu J., Marculescu R.** Key research problems in NoC design: A holistic perspective // Proc. of the Intl. Conf. on Hardware/Software Codesign and System Synthesis, September 2005. P. 69—74.

29. **Amde M., Felicijan T.** et al. Asynchronous on-chip networks. IEE Proceedings Computers and Digital Techniques. March 2005. 152 (02). P. 273—283.

30. **Soteriou V., Wang H., Peh Li-Shiuan.** A statistical traffic model for on-chip interconnection networks // Proc. MASCOTS '06. September 2006. P. 104—116.

31. **Sanguinetti J., Pursley D.** High-Level Modeling and Hardware Implementation with General-Purpose Languages and High-level Synthesis, April. 2002. URL: http://www.cynapps.com/products/paper_0402_highlevelmodeling.pdf (01.03.2012).

32. **Das R., Mutlu O.** et al. A Network-on-Chip Exploiting Packet Latency Slack // IEEE Micro. Jan./Feb. 2011. Vol. 31. N 1. P. 29—41.

33. **McMillan K. L.** The SMV Language. 1998. URL: http://santos.cis.ksu.edu/smv-doc/language/language.html (15.12.2011).

34. **Dill D. L., Drexler A. J., Hu A. J., Han Yang C.** Protocol Verification as a Hardware Design Aid // Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors, IEEE Computer Society. 1992. P. 522—525.

35. **Galton A.** et al. Temporal Logic. Feb. 2008. URL: http://plato.stanford.edu/entries/logic-temporal/ (15.12.2011).

36. **Sheeran M.** et al. Checking Safety Properties Using Induction and a SAT-Solver // FMCAD 2000, LNCS 1954. P. 108—125.

37. **Chaudhuri K., Doligezet D.** et al. A TLA+ Proof System, CoRR, Vol. abs/0811.1914. 2008. URL: http://hal.inria.fr/inria-00338299/en/ (10.01.2012).

38. **Borrione P., Helmy A.** et al. A formal approach to the verification of Networks on EURASIP Journal on Embedded Systems. 2009. Vol. 2009. P. 299—304.

39. **Готманов А., Кишиневский М., Галсеран-Омс М.** Технология построения синхронных эластичных схем и ее применение к оптимизации производительности аппаратного декодера H.264 CABAC // МЭС 2010. Окт. 2010. С. 8—13.

40. **Hansson A.** et al. Enabling Application-Level Performance Guarantees in Network-Based Systems on Chip by Applying Dataflow Analysis // Computers & Digital Techniques, IET. September 2009. P. 398—412.

41. **Chatterjee S., Kishinevsky M., Ogras U. Y.** Quick formal modeling of communication fabrics to enable verification // Proc. HLDVT. 2010. P. 108—125.

42. **ABC** Berkeley — A System for Sequential Synthesis and Verification. http://www.eecs.berkeley.edu/~alanmi/abc/ (15.12.2011).

43. **Biere A., Cimatti A.** et al. Symbolic Model Checking without BDD // Proc. TACAS'99, Jan. 1999. P. 193—207.

44. **McMillan K. L.** Interpolation and SAT-based Model Checking // Proc. CAV'03, 2003. P. 1—13.

45. **Chatterjee S., Kishinevsky M.** Automatic generation of inductive invariants from highlevel microarchitectural models of communication fabrics // Computer Aided Verification'10. 2010. LNCS, vol. 6174. P. 321—338.

46. **Gotmanov A., Chatterjee S., Kishinevsky M.** Verifying deadlock-freedom of communication fabrics // VMCAI'11, 2011. LNCS, vol. 6538. P. 214—231.

47. **Dally W. J., Towles B.** Principles and Practices on Interconnection Networks, Morgan Kaufmann Publishers, 2004.

УДК 621.391

Н. И. Червяков,
д-р техн. наук, проф., зав. каф.,
В. М. Авербух,
д-р техн. наук, проф., зав. сектором,
М. Г. Бабенко,
канд. физ.-мат. наук, мл. науч. сотр.,
e-mail: whbear@yandex.ru,
И. Н. Лавриненко,
канд. физ.-мат. наук, доц.,
П. А. Ляхов, аспирант,
e-mail: ljahov@mail.ru,
Ставропольский государственный университет

Эффективный метод приближенного вычисления позиционной характеристики модулярного представления числа*

Предлагается новый эффективный метод определения позиционной характеристики числа в системе остаточных классов на основе использования относительной величины числа и показаны способы его применения для реализации важнейших немодульных операций.

Ключевые слова: *система остаточных классов, позиционная характеристика, численный метод, приближенный метод, периодическая дробь*

Введение

Современное состояние развития инфокоммуникационных технологий в области обработки и передачи данных характеризуется интенсивным внедрением новых принципов и подходов к обработке информации. Результаты теоретических и практических разработок отечественных и зарубежных специалистов со всей определенностью указывают на то, что одним из перспективных многообещающих путей решения задач сокращения времени обработки данных и повышения надежности вычислительных средств является применение различных форм параллельной обработки данных, в том числе и на основе числовых систем с параллельной структурой. Одним из магистральных направлений среди современных подходов к созданию отказоустойчивых высокопроизводительных средств обработки данных является использование системы остаточных классов (СОК).

* Работа выполнена при поддержке гранта РФФИ 12-07-00482-а.

Арифметика СОК долгое время привлекала интерес только на теоретическом уровне из-за сложности архитектур, определяемых использованием представляемых данных. Однако быстрый рост технологий вычислительной базы делает СОК удобным для многих приложений цифровой обработки данных, криптографии, систем передачи данных, основанных на множественном доступе с кодовым разделением каналов и др.

Главным преимуществом СОК является разложение динамического диапазона на параллельные каналы с меньшими динамическими диапазонами, определяемыми выбором оснований СОК, которые приводят к операциям без переноса между каналами с различными основаниями и сокращению задержек сигналов.

В качестве недостатка СОК можно отметить трудность выполнения немодульных операций.

Система остаточных классов

Основной теоретико-числовой базой системы остаточных классов является теория сравнений. Полной системой вычетов по модулю p называется совокупность p целых чисел, содержащая точно по одному представителю из каждого класса вычетов по модулю p. Каждый класс вычетов по модулю p содержит в точности одно из чисел совокупности всех возможных остатков от деления на p: $\{0,1, ..., p-1\}$. Множество $\{0,1, ..., p-1\}$ также называется полной системой наименьших неотрицательных вычетов по модулю p.

Один из методов выполнения арифметических операций над длинными целыми числами основан на простых положениях теории чисел. Представление чисел в СОК позволяет заменить операции с большими числами на операции с малыми числами, которые представлены в виде остатков от деления больших чисел на заранее выбранные взаимно простые модули $p_1, p_2, ..., p_n$. Пусть

$$A \equiv \alpha_1(\mathrm{mod}\, p_1),\ A \equiv \alpha_2(\mathrm{mod}\, p_2),\ ...,\ A \equiv \alpha_n(\mathrm{mod}\, p_n). \quad (1)$$

Тогда целому числу A можно поставить в соответствие кортеж $(\alpha_1, \alpha_2, ..., \alpha_n)$ наименьших неотрицательных вычетов по одному из соответствующих классов. Данное соответствие будет взаимно однозначным, пока $A < p_1p_2, ..., p_n$, в силу Китайской теоремы об остатках (КТО). Кортеж $(\alpha_1, \alpha_2, ..., \alpha_n)$ можно рассматривать как один из способов представления целого числа A в ЭВМ — модулярное представление или представление в СОК.

Основным преимуществом такого представления является тот факт, что выполнение операций

сложения, вычитания и умножения реализуется очень просто, по следующим формулам:

$$A \pm B = (\alpha_1, \alpha_2, ..., \alpha_n) \pm (\beta_1, \beta_2, ..., \beta_n) =$$
$$= ((\alpha_1 \pm \beta_1) \bmod p_1 (\alpha_2 \pm \beta_2) \bmod p_2, ..., (\alpha_n \pm \beta_n) \bmod p_n);$$
$$A \times B = (\alpha_1, \alpha_2, ..., \alpha_n) \times (\beta_1, \beta_2, ..., \beta_n) =$$
$$= ((\alpha_1 \times \beta_1) \bmod p_1, (\alpha_2 \times \beta_2) \bmod p_2, ...,$$
$$(\alpha_n \times \beta_n) \bmod p_n). \qquad (2)$$

Эти операции носят название *модульных*, так как для их выполнения в СОК достаточно одного такта обработки числовых значений, причем эта обработка происходит параллельно, и значения информации в каждом разряде не зависят от других разрядов.

Основной недостаток модулярного представления чисел состоит в том, что трудно упорядочить множество всех целочисленных кортежей длины n так, чтобы этот порядок соответствовал естественному порядку на множестве целых чисел. Как следствие этого факта, трудно установить, является ли кортеж $(\alpha_1, \alpha_2, ..., \alpha_n)$ большим (меньшим), чем $(\beta_1, \beta_2, ..., \beta_n)$. Трудно также проверить, возникло ли переполнение допустимого диапазона чисел $P = p_1 p_2 ... p_n$ в результате выполнения операций сложения или умножения, но еще труднее выполнить операцию деления. Эти и некоторые другие операции носят название *немодульных*, так как для их выполнения требуется знание о величине числа в целом, которое называется позиционной характеристикой числа.

Модель целочисленной модулярной арифметики можно задать следующей сигнатурой:

$$\langle |P|, |\bullet|_{p_i}^{+}, \text{МО}, \text{НО} \rangle,$$

где $|P|$ — полная система вычетов по модулю полного динамического диапазона; $|\bullet|_{p_i}^{+}$ — вычет чисел по модулю p_i; МО — множество модульных операций, к которым относятся арифметические операции сложения, вычитания, умножения и деления нацело или умножение на обратный элемент; НО — множество немодульных операций, к которым относятся операции определения знаков чисел и переполнения динамического диапазона, сравнение, определение интервалов чисел, определение и локализация ошибочного разряда и др.

Немодульные операции обусловлены знанием числового значения модулярной величины, которая определенным образом связана со значениями компонент модулярного представления. Эти операции являются медленными, что снижает эффективность применения модулярной алгебры. Для реализации немодульных операций используются специальные функционалы, которые определяют количественные характеристики отношения порядка над множеством модулярных векторов. Одно из устоявшихся названий функционалов — позиционная характеристика (ПХ) модулярной величины или числовой величины в модулярном коде. В основе алгоритмов выполнения немодульных операций лежат методы вычисления ПХ, сложность которых непосредственно влияет на скорость выполнения немодульных операций в модулярной алгебре. Поиск эффективных и универсальных ПХ важен для теоретических основ модулярных вычислительных структур и вычислительных средств на их основе.

В настоящее время известны следующие методы определения позиционных характеристик модулярного представления чисел [1—3]:

- метод ортогональных базисов;
- метод интервальных оценок;
- метод с использованием коэффициентов обобщенной позиционной системы счисления (ОПСС) и др.

Суть метода ортогональных базисов состоит в переходе к позиционному представлению через модуль всей системы $P = p_1 p_2 ... p_n$, что разрушает идею модульной арифметики. Недостаток метода ортогональных базисов заключается в том, что приходится иметь дело с большими числами ортогональных базисов и, кроме того, действия сложения и умножения надо выполнять в позиционной системе счисления, а полученный результат необходимо вводить в диапазон вычитанием величины, кратной P, определяемой рангом числа, определение которого также связано с вычислительной сложностью.

Метод интервальных оценок сокращает модуль до величины $\frac{P}{p_i}$, и только метод ОПСС позволяет выполнять преобразования по модулю p_i.

Метод ОПСС является универсальным. К недостаткам этого метода можно отнести его сложность и избыточность позиционной характеристики при вычислении некоторых немодульных процедур.

Метод ортогональных базисов и метод ОПСС являются точными методами определения позиционных характеристик, а метод интервальных оценок является приближенным методом, к характеристике которого относится позиционная характеристика номера интервала числа. Суть метода интервальных оценок состоит в том, что числовой диапазон P разбивается на p_i интервалов

$$\left[j\frac{P}{p_i}, (j+1)\frac{P}{p_i} \right], j = 1, 2, ..., p_i - 1. \qquad (3)$$

Определение номера интервала, в котором расположено число, позволяет получить оценку немодульного числа по его значению с точностью до интервала, что ограничивает область его применения.

Процесс определения знака числа сводится к операции выявления принадлежности интервала, в котором находится число, представленное в СОК, к группе положительных или отрицательных интервалов по заданному p_i, на которые разбит диапазон P. Число интервалов определяется величиной $\frac{P}{p_i}$. Если диапазон разбивается на нечетное число интервалов по выбранному модулю p_i, т. е. все основания нечетные, то имеется критический интервал, который является границей между положительными и отрицательными числами. В этом случае критический интервал делится на две части, а процесс определения знака числа при этом сводится к сравнению остатка по модулю p_i, что резко усложняет процесс определения знака числа.

В целях повышения эффективности вычисления позиционной характеристики предлагается новый приближенный метод определения позиционной характеристики, который позволяет реализовать практически все немодульные процедуры модулярного кода.

Исторически так сложилось, что поиск некоторого компромисса в удовлетворении требований, предъявляемых к ПХ, привел исследователей к введению таких характеристик модулярной алгебры, как ранг, след, нормированный ранг, неточный ранг, ядро числа и др. [1, 2, 3, 6]. Анализ этих ПХ показал, что значение модулярной величины по ним определяется сложно и не всегда однозначно. Кроме того, при выполнении некоторых НО нет необходимости в точном их определении, а достаточно знать значения в пределах каких-то интервалов, т. е. при определении этих характеристик появляется избыточная информация, которая не используется. Эта идея и подтолкнула к поиску позиционной характеристики, не содержащей избыточной информации, на нахождение которой требуются дополнительные вычислительные ресурсы.

Приближенный метод вычисления позиционной характеристики числа на основе использования относительных величин

Анализ немодульных операций показал, что их можно представить точно или приближенно, поэтому методы вычисления позиционных характеристик можно разделить на две группы:

- методы точного вычисления позиционных характеристик;
- методы приближенного вычисления позиционных характеристик.

Методы точного вычисления позиционных характеристик рассмотрены в работах [1—3]. В данной статье исследуются приближенные методы вычисления позиционных характеристик, которые позволяют существенно сократить аппаратные и временные затраты, обусловленные операциями, выполняемыми над позиционными кодами уменьшенной разрядности. В связи с этим возникает задача использования приближенного метода при вычислении определенного ряда немодульных процедур: определения интервалов чисел, знака числа, сравнения чисел, в том случае когда не требуется знания точного значения, насколько одно число больше или меньше другого.

Суть приближенного метода вычисления позиционных характеристик основан на использовании относительных величин анализируемых чисел к полному диапазону:

$$\frac{A}{P} = \left| \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} \alpha_i \right|_1 \approx \left| \sum_{i=1}^{n} K_i \alpha_i \right|_1, \qquad (4)$$

где A — исследуемое число; p_i — модули СОК; $\left|P_i^{-1}\right|_{p_i}$ — мультипликативная инверсия P_i относительно p_i; $P_i = \frac{P}{p_i} = p_1 p_2 ... p_{i-1} p_{i+1} ... p_n$; $K_i = \frac{\left|P_i^{-1}\right|}{p_i}$ — константы выбранной системы; α_i — разряды числа, представленного в СОК; $|\bullet|_1$ означает дробную часть выражения.

Приближенный метод вычисления позиционной характеристики описывается следующей последовательностью действий [4, 5]:

1. Вычисление констант СОК $k_i = \frac{\left|P_i^{-1}\right|}{p_i} p_i$ с требуемой точностью.

2. Вычисление приближенных значений $\alpha_i k_i$ и запись их в LUT-память вычислительной системы, где k_i — константы, найденные в п. 1, $1 \leqslant \alpha_i \leqslant p_i - 1$. Адресами выборки значений $\alpha_i k_i$ являются разряды СОК α_i, где $i = 1, ..., n$.

3. Вычисление приближенного значения позиционной характеристики $\left| \sum_{i=1}^{n} K_i \alpha_i \right|_1$ в интервале [0, 1). Конечный результат определяется после суммирования и отбрасывания целой части числа с сохра-

нением дробной части суммы. Тогда позиционная характеристика определяется в виде относительного значения величины:

$$\frac{A}{P} = \left| \sum_{i=1}^{n} \frac{\left|P_i^{-1}\right|_{p_i}}{p_i} \alpha_i \right|_1 \approx \left| \sum_{i=1}^{n} K_i \alpha_i \right|_1 .$$

4. Конструируются некоторые правила Ψ_i, $i = 1, ..., 4$, согласно которым вычисляется i-я немодульная операция (определение знака числа, сравнение чисел, обнаружение ошибки и переполнения, а также локализация ошибочного разряда).

Правило Ψ_1. Определение знака числа в случае, если $p_1 = 2$:

- если $\frac{A}{P} < \frac{1}{2}$, то число положительное;
- если $\frac{A}{P} > \frac{1}{2}$, то число отрицательное.

Правило Ψ_2. Сравнение модулярных чисел A и B:

- если $\frac{A}{P} - \frac{B}{P} = 0$, то $A = B$;
- если $\frac{A}{P} - \frac{B}{P} > 0$, то $A > B$;
- если $\frac{A}{P} - \frac{B}{P} < 0$, то $A < B$.

Правило Ψ_3. Обнаружение ошибки и переполнения динамического диапазона:

- если $\frac{\overline{A}}{P_{\text{изб}}} < \frac{M}{P_{\text{изб}}}$, тогда ошибки нет, где $\overline{A}$ — искаженное число; $P_{\text{изб}} = p_{n+1}p_{n+2}P$ — избыточный диапазон при двух избыточных модулях p_{n+1} и p_{n+2}; $M = P = \prod_{i=1}^{n} p_i$ — рабочий диапазон;
- если $\frac{\overline{A}}{P_{\text{изб}}} \geqslant \frac{M}{P_{\text{изб}}}$, тогда есть ошибка и установлено переполнение динамического диапазона.

Правило Ψ_4. Локализация неисправного канала:

- если $\frac{\overline{A}_i}{P_i} < \frac{M_i}{P_i}$, то в разряде i нет ошибки;
- если $\frac{\overline{A}_i}{P_i} \geqslant \frac{M_i}{P_i}$, то в разряде i есть ошибка, где $\overline{A}_i = (\alpha_1, \alpha_2, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_n, \alpha_{n+1}, \alpha_{n+2})$ — проекция искаженного числа $\overline{A}$; $M_i = (m_1, m_2, ..., m_{i-1}, m_{i+1}, ..., m_n, m_{n+1}, m_{n+2})$ — проекция рабочего диапазона.

Проблема синтеза немодульных устройств на основе предложенного приближенного метода побуждает к разработке таких правил Ψ_i, которые минимизировали бы число операций и вместе с тем могли бы быть достаточно просто реализованы на современной элементной базе.

В зависимости от типа немодульной операции применяются либо точные, либо приближенные методы или их комбинация. При необходимости можно использовать приближенные методы и для точных вычислений. В первую очередь необходимо исследовать возможность использования данного приближенного метода для восстановления остаточного кода.

Использование дробных величин при реализации приближенного метода вычисления позиционной характеристики

При использовании двоичной системы счисления воспользуемся некоторыми свойствами двоичного представления дробей $\frac{\alpha_i}{p_i}$, где α_i — разряды числа в СОК; p_i — основания СОК, которые выбираются нечетными, $i = [1, n]$, $1 \leqslant \alpha_i \leqslant p_i - 1$, которые исследованы в работе [6].

Поскольку $(p_i, 2) = 1$, то дробь $\frac{\alpha_i}{p_i}$ разлагается в бесконечную периодическую дробь по степеням двойки.

Определение периодов, отвечающих каждому значению α_i, может быть осуществлено, как в работе [6]. Положим $\alpha_i = 1$ и образуем последовательности типа $|\alpha_1 2^0|_{p_i}$, $|\alpha_1 2^1|_{p_i}$, $|\alpha_1 2^2|_{p_i}$, ..., $|\alpha_1 2^{\tau_1}|_{p_i} = \alpha_1$. Полученная последовательность вычетов имеет период τ_1. Вычет α_1 назовем начальным элементом группы. Далее берем младший вычет α_2, не попавший в первую группу, и формируем для него новую последовательность $|\alpha_2 2^0|_{p_i}$, $|\alpha_2 2^1|_{p_i}$, $|\alpha_2 2^2|_{p_i}$, ..., $|\alpha_2 2^{\tau_2}|_{p_i}^{+} = \alpha_2$. Данная последовательность имеет период τ_2. Вычет α_2 — начальный элемент второй группы. Продолжив эту процедуру, разобьем $|\bullet|_{p_i}^{+}$ на группы вычетов, каждая из которых имеет свой период. При этом должно выполняться $\tau_1 + \tau_2 + ... + \tau_s = p_i - 1$.

Пример 1. Определить периоды вычетов для оснований $p = 21; 23; 7; 13; 3; 5$.

1) $p = 21$.

k	0	1	2	3	4	5	6
$\lvert\alpha_1 2^k\rvert_p^+$	1	2	4	8	16	11	1

$\tau_1 = 6$

k	0	1	2	3
$\lvert\alpha_2 2^k\rvert_p^+$	3	6	12	3

$\tau_2 = 3$

k	0	1	2	3	4	5	6
$\lvert\alpha_3 2^k\rvert_p^+$	5	10	20	19	17	13	5

$\tau_3 = 6$

k	0	1	2
$\lvert\alpha_4 2^k\rvert_p^+$	7	14	7

$\tau_4 = 2$

k	0	1	2	3
$\lvert\alpha_5 2^k\rvert_p^+$	9	18	15	9

$\tau_5 = 3$

Анализ всех групп вычетов показывает, что в данных таблицах представлены все вычеты по модулю 21. На этом процесс построения прекращается. При сложении всех τ_i имеем:

$$\tau_1 + \tau_2 + \tau_3 + \tau_4 + \tau_5 = \\ = 6 + 3 + 6 + 2 + 3 = 20 = p - 1.$$

Аналогично проведем построение для остальных модулей.

2) $p = 23$.

k	0	1	2	3	4	5	6	7	8	9	10	11
$\lvert\alpha_1 2^k\rvert_p^+$	1	2	4	8	16	9	18	13	3	6	12	1

$\tau_1 = 11$

k	0	1	2	3	4	5	6	7	8	9	10	11
$\lvert\alpha_2 2^k\rvert_p^+$	5	10	20	17	11	22	21	29	15	7	14	5

$\tau_2 = 11$

Окончательно имеем $\tau_1 + \tau_2 = 11 + 11 = 22 = p - 1$.

3) $p = 7$.

k	0	1	2	3
$\lvert\alpha_1 2^k\rvert_p^+$	1	2	4	1

$\tau_1 = 3$

k	0	1	2	3
$\lvert\alpha_2 2^k\rvert_p^+$	3	6	5	3

$\tau_2 = 3$

$$\tau_1 + \tau_2 = 6 = p - 1.$$

Для модулей, у которых число 2 является первообразным корнем, соответственно имеем:

4) $p = 13$.

k	0	1	2	3	4	5	6	7	8	9	10	11	12
$\lvert\alpha_1 2^k\rvert_p^+$	1	2	4	8	3	6	12	11	9	5	10	7	1

$\tau_1 = 12$

$$\tau_1 = 12 = p - 1.$$

5) $p = 3$.

k	0	1	2
$\lvert\alpha_1 2^k\rvert_p^+$	1	2	1

$\tau_1 = 2$

$$\tau_1 = 2 = p - 1.$$

6) $p = 5$.

k	0	1	2	3	4
$\lvert\alpha_1 2^k\rvert_p^+$	1	2	4	3	1

$\tau_1 = 4$

$$\tau_1 = 4 = p - 1.$$

Проведем перекодировку дроби $\frac{\alpha}{p}$, $(0 < \alpha < p)$ в двоичную дробь.

Для формирования двоичного разложения дробей $\frac{\alpha}{p}$ достаточно указать (в пределах одного периода) разложение дробей $\frac{\alpha'}{p}$, где α' — начальные вычеты группы вычетов периода τ'. Пусть, например, α_0 — начальный вычет группы вычетов периода τ. Тогда последовательность вычетов этой группы имеет вид α_0, $\alpha_1 = \varphi(\alpha_0)$, $\alpha_2 = \varphi(\alpha_0^2)$, ..., $\alpha_{\tau-1} = \varphi(\alpha_0^{\tau-1})$. Соответственно, последовательность цифр двоичного разложения "начальной" дроби $\frac{\alpha_0}{p}$ в пределах одного периода имеет вид $\theta(\alpha_0), \theta(\alpha_1), ..., \theta(\alpha_{\tau-1})$, где

$$\theta(\alpha_i) = \begin{cases} 1, \text{ если } \alpha_i > \frac{p}{2}; \\ 0, \text{ если } \alpha_i < \frac{p}{2}. \end{cases}$$

Отсюда следует правило разложения дроби $\frac{\alpha_k}{p}$, $(\alpha_k = \varphi(\alpha_0^k))$ в двоичную дробь любой длины. Приведенные таблицы служат для определения констант.

Пример 2. Провести перекодировку дробей из примера 1.

1) $p = 21$.

$\alpha_1 = 1$, $\tau = 6$.

k	0	1	2	3	4	5
α_k	1	2	4	8	16	11
$\theta(\alpha_k)$	0	0	0	0	1	1

$\alpha_2 = 3$, $\tau = 3$.

k	0	1	2
α_k	3	6	12
$\theta(\alpha_k)$	0	0	1

$\alpha_3 = 5$, $\tau = 6$.

k	0	1	2	3	4	5
α_k	5	10	20	19	17	13
$\theta(\alpha_k)$	0	0	1	1	1	1

$\alpha_4 = 7$, $\tau = 2$.

k	0	1
α_k	7	14
$\theta(\alpha_k)$	0	1

$\alpha_5 = 9$, $\tau = 3$.

k	0	1	2
α_k	9	18	15
$\theta(\alpha_k)$	0	1	1

На основе полученных таблиц представим некоторые дроби $\frac{\alpha}{p}$ в двоичном виде:

$$\frac{4}{21} = 0,\underbrace{001100}_{\tau}\underbrace{001100}_{\tau}...\underbrace{001100}_{\tau}... ,$$

$$\frac{17}{21} = 0,\underbrace{1101}_{\tau}\underbrace{1110}_{\tau}...\underbrace{11011}_{\tau}... ,$$

$$\frac{3}{21} = 0,\underbrace{001}_{\tau}\underbrace{001}_{\tau}...\underbrace{001}_{\tau}... , \quad \frac{18}{21} = 0,\underbrace{110}_{\tau}\underbrace{110}_{\tau}...\underbrace{110}_{\tau}... ,$$

$$\frac{7}{21} = 0,\underbrace{01}_{\tau}\underbrace{01}_{\tau}...\underbrace{01}_{\tau}... , \quad \frac{14}{21} = 0,\underbrace{10}_{\tau}\underbrace{10}_{\tau}...\underbrace{10}_{\tau}... ,$$

2) $p = 13$.

$\alpha_1 = 1$, $\tau = 12$.

k	0	1	2	3	4	5	6	7	8	9	10	11
α_k	1	2	4	8	3	6	12	11	9	5	10	7
$\theta(\alpha_k)$	0	0	0	1	0	0	1	1	1	0	1	1
	τ'						τ''					

В случае, когда p — простое число и 2 — первообразный корень, вторая половина периода дроби $\frac{1}{p}$ оказывается двойственной к первой. Эта особенность позволит уменьшить объем памяти в два раза при хранении дробных чисел.

Представим двоичные дроби, например $\frac{1}{13}$ и $\frac{9}{13}$:

$$\frac{1}{13} = 0,\underbrace{\underbrace{0001}_{\tau'}\underbrace{0011}_{\tau''}}_{\tau}\underbrace{101101}_{\tau}...\underbrace{010011101100}_{\tau}... ,$$

$$\frac{9}{13} = 0,\underbrace{101100}_{\tau}\underbrace{101101}_{\tau}...\underbrace{101100010011}_{\tau}... .$$

3) $p = 23$.

$\alpha_1 = 1$, $\tau = 11$.

k	0	1	2	3	4	5	6	7	8	9	10
α_k	1	2	4	8	16	9	18	13	3	6	12
$\theta(\alpha_k)$	0	0	0	0	1	0	1	1	0	0	1

$\alpha_2 = 5$, $\tau = 11$.

k	0	1	2	3	4	5	6	7	8	9	10
α_k	5	10	20	17	11	22	21	19	15	7	14
$\theta(\alpha_k)$	0	0	1	1	0	1	1	1	1	0	1

Представим двоичные дроби, например $\frac{16}{23}$ и $\frac{7}{23}$:

$$\frac{16}{23} = \underbrace{0,10110010000}_{\tau}\underbrace{10110010000}_{\tau}...\underbrace{10110010000}_{\tau}... ,$$

$$\frac{7}{23} = \underbrace{0,01001101111}_{\tau}\underbrace{01001101111}_{\tau}...\underbrace{01001101111}_{\tau}... ,$$

4) $p = 7$.

$\alpha_1 = 1$, $\tau = 3$.

k	0	1	2
α_k	1	2	4
$\theta(\alpha_k)$	0	0	1

$\alpha_2 = 3$, $\tau = 3$.

k	0	1	2
α_k	3	6	5
$\theta(\alpha_k)$	0	1	1

Представим двоичную дробь, например $\frac{4}{7}$:

$$\frac{4}{7} = 0,\underbrace{100}_{\tau}\underbrace{100}_{\tau}...\underbrace{100}_{\tau}... ,$$

5) $p = 3$.

$\alpha_1 = 1$, $\tau = 2$.

k	0	1
α_k	1	2
$\theta(\alpha_k)$	0	1

Представим двоичную дробь, например $\frac{1}{3}$:

$$\frac{1}{3} = 0,\underbrace{01}_{\tau}\underbrace{01}_{\tau}...\underbrace{01}_{\tau}... ,$$

6) $p = 5$.

$\alpha_1 = 1$, $\tau = 4$.

k	0	1	2	3
α_k	1	2	4	3
$\theta(\alpha_k)$	0	0	1	1

Представим двоичную дробь, например $\frac{3}{5}$:

$$\frac{3}{5} = 0,\underbrace{1001}_{\tau}\underbrace{1001}_{\tau}...\underbrace{1001}_{\tau}... .$$

Таким образом, таблицы разложения начальных дробей в двоичную дробь в пределах периода могут служить исходной информацией для разложения любой дроби $\frac{\alpha}{p}$ в двоичную дробь произвольной длины, которая в дальнейшем используется при сложении двоичных вычетов.

Пример 3. Пусть задана система оснований $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$. Объем диапазона СОК равен $P = 2 \cdot 3 \cdot 5 \cdot 7 = 210$. При этом величины P_i будут иметь следующие значения: $P_1 = \frac{P}{p_1} = 105$, $P_2 = \frac{P}{p_2} = 70$, $P_3 = \frac{P}{p_3} = 42$, $P_4 = \frac{P}{p_4} = 30$.

Для примера возьмем наихудший случай, когда необходимо сравнить два соседних числа. Сравним два числа $A_1 = 1$ и $A_2 = 2$, представленные в СОК по основаниям p_1, p_2, p_3, p_4. Тогда $A_1 = (1, 1, 1, 1)$, $A_2 = (0, 2, 2, 2)$. Для решения задачи определим константы $K_i = \frac{\left|P_i^{-1}\right|}{p_i}$, представленные в виде дробей

$$K_1 = \frac{\left|\frac{1}{105}\right|_2}{2} = \frac{1}{2};\ K_2 = \frac{\left|\frac{1}{70}\right|_3}{3} = \frac{1}{3};$$

$$K_3 = \frac{\left|\frac{1}{42}\right|_5}{5} = \frac{3}{5};\ K_4 = \frac{\left|\frac{1}{30}\right|_7}{7} = \frac{4}{7}.$$

Представим константы K_i в виде десятичных и двоичных дробей, тогда для десятичных дробей константы:

$$K_1 = 0,5;\ K_2 = 0,3333...;\ K_3 = 0,6;\ K_4 = 0,5714...,$$

а двоичные значения констант для модулей 3,5,7 возьмем из примера 2:

$$K_1 = 0,1;\ K_2 = 0,0101...;\ K_3 = 0,1011...;\ K_4 = 0,1001... .$$

Используя выражение (4), найдем $\frac{A_1}{P}$ и $\frac{A_2}{P}$. Для этого вычислим приближенные произведения $K_i\alpha_i$ в виде десятичных и двоичных дробей.

Десятичные дроби

Вычислим $\frac{A_1}{P} \approx |1 \cdot 0,5 + 1 \cdot 0,3333 + 1 \cdot 0,6 + 1 \cdot 0,5714|_1 = |0,5 + 0,3333 + 0,6 + 0,5714|_1$.

Далее найдем сумму полученных приближенных чисел, округляя дроби до 1, 2, 3 и 4 знаков после запятой:

$$\begin{array}{|l} 0,5 \\ +\ 0,3 \\ \ \ 0,6 \\ \ \ 0,6 \end{array}_1 \quad \begin{array}{|l} 0,5 \\ +\ 0,33 \\ \ \ 0,6 \\ \ \ 0,57 \end{array}_1 \quad \begin{array}{|l} 0,5 \\ +\ 0,333 \\ \ \ 0,6 \\ \ \ 0,571 \end{array}_1 \quad \begin{array}{|l} 0,5 \\ +\ 0,3333 \\ \ \ 0,6 \\ \ \ 0,5714 \end{array}_1 .$$

$$0,0 \qquad 0,00 \qquad 0,004 \qquad 0,0047$$

Из четырех возможных способов, очевидно, четвертый является наиболее точным.

Аналогичным образом поступим с числом A_2:

$$\frac{A_2}{P} \approx |0 \cdot 0,5 + 2 \cdot 0,3333 + 2 \cdot 0,6 + 2 \cdot 0,5714|_1 = |0,6666 + 0,2 + 0,1428|_1;$$

$$\begin{array}{|l} 0,7 \\ +\ 0,2 \\ \ \ 0,1 \end{array}_1 \quad \begin{array}{|l} 0,67 \\ +\ 0,2 \\ \ \ 0,14 \end{array}_1 \quad \begin{array}{|l} 0,667 \\ +\ 0,2 \\ \ \ 0,143 \end{array}_1 \quad \begin{array}{|l} 0,6666 \\ +\ 0,2 \\ \ \ 0,1428 \end{array}_1 .$$

$$0,0 \qquad 0,01 \qquad 0,010 \qquad 0,0094$$

Для установления факта, какое число больше (меньше), достаточно воспользоваться вторым столбцом, где $\frac{A_2}{P} = 0,01$, а $\frac{A_1}{P} = 0,00$, так как $\frac{A_2}{P} > \frac{A_1}{P}$, то $A_2 > A_1$. Первый столбец дает неверный результат при сравнении — не определяет, какое число больше (меньше). Третий и четвертый столбцы определяют верно, какое из сравниваемых чисел больше, но в них третий и четвертый десятичные знаки после запятой являются избыточными, так как нас не интересует вопрос о том, насколько одно из чисел больше или меньше другого. Итак, для сравнения чисел в условиях примера достаточно иметь два десятичных разряда после запятой.

Известно, что абсолютная погрешность суммы равна сумме граничных абсолютных погрешностей слагаемых. Это утверждение дает возможность оценить точность получаемой суммы и предвидеть, с какой точностью надо взять слагаемые для того, чтобы гарантировать необходимую точность результата. При большом числе слагаемых вычисле-

ния лучше вести с двумя запасными десятичными знаками, а после сложения результат округлить согласно точности числа, имеющего наибольшую абсолютную погрешность.

Двоичные дроби

Аналогично десятичным дробям проведем вычисления для двоичных дробей. Для возможности округления возьмем два периода, тогда:

$$\frac{A_1}{P} \approx |1 \cdot 0{,}1|_1 + |1 \cdot 0{,}0101|_1 + \\ + |1 \cdot 0{,}10111011|_1 + |1 \cdot 0{,}10011001|_1 = \\ = 0{,}1 + 0{,}0101 + 0{,}10111011 + 0{,}10011001.$$

Далее найдем сумму приближенных чисел, взяв по 4 и 5 знаков после запятой. Очевидно, что во втором случае вычисления будут более точными:

$$\begin{array}{|l} 0{,}1 \\ +\ 0{,}01 \\ \quad 0{,}1011 \\ \quad 0{,}1001 \end{array}_1 \quad \begin{array}{|l} 0{,}1 \\ +\ 0{,}01 \\ \quad 0{,}10111 \\ \quad 0{,}10011 \end{array}_1 .$$

$$0{,}0000 \qquad 0{,}00010$$

Далее найдем суммы приближенных значений $\frac{A_2}{P}$ аналогично вычислениям для числа $\frac{A_1}{P}$:

$$\begin{array}{|l} 0{,}10 \\ +\ 0{,}0111 \\ \quad 0{,}0011 \end{array}_1 \quad \begin{array}{|l} 0{,}10 \\ +\ 0{,}01111 \\ \quad 0{,}00110 \end{array}_1 .$$

$$0{,}0000 \qquad 0{,}00101$$

Для установления факта, какое из чисел больше (меньше), воспользуемся вторым способом, где $\frac{A_2}{P} = 0{,}00101$, а $\frac{A_1}{P} = 0{,}00010$, так как $\frac{A_2}{P} > \frac{A_1}{P}$, то $A_2 > A_1$. Первый способ менее точен. Итак, для сравнения чисел достаточно взять один период с дополнительным операционным разрядом второго периода.

В общем случае выражение $\left|\sum_{i=1}^{4} K_i \alpha_i\right|_1$ — это двоичная дробь, заключенная между 0 и 1, полученная в результате суммирования произведения разрядов остаточного кода и двоичных компонент дробей $\frac{\left|P_i^{-1}\right|_{p_i}}{p_i}$. Причем погрешность ε представления двоичной дроби должна быть заключена в пределах $0 \leqslant \varepsilon \leqslant \frac{1}{P} \approx 2^{-r}$, где r — младший разряд дроби $\frac{1}{P}$.

Для оценки качества предлагаемых решений, выраженного в скорости и сложности выполнения операций над данными, представленными в десятичных и двоичных дробях, введем определенные критерии с учетом нежестких допущений, позволяющих сравнить эффективность точных и приближенных методов вычисления позиционных характеристик модулярного кода.

Введем нежесткие допущения:

- динамические диапазоны сравниваемых различными методами чисел равны между собой;
- длительность выполнения операций определяется циклами синхронизации, или в тактах, и линейно зависит от размерности обрабатываемых данных;
- сложность определяется числом разрядов;
- сравнение методов определения позиционных характеристик между собой осуществляется по двум критериям: временным затратам и разрядности обрабатываемых чисел.

Предложенный в данной работе метод приближенного вычисления позиционной характеристики сравним с точным методом определения позиционной характеристики. В качестве точного метода возьмем метод с использованием коэффициентов обобщенной позиционной системы счисления [3], который является наилучшим среди известных методов. В таблице приведены временные затраты и разрядности обрабатываемых чисел для конкретного случая.

Анализируя данные, приведенные в таблице, можно отметить, что при использовании разработанного приближенного метода требуются два разряда для операций с десятичными числами и пять разрядов для операций с двоичными числами. Отсюда можно сделать вывод, что представление и обработка данных двоичными разрядами более эффективнее по сравнению с десятичным представлением.

При сравнении приближенного метода определения позиционной характеристики на основе от-

Временные затраты и разрядности обрабатываемых чисел

Модули СОК	Диапазон СОК	Метод	Десятичное представление		Двоичное представление	
			Число разрядов	Число тактов	Число разрядов	Число тактов
$p_1 = 2$, $p_2 = 3$, $p_3 = 5$, $p_4 = 7$	$P = 210$	ОПСС	4	9	9	8
		Приближенный	2	6	5	4

носительных величин с точным методом на основе обобщенной позиционной системы счисления, который считается наиболее быстродействующим, можно отметить высокую эффективность приближенного метода. Так, для приведенного примера при десятичной арифметике разрядность уменьшается в 2 раза, скорость вычисления — в 1,5 раза, а при двоичных вычислениях разрядность уменьшается в 1,8 раз, а скорость повышается в 2 раза.

Заключение

Разработанный и исследованный эффективный метод приближенного вычисления позиционной характеристики на основе использования относительных величин, представленных в виде периодических дробей, обладает привлекательными свойствами: высокая скорость, малые аппаратурные затраты и универсальность. Основными немодульными операциями, которые допускают применение приближенного метода, являются обнаружение и локализация ошибки числа, а также операции, в которых достаточно знать интервалы расположения числа (определение знака, сравнение чисел и переполнение динамического диапазона). Кроме того, приближенный метод успешно может быть использован в сочетании с точными методами для реализации некоторых других операций (округление, масштабирование и др.). Применение приближенного метода позволяет сократить временные и аппаратурные затраты на выполнение немодульных операций, что открывает хорошие перспективы для дальнейшего развития и применения теории модулярной арифметики на практике.

Список литературы

1. **Акушский И. А., Юдицкий Д. И.** Машинная арифметика в остаточных классах. М.: Советское радио, 1968. 440 с.
2. **Червяков Н. И., Сахнюк П. А., Шапошников А. В., Рядов С. А.** Модулярные параллельные вычислительные структуры нейропроцессорных систем / Под ред. Н. И. Червякова. М.: ФИЗМАТЛИТ, 2003. 288 с.
3. **Червяков Н. И., Сахнюк П. А. Шапошников А. В., Макоха А. Н.** Нейрокомпьютеры в остаточных классах / Под ред. А. И. Галушкина. М.: Радиотехника, 2003. 272 с.
4. **Червяков Н. И.** Методы, алгоритмы и техническая реализация основных проблемных операций, выполняемых в системе остаточных классов // Инфокоммуникационные технологии. 2011. № 4. С. 4—12.
5. **Червяков Н. И., Бабенко М. Г., Ляхов П. А., Лавриненко И. Н.** Приближенный метод ускоренного обнаружения и локализации неисправного вычислительного канала ЭВМ, функционирующей в системе остаточных классов // Нейрокомпьютеры: разработка, применение. 2011. № 10. С. 13—19.
6. **Амербаев В. М.** Теоретические основы машинной арифметики. Алма-Ата: Наука, 1976. 324 с.

Поздравляем юбиляра!

Заслуженному деятелю науки и техники РФ,
доктору технических наук,
профессору Московского государственного
технологического университета «Станкин»
и Российского государственного университета
инновационных технологий и предпринимательства

Виктору Владимировичу ПАВЛОВУ,

известному ученому, крупному специалисту в области современных информационных технологий в машиностроении, члену редколлегии журнала «Информационные технологии» исполнилось 80 лет.

Талантливый ученый и преподаватель, Виктор Владимирович является автором нового научного направления в области САПР, связанного с математическим моделированием систем автоматизации технологического проектирования изделий машиностроения на основе аппарата полихроматических множеств и графов.

Высокий профессионализм, широкая эрудиция, большая трудоспособность, чуткость и отзывчивость снискали ему большое уважение учеников и коллег.

Дорогой Виктор Владимирович!

Сердечно поздравляем Вас с юбилеем и желаем Вам крепкого здоровья и новых творческих успехов в Вашей научной и педагогической деятельности!

Редколлегия и редакция журнала.

УДК 004.3.06

Е. В. Ворожцов,
д-р физ.-мат. наук, вед. науч. сотр.,
Институт теоретической и прикладной механики
СО РАН, г. Новосибирск,
e-mail: vorozh@itam.nsc.ru

Реализация интерфейса между Фортран-программами и графической системой комплекса Mathematica

Кратко описываются возможности программного комплекса Mathematica при его применении для выполнения аналитических вычислений и численных расчетов. Приводятся примеры использования различных графических функций системы Mathematica для визуализации аналитических или численных решений многомерных задач механики сплошной среды. На примерах расчетов двумерных задач газовой динамики показано, что программа, написанная на языке системы Mathematica, считает в тысячу раз медленнее, чем Фортран-программа. В связи с этим предлагается простая реализация интерфейса между Фортран-программой и системой Mathematica, которая позволяет в полной мере реализовать те обширные возможности, которые предоставляет широкий спектр функции компьютерной графики системы Mathematica. Приводятся примеры программной реализации интерфейса на языке Фортран-90.

Ключевые слова: *система Mathematica, Фортран, компьютерная графика, газовая динамика*

Введение

Стивен Уолфрэм (Stephen Wolfram) начал разработку программного комплекса *Mathematica* в 1986 г. [1]. Первая коммерческая версия этого комплекса, *Mathematica 1*, была выпущена в 1988 г. *Mathematica* была задумана как система, автоматизирующая труд научных работников и математиков. Начиная с версии 2, вышедшей в 1991 г., эта и следующие версии системы *Mathematica* являются мировыми лидерами среди компьютерных систем символьной математики для персональных компьютеров (ПК), обеспечивающими не только возможности выполнения численных расчетов с выводом их результатов в самом изысканном графическом виде, но и проведение особо трудоемких аналитических вычислений и преобразований. В настоящее время система *Mathematica* оптимизирована под новейшие операционные системы и аппаратное обеспечение, поэтому пригодна любая система, нужная пользователю (см. подробнее на сайте www.wolfram.com/mathematica/features/system-requirements.html). *Mathemalica 8* имеется в наличии на следующих платформах: Microsoft Windows, Apple Mac, Linux. Обширный список книг, посвященных либо описанию элементов системы, либо ее разнообразным применениям, представлен на сайте http://library.wolfram.com/infocenter/Books/. В настоящее время система *Mathematica* является одним из самых мощных средств поддержки научных исследований, прикладного проектирования и обучения специалистов, аспирантов и студентов. Комплекс *Mathematica* содержит широкий спектр математических методов, вычислительных алгоритмов и компьютерных программ для проведения теоретических и прикладных исследований. Комплекс снабжен графической системой визуализации полученных решений. Интерфейсы графической и вычислительной систем имеют ряд проблем и тонкостей при их практическом применении. Настоящая работа посвящена описанию этих важных проблем и предлагает решение некоторых из них.

1. Общее описание комплекса *Mathematica*

Вопросы инсталляции системы *Mathematica*, описание пользовательского интерфейса, работа с ячейками освещены достаточно подробно в работе [2]. Ниже в данном разделе будут даны примеры реализации символьных вычислений и численных расчетов в системе *Mathematica*.

В системе *Mathematica* можно, в отличие от языка численных расчетов Фортран, вместо оператора присваивания $a = b$, где b — некоторое выражение, вводить просто b. Далее имена всех встроенных функций системы *Mathematica* начинаются с заглавной буквы. Например, функция Log[x] вычисляет натуральный логарифм lnx переменной x, Sqrt[x] — квадратный корень из x и т. д.

1.1. Символьные вычисления

Ниже приводятся сравнительно простые примеры, иллюстрирующие возможности системы *Mathematica* при ее применении для символьных (аналитических) вычислений на ПК.

Пример 1. Вычислить бином Ньютона $(a-3b)^{20}$.

Решение с помощью системы *Mathematica*:

```
Expand[(a - 3b)^20]//TraditionalForm
```

Опция `TraditionalForm` заставляет систему *Mathematica* выдавать результат в привычной математической форме:

$$a^{20} - 60ba^{19} + 1710b^2a^{18} - 30780b^3a^{17} + \\ + 392445b^4a^{16} - 3767472b^5a^{15} + 28256040b^6a^{14} - \\ - 169536240b^7a^{13} + 826489170b^8a^{12} - \\ - 3305956680b^9a^{11} + 10909657044b^{10}a^{10} - \\ - 29753610120b^{11}a^9 + 66945622770b^{12}a^8 - \\ - 123591918960b^{13}a^7 + 185387878440b^{14}a^6 - \\ - 222465454128b^{15}a^5 + 208561363245b^{16}a^4 - \\ - 147219785820b^{17}a^3 + 73609892910b^{18}a^2 - \\ - 23245229340b^{19}a + 3486784401b^{20}.$$

Пример 2. Упростить выражение, содержащее гиперболические функции:

$$\frac{\mathrm{ch}(y)\mathrm{sh}(x)}{\mathrm{ch}(x)\mathrm{ch}(y)+\mathrm{sh}(x)\mathrm{sh}(y)} + \frac{\mathrm{ch}(x)\mathrm{sh}(y)}{\mathrm{ch}(x)\mathrm{ch}(y)+\mathrm{sh}(x)\mathrm{sh}(y)}.$$

Решение с помощью системы *Mathematica*:

```
Trigreduce[Cosh[y]*Sinh[x]/
(Cosh[x]Cosh[y]   +   Sinh[x]*Sinh[y])   +
Cosh[x]*Sinh[y]/(Cosh[x]Cosh[y]
+Sinh[x]*Sinh[y])] //TraditionalForm
```

Ответ: $\tanh(x+y)$.

Пример 3. Вычислить в аналитическом виде неопределенный интеграл

$$\int \frac{\ln(1+x)}{\sqrt{x}}\,dx.$$

Решение с помощью системы *Mathematica*:

```
Integrate[Log[1 + x]/Sqrt[x], x]
```

Ответ: $4\mathrm{ArcTan}[\sqrt{x}\,] + 2\sqrt{x}\,](-2 + \mathrm{Log}[1+x])$.

Здесь не применялась опция `TraditionalForm` для того, чтобы показать, что результат, полученный с помощью системы *Mathematica*, можно тут же вводить в ту же или в другую *Mathematica*-программу и использовать в дальнейших символьных вычислениях. Для этого нужно скопировать результат в буфер (копирование выражения или фрагмента текста выполняется в текстовом редакторе Word). Затем его ввести в нужном месте *Mathematica*-программы с помощью комбинации клавиш CTRL V. Выражение, обрабатываемое в следующем примере, было введено в *Mathematica*-программу именно таким способом.

Пример 4. Дифференцирование результата, полученного в примере 3, должно дать подынтегральную функцию. Вот как дифференцирование реализуется в системе *Mathematica*:

```
Simplify[D[4ArcTan [√x ] + 2√x ]
(-2 + Log[1 + x]), x]]
```

Ответ: $\dfrac{\log(1+x)}{\sqrt{x}}$.

1.2. Численные расчеты в системе Mathematica

В системе *Mathematica* можно также выполнять численные расчеты. При этом точность вычислений в арифметике машинных чисел с плавающей запятой, если применять точность, заданную в системе по умолчанию, примерно такая же, как при использовании двойной точности в Фортран-программах. Но, в отличие от Фортрана, в системе *Mathematica* можно задавать точность численных вычислений с помощью встроенной функции `N[expr, m]`, где `expr` — выражение, числовое значение которого необходимо вычислить, а `m - 1` — задаваемое пользователем число правильных цифр мантиссы числового результата.

Пример 5. Вычислить число π с точностью 39 правильных цифр мантиссы. Применение функции `N[ Pi, 40]` дает следующий результат:

3.14159265358979323846264338327950288419 7.

В работе [3] число π задано только с 24 цифрами мантиссы: 3.14159265358979323846264 3. Из сравнения табличного результата с результатом, полученным выше с помощью системы *Mathematica*, видно, что 24 цифры мантиссы совпадают в обоих случаях. Конечно, с увеличением машинной точности `m` требуется и большее машинное время для численного счета.

Пример 6. Найти с помощью системы *Mathematica* числовое значение выражения 2/3 + 4/7. Система дает ответ 26/21, который является точным аналитическим результатом. Однако в приложениях часто требуется результат в форме машинного числа с плавающей запятой. Это можно обеспечить в системе *Mathematica* одним из трех способов [1]:

```
N[2/3 + 4/7]; 2/3 + 4/7//N; 2./3 + 4/7.
```

Во всех трех случаях ответ будет 1.2381. Достаточно объявить одно из целых чисел (в данном примере число 2) машинным числом с плавающей запятой, и результат будет выдан как машинное число с плавающей запятой.

2. Описание графической системы

Графические возможности системы *Mathematica* достигаются как обилием встроенных функций компьютерной графики, так и средствами их модификации с помощью директив, опций и примитивов.

Задачи механики сплошных сред и математической физики, которые описываются системами уравнений в частных производных и зависят от двух или трех пространственных переменных и времени t, принадлежат к классу вычислительно

трудоемких задач. Особенности применения графических функций системы *Mathematica* для визуализации решений указанных задач описаны в работах [4, 5]. Следует, однако, заметить, что в [4, 5] не рассматривались вопросы организации интерфейса между Фортран-программами и системой *Mathematica*. Это связано с тем, что в [4, 5] были приведены примеры численного решения только тех сравнительно простых одномерных, двумерных и трехмерных задач механики сплошных сред, которые могут быть сосчитаны с приемлемым расходом машинного времени в рамках системы *Mathematica* при достаточно умеренном общем числе узлов пространственной расчетной сетки. Попытки непосредственного применения системы *Mathematica* для численного решения более сложных задач наталкиваются на серьезные технические трудности, которые будут обсуждаться подробнее в следующем разделе. А в данном разделе на нескольких примерах будет показано, как можно представлять в различных графических формах результаты численного или аналитического решения двумерных и трехмерных задач механики сплошных сред и математической физики с помощью встроенных функций компьютерной графики системы *Mathematica*.

Задача о безвихревом обтекании неподвижной сферы потоком идеальной несжимаемой жидкости является сравнительно простым примером трехмерной задачи гидродинамики. Предполагается, что на бесконечности поток жидкости направлен вдоль оси $\bar{z}$ декартовой системы координат $O\bar{x}\,\bar{y}\,\bar{z}$. Решение рассматриваемой задачи зависит от трех размерных координат $\bar{x}$, $\bar{y}$, $\bar{z}$. Пусть $\bar{u}$, $\bar{v}$, $\bar{w}$ — размерные составляющие вектора скорости жидкости вдоль осей, соответственно, $\bar{x}$, $\bar{y}$, $\bar{z}$, и пусть $\bar{p}$ — размерное давление жидкости. Обезразмеривание независимых и зависимых переменных в рассматриваемой задаче осуществляется по формулам

$$x = \frac{\bar{x}}{R};\ y = \frac{\bar{y}}{R};\ z = \frac{\bar{z}}{R};$$

$$u = \frac{\bar{u}}{U_\infty};\ v = \frac{\bar{v}}{U_\infty};\ w = \frac{\bar{w}}{U_\infty};\ p = \frac{\bar{p}}{\rho U_\infty^2},$$

где R — размерный радиус сферы; U_∞ — скорость невозмущенного потока; ρ — плотность жидкости. Для определенности предполагается, что центр сферы находится в начале координат (0, 0, 0) системы координат $Oxyz$. Тогда точное решение рассматриваемой задачи имеет вид [6]

$$u = -\frac{3}{2}\frac{xz}{r^5};\ v = -\frac{3}{2}\frac{yz}{r^5};\ w = -\frac{3}{2}\frac{z^2}{r^5} + 1 + \frac{1}{2r^3};$$

$$p = \frac{p_\infty}{\rho U_\infty^2} + \frac{1}{2}(1 - u^2 - v^2 - w^2), \qquad (1)$$

где $r = (x^2 + y^2 + z^2)^{1/2}$.

Ниже в данном разделе на примере решения (1) иллюстрируются возможности функций компьютерной графики системы *Mathematica*. Демонстрация применения графических функций системы *Mathematica* осуществляется в некоторых сечениях в целях изучения поведения решения рассматриваемой задачи. Варьируя число этих сечений, можно свести рассмотрение либо к одномерным графикам вида $u = u(x, y_0, z_0)$, либо к поверхностям вида $u = u(x, y_0, z)$.

Как видно из формул (1), с их использованием можно визуализировать поведение решения и внутри сферы. Предполагается, что сфера твердая и непроницаемая для жидкости (т. е. не содержит пор). Для того чтобы на графике идентифицировать подобласть, занятую сферой, можно модифицировать формулы (1) следующим образом:

$$u = -\frac{3}{2}f(r)\frac{xz}{r^5};\ v = -\frac{3}{2}f(r)\frac{yz}{r^5};$$

$$w = f(r)\left(-\frac{3}{2}\frac{z^2}{r^5} + 1 + \frac{1}{2r^3}\right); \qquad (2)$$

$$p = f(r)\left[\frac{p_\infty}{\rho U_\infty^2} + \frac{1}{2}(1 - u^2 - v^2 - w^2)\right],$$

где $f(r) = (1/2)[1 + \mathrm{sgn}(r^2 - 1)]$. Тогда $u = v = w = p = 0$ внутри сферы. Это удобно для идентификации подобластей течения, где, например, u имеет различные знаки. Кроме того, равенства $u = v = w = 0$ отражают тот факт, что сфера неподвижна.

Из формул (2) следует, что безразмерное давление в потоке, обтекающем сферу, можно представить в виде суммы $p = f(r)p_\infty/\rho U_\infty^2 + \Delta p$, где Δp — приращение давления, вызванное наличием сферы в потоке; $\Delta p = (1/2)f(r)(1 - u^2 - v^2 - w^2)$.

На рис. 1 (см. четвертую сторону обложки) показаны поверхности $u = u(x, 0, z)$, $w = w(x, 0, z)$, $\Delta p = \Delta p(x, 0, z)$. Ниже приводится программа на языке системы *Mathematica* для отрисовки поверхности $u = u(x, 0, z)$:

```
r = Sqrt[x^2 + y^2 + z^2];
u = -(3/2)*x*z/r^5;
uy0 = 0.5*(1 + Sign[x^2 + y^2 + z^2 - 1])*
u/.y -> 0;
Plot3D[uy0, {x, -2, 2}, {z, -3, 3},
PlotRange -> All, DefaultFont-> "Times",
AspectRatio-> Automatic,
PlotPoints-> {30,45},
BoxRatios-> {1,1.5,1}, AxesLabel->
{Style-Form["x", FontFamily-> "Times",
FontSlant -> "Oblique", FontSize-> 10,
FontWeight-> "Plain"], StyleForm["z", Font-
Family-> "Times", FontSlant -> "Oblique",
```

```
FontSize-> 10, FontWeight-> "Plain"],
Style-Form["u",FontFamily -> "Times",
FontSlant -> "Oblique", FontSize-> 10,
FontWeight->"Plain"]}];
```

Mathematica-команды для отрисовки w и Δp в сечении $y = 0$ аналогичны, поэтому здесь не приводятся в целях краткости. Кроме того, здесь и ниже опущено описание используемых опций в аргументах графических функций; эти описания имеются в работах [1, 2, 4, 5, 7].

На рис. 2 показаны одномерные профили решения в сечении $y = 0$, $z = 0{,}5$. Они были получены с помощью следующих *Mathematica*-команд:

```
w = -(3/2)*z*z/r^5 + 1 + 1/(2r^3);
wy0 = 0.5*(1 + Sign[x^2 + y^2 + z^2 - 1])
* w/.y-> 0; uy0z0 = uy0/.z->.5;
wy0z0 = wy0/.z->0.5;
gruld = Plot[{uy0z0, wy0z0}, {x, -2, 2},
PlotStyle->   {Thickness[.01],   {Dashing
[{0.03, 0.03}], Thickness [0.015]}},
DefaultFont-> "Times", AxesLabel->
{StyleForm["x", FontFamily-> "Times",
FontSlant-> "Oblique", FontSize-> 10,
FontWeight-> "Plain"], None }];
```

С помощью системы *Mathematica* можно также строить изолинии (рис. 3). Рис. 3, *а* был получен с помощью следующих *Mathematica*-команд:

```
g4 = ContourPlot[uy0, {x, -2,2}, {z, -3,3},
AspectRatio-> Automatic, Plot-Points-> 60,
Contours-> 20, ContourShading-> False,
ContourStyle-> Thickness[0.015],
DefaultFont-> {"Times", 10},
DisplayFunction-> Identity];
sfera=Polygon[Table[{Cos[n Pi/40],
Sin[n Pi/40]}, {n, 80}]];
Show[g4,  Graphics[{GrayLevel[0.6],  sfera}],
DisplayFunction-> $DisplayFunction];
```

С помощью той же функции `ContourPlot` [...] можно не только отрисовывать изолинии, но и закрашивать определенными цветами подобласти, в которых значение функции мало отличается от заданного значения (рис. 4, см. четвертую сторону обложки). Для этого нужно заменить в обращении к этой функции опцию `ContourShading-> False` на опцию `ColorFunction-> Hue`.

На рис. 5 показано поле векторов скорости жидкости в сечении $y = 0$. Локальная длина каждого вектора пропорциональна локальной скорости. Видно, что длины векторов скорости наименьшие на линии $x = 0$ вблизи сферы. Как известно, в точках с координатами $x = 0$, $y = 0$, $z = \pm 1$ имеет место торможение потока, см. также формулы (1). Рис. 5 был получен с помощью следующих *Mathematica*-команд:

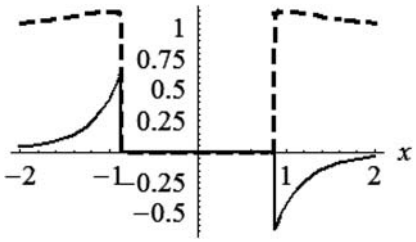


Рис. 2. Профили решения в сечении $y = 0$, $z = 0{,}5$:
$u(x, 0, 0.5)$ (сплошная линия), $w(x, 0, 0.5)$ (штриховая линия)

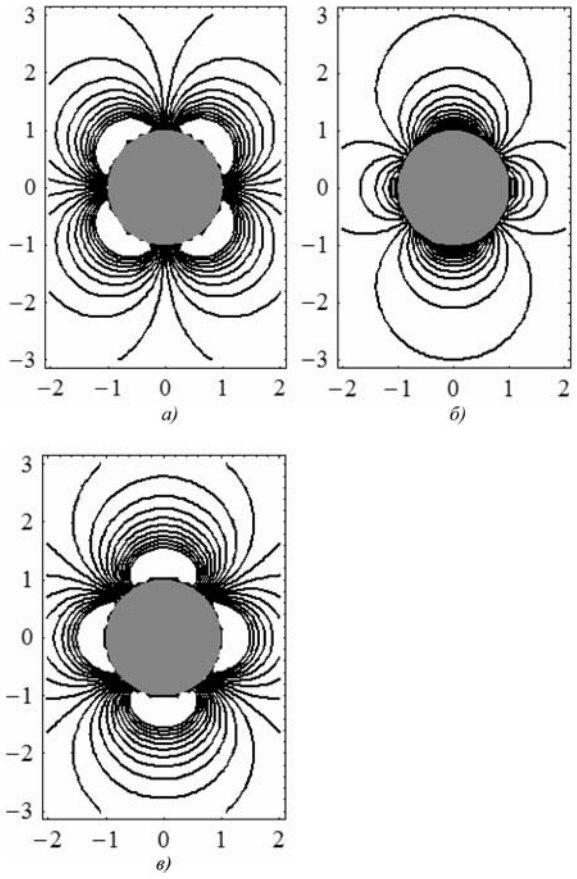


Рис. 3. Изолинии компонент решения в сечении $y = 0$:
а — изолинии функции $u(x, 0, z)$; *б* — изолинии функции $w(x, 0, z)$; *в* — изолинии функции $\Delta p(x, 0, z)$

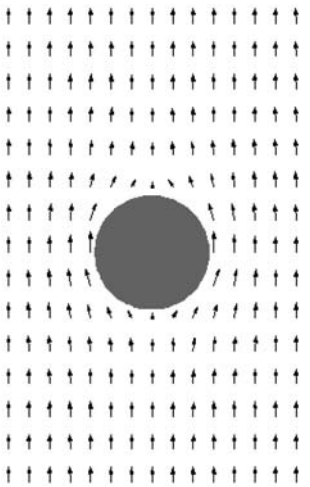


Рис. 5. Поле векторов скорости в сечении $y = 0$

```
«Graphics 'PlotField';
g4 = PlotVectorField[{uy0, wy0},
{x, -2.5, 2.5}, {z, -4,4},
DisplayFunction-> Identity];
Show[g4, Graphics[{GrayLevel[0.6] sfera}],
DisplayFunction-> $DisplayFunction];
```

Зависимость вида $u = u(x, y, z)$ описывает гиперповерхность в четырехмерном евклидовом пространстве точек (x, y, z, u). В системе *Mathematica* имеется встроенная функция `ContourPlot3D[...]`, позволяющая визуализировать трехмерные сечения $u(x, y, z) = \text{const}$ гиперповерхности $u = u\{x, y, z)$. Ниже приводятся примеры таких трехмерных сечений. Из них видно, что изображения таких сечений трудно интерпретировать и анализировать в случае рассматриваемой задачи гидродинамики. На рис. 6 (см. четвертую сторону обложки) показаны половины изоповерхностей $u(x, y, z) = u_0$ при $u_0 = -0{,}1$, $u_0 = -0{,}5$. Эти изоповерхности показаны не полностью для того, чтобы можно было видеть взаимное расположение различных сечений гиперповерхности $u = u(x, y, z)$. Рис. 6 был получен с помощью следующих *Mathematica*-команд:

```
«Graphics 'ContourPlot3D'
gr1 = ContourPlot3D[u, {x, 0.001, 2},
{y, 0.001, 3}, {z, 0.001, 2},
Contours-> {-.1, -.5},
PlotPoints-> {6, 8},
DisplayFunction-> Identity];
gr4 = ContourPlot3D[u, {x, -2, -0.001},
{y, 0.001, 3}, {z, -2, -0.001},
Contours->{-.1, -.5}, PlotPoints->{6, 8},
DisplayFunction-> Identity];
Show[gr1, gr4, Axes-> True,
DefaultFont -> "Times",
DisplayFunction->$DisplayFunction];
```

На аналогичном рис. 7 (см. четвертую сторону обложки) показаны три четверти изоповерхностей $w(x, y, z) = w_0$ при $w_0 = 0{,}1$, $w_0 = 0{,}8$.

3. Интерфейс вычислительной и графической систем

Задачи динамики невязкого сжимаемого газа, решения которых зависят от двух или трех пространственных переменных, относятся к классу вычислительно трудоемких задач. При численном решении этих задач в настоящее время является обычным использование пространственных сеток, содержащих несколько миллионов ячеек. Уравнения Эйлера, являющиеся основной математической моделью для этих задач, имеют следующий вид для двумерного течения невязкого нетеплопроводного сжимаемого газа:

$$\frac{\partial w}{\partial t} + \frac{\partial f(w)}{\partial x} + \frac{\partial g(w)}{\partial y} = 0, \qquad (3)$$

где x, y — декартовы пространственные координаты, t — время, и

$$w = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}; f(w) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix}; g(w) = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vH \end{pmatrix}. \qquad (4)$$

Здесь p, ρ, u, v обозначают давление, плотность, составляющие скорости; $E = \varepsilon + 1/2(u^2 + v^2)$, ε — удельная внутренняя энергия; $H = E + \frac{p}{\rho}$. Используется уравнение состояния идеального газа $p = (\gamma - 1)\rho\varepsilon$, $\gamma = \text{const} > 1$. В работе [8] (см. также [9]) был подробно описан явный TVD-метод применительно к уравнениям (3), (4), поэтому здесь не приводятся расчетные формулы этого метода, который, в зависимости от числового значения присутствующего в нем параметра ϕ, в подобластях гладкого течения имеет либо второй, либо третий порядок точности по пространственным переменным.

Применяются ограничители потоков, а также осреднение по Роу. В этом методе, как и в других TVD-методах, проводится переключение с высокого порядка аппроксимации на первый в областях ударных волн и контактных разрывов в целях получения монотонных профилей численного решения.

Задача об отражении косой ударной волны от стенки часто используется для тестирования новых численных методов решения уравнений (3), (4). Точное решение этой тестовой задачи представляет собой кусочно-постоянную функцию и находится численно с машинной точностью с помощью теории косых ударных волн [5]. В приводимых ниже примерах расчетов вводилось значение $\varphi = \pi/6$ для угла между фронтом падающей ударной волны и осью x (рис. 8). В случае совершенного газа (воздуха) с $\gamma = 1{,}4$ постоянные точного решения в подобластях 1, 2, 3, указанных на рис. 8, таковы:

подобласть 1: $u_1 = 1.0$, $v_1 = 0.0$, $p_1 = 0.084932903$, $\rho_1 = 1.0$, $M_1 = 2.90$;

подобласть 2: $u_2 = 0.890755053$, $v_2 = 0.189217798$, $p_2 = 0.194177850$, $\rho_2 = 1.776135164$, $M_2 = 2.327642861$.

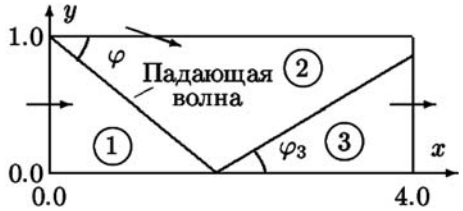


Рис. 8. Пространственная область в задаче об отражении косой ударной волны

подобласть 3: $u_3 = 0.806645743$, $v_3 = 0.0$, $p_3 = 0.390838939$, $\rho_3 = 2.898621574$, $M_3 = 1.856588584$.

Здесь M_1, M_2, M_3 — значения числа Маха $M = \sqrt{u^2 + v^2}/c$ в подобластях 1, 2, 3, соответственно; $c = \sqrt{\gamma p/\rho}$ — скорость звука. Отраженная ударная волна образует угол $\varphi_3 = 0.418279545$ с положительным направлением оси x.

Численное решение находилось методом установления. В начальный момент времени все поле течения задавалось равным значениям невозмущенного сверхзвукового втекающего потока, заданным выше для подобласти 1, т. е. начальное течение газа параллельно оси x.

Критерий сходимости разностного решения w^n к предельному стационарному решению брался в виде $\mathrm{Res}(n) < \varepsilon$, где ε — задаваемое пользователем малое положительное число;

$$\mathrm{Res}(n) = \max_{j,\,k}\{\max(|R^n_{1,j,k}|, |R^n_{2,j,k}|, |R^n_{3,j,k}|, |R^n_{4,j,k}|)\}, \quad (5)$$

где

$$\{R^n_{1,j,k},\ R^n_{2,j,k},\ R^n_{3,j,k},\ R^n_{4,j,k}\}^{\mathrm{T}} = \frac{\hat{f}^n_{j+1/2,k} - \hat{f}^n_{j-1/2,k}}{h_1} + \frac{\hat{g}^n_{j,k+1/2} - \hat{g}^n_{j,k-1/2}}{h_2}$$

в случае метода [8]; n — номер временного слоя $\{n = 0, 1, 2, ...\}$; $\hat{f}^n_{j\pm1/2,k}$, $\hat{g}^n_{j,k\pm1/2}$ — разностные аппроксимации потоков $f(w)$ и $g(w)$ в формуле (3) по методу [8], верхний индекс т обозначает операцию транспонирования; h_1, h_2 — шаги равномерной прямоугольной расчетной сетки в плоскости (x, y).

На рис. 9 приведены результаты численного расчета задачи об отражении косой ударной волны от твердой стенки, полученные счетом на установление по методу [8] на равномерной сетке из 160 × 40 ячеек; параметр метода $\phi = 1/3$, что обеспечивает его третий порядок точности по пространственным переменным в подобластях гладкого течения. Число изолиний $M = \mathrm{const}$ равно 16. На рис. 9, *б* показана невязка (5) в случае метода [8]. Видно, что требуется 2400 шагов во времени для обеспечения падения нормы невязки решения до уровня машинных ошибок округления $10^{-10}...10^{-12}$. На рис. 9, *в* сплошная линия — точное решение, штриховая линия — результат расчета по методу [8].

Для численного решения данной задачи были созданы две программы: одна на языке системы *Mathematica* и другая — на языке Фортран-90 (транслятор Compaq Visual Fortran, version 6.6), см. также [10]. Результаты расчетов по обеим программам совпали. В обеих программах был реализован способ оптимизации программирования разностной схемы, согласно которому поток $\hat{f}^n_{j+1/2,k}$ через правую границу $j + 1/2$ ячейки (j, k) равен потоку $\hat{f}^n_{j+1/2,k}$ через левую границу $j + 1/2$ ячейки $(j + 1, k)$. Это позволяет сократить потребное машинное время в 2 раза при численном решении двумерных и трехмерных задач газовой динамики по сравнению со способом программирования, в котором указанные потоки вычисляются независимо для каждой ячейки [4].

Все расчеты были выполнены на ПК с процессором фирмы Intel, тактовая частота 3 ГГц. Для выполнения 2600 временных шагов по *Mathematica*-программе потребовалось 22 457 с, или более 6 ч машинного времени. Фортран-программа выполнила этот же расчет за 21,45 с. Таким образом, сис-

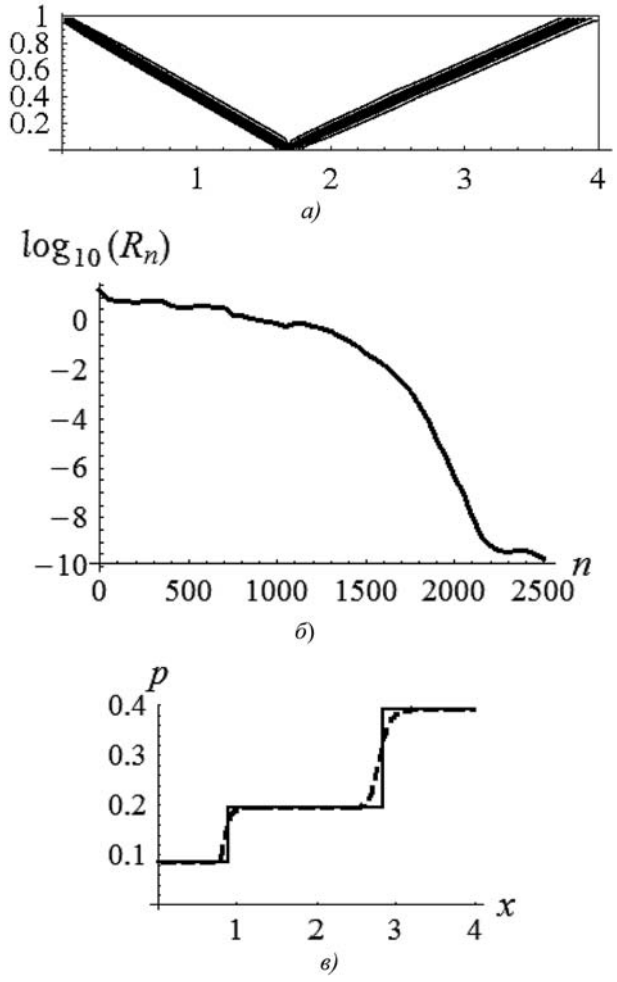


Рис. 9. Задача об отражении косой ударной волны от твердой стенки:
а — линии постоянных чисел Маха; *б* — невязка (5) как функция числа временных шагов n; *в* — давление газа в сечении $y = 0.5$

тема *Mathematica* считает эту двумерную задачу в 1047 раз медленнее, чем Фортран-программа. Были также проведены расчеты этой же задачи по обеим программам на более грубой сетке из 80 × 20 ячеек, было сосчитано 1600 шагов по *t*. В этом случае *Mathematica*-программа выполнила расчет за 3455 с, а Фортран-программа — за 3,36 с. Соответствующее отношение 3455/3,36 = 1028,28. Расхождение в величинах замедления счета 1047 и 1028 объясняется тем хорошо известным фактом, что чем больше промежуточных выдач делается при счете в системе *Mathematica*, тем медленнее она считает. В обоих вариантах счета — на сетках 160 × 40 и 80 × 20 — информация о невязке решения выдавалась через каждые 50 шагов по *t*. На сетке 160 × 40 было сделано 2600 шагов, а на сетке 80 × 20 — гораздо меньшее число шагов по *t*.

С небольшой погрешностью можно полагать, что двумерные задачи газовой динамики считаются в системе *Mathematica* в 1000 раз медленнее, чем на Фортране. Такое неблагоприятное соотношение расходов машинного времени делает систему *Mathematica* неприемлемой для численного решения сложных многомерных задач аэрогидродинамики. С учетом хорошо развитой и высококачественной компьютерной графики этой системы представляется целесообразным организовать достаточно простой в реализации интерфейс между Фортран-программой и системой *Mathematica*. Идея этого интерфейса состоит в том, чтобы по окончании основного счета в Фортран-программе в этой же программе организовать вывод результатов счета во внешний файл или в несколько внешних файлов в формате системы *Mathematica*. А на языке системы *Mathematica* пишется небольшая программа, обеспечивающая графическую визуализацию результатов счета в форме, нужной исследователю. Выход из Фортран-компилятора и вход в *Mathematica*-программу требуют 3...4 с времени пользователя, затем *Mathematica*-программа запускается на счет нажатием клавиш Shift Enter. Она считывает массивы числовых результатов из памяти компьютера, и через несколько секунд пользователь уже видит на экране монитора результаты расчета в графическом виде.

В качестве примера ниже приводится листинг Фортран-программы, в которой одномерный массив uld и думерный массив u2d записываются во внешний файл exmpl.res в формате системы *Mathematica*, т. е. в виде одно- и двумерных списков:

```
      PROGRAM example
      IMPLICIT REAL*8 (a - h, o - z)
      DIMENSION u1d(100),
      u2d(100, 100)
      ix = 30; jy = 15; XL = 0.0d0;
      XR = 2.d0; YL = 0.d0; YR = 1.d0
      h1 = (XR - XL)/dble(ix - 1)
      h2 = (YR - YL)/dble(jy - 1);
      pi = 4.0d0*datan(1.0d0)
      do i = 1,ix
      xi = XL + dble(i - 1)*hl;
      uld(i) = dexp(-0.3d0*xi)*
      DCOS(pi*xi)
      do j = 1, jy;
      yj = YL + dble(j - 1)*h2;
      u2d(i, j) = DSIN(pi*xi)*
      DSIN(pi*yj)
      enddo; enddo
      open(1, file = 'exmpl.res');
      write(1, 100) XL, XR, YL, YR
! Запись массива u1d в файл exmpl.res
      write(1, 101)(uld(i),
      i = 1,ix - 1);
      write(1, 102) u1d(ix)
! Запись массива u2d в файл exmpl.res
      do j = 1, jy; write(1, 103)
      write(1, 101)(u2d(i, j),
      i = 1, ix - 1);
      write(1, 104) u2d(ix, j)
      if(j == jy)then
      write(1, 106);
      else; write(1, 105);
      endif; enddo
  100 format('{', 4(f12.6, ', '),
     '{')
  101 format(6(f12.6, ', '))
  102 format(f12.6, '}, {')
  103 format('{')
  104 format(f12.6)
  105 format{'}, ')
  106 format('}}}')
      STOP; END
```

Чтобы не быть привязанными к какой-либо конкретной прикладной задаче, в Фортран-программе произвольно выбрана для табулирования функция одной переменной $\mathrm{uld}(x) = \mathbf{e}^{-0,3x}\cos(\pi x)$, а для получения двумерного списка — функция двух переменных $\mathrm{u2d}(x, y) = \sin(\pi x)\sin(\pi y)$.

Ниже приводится *Mathematica*-программа exmpl.nb, которая сначала считывает внешний файл exmpl.res из директории, задаваемой в явном виде в программе, и затем отрисовывает результаты в том графическом виде, который интересует пользователя:

```
{XL, XR, YL, YR, uld, u2d} = «"D:\\Papers
\\informtech\\fortr\\exmpl.res";
ix = Length[uld]; jy = Length[u2d];
Print["ix = ", ix, "; jy = ", jy];
xx = Table[XL + (i - 1)*(XR - XL)/(ix - 1),
{i, ix}];
ListPlot[MapThread[List, {xx, uld}],
```

```
PlotRange-> All, PlotJoined-> True,
DefaultFont-> "Times", PlotStyle->
{Thickness[0.012]},
AxesLabel-> {StyleForm["x",FontFamily->
"Times", FontSlant-> "Oblique",
FontSize-> 10,FontWeight-> "Plain"],
StyleForm["uld", FontFamily-> "Times",
FontSlant-> "Oblique", FontSize-> 10,
FontWeight-> "Plain"]}];
ListContourPlot[u2d, AspectRatio->
Automatic, Contours-> 20, ContourShading->
False, MeshRange-> {{XL, XR}, {YL, YR}},
ContourStyle-> Thickness[0.01],
DefaultFont-> "Times"];
ListPlot3D[u2d, PlotRange-> All, MeshRange->
{{XL, XR}, {YL, YR}}, DefaultFont-> "Times",
AspectRatio-> Automatic, BoxRatios->
{2, 1, 1}, AxesLabel-> {StyleForm["x", Font-
Family-> "Times", FontSlant-> "Oblique",
FontSize-> 10, FontWeight-> "Plain"],
StyleForm[" y", FontFamily-> "Times",
FontSlant->"Oblique", FontSize-> 10,
FontWeight-> "Plain"],
StyleForm[" u2d", FontFamily->
"Times", FontSlant-> "Oblique",
FontSize-> 10, FontWeight-> "Plain"]}];
```

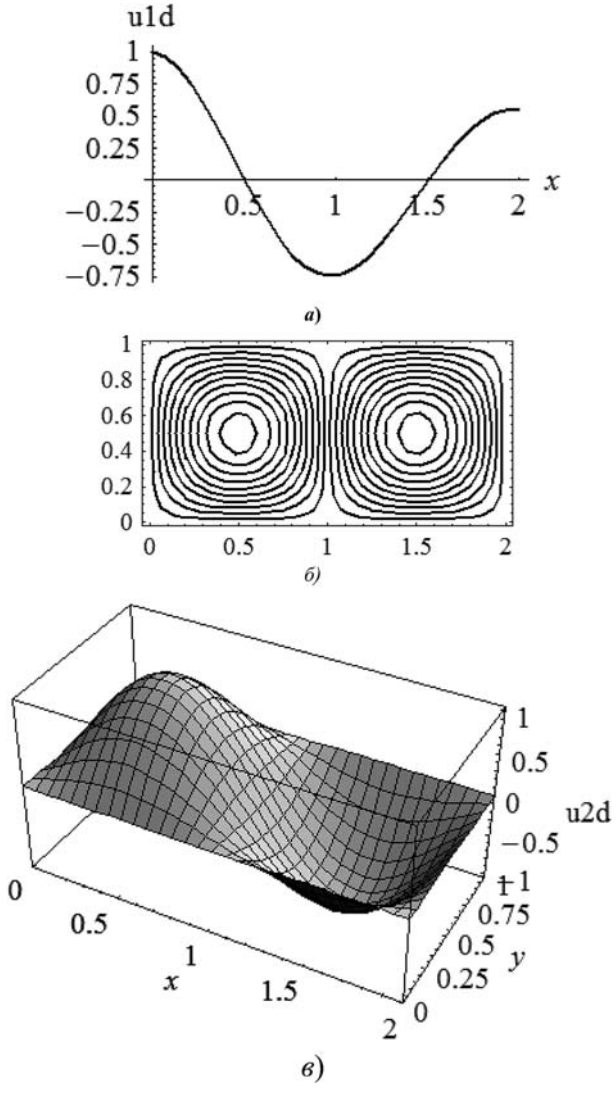


Рис. 10. Графические объекты, построенные с помощью программы exmpl.nb:
а — график функции uld(x); *б* — изолинии функции u2d(x, y); *в* — поверхность u2d = u2d(x, y)

На рис. 10 представлены результаты работы вышеприведенной *Mathematica*-программы.

Заключение

Кратко описаны возможности программного комплекса *Mathematica* при его применении для выполнения аналитических вычислений и численных расчетов. Приводятся примеры использования различных графических функций системы *Mathematica* для визуализации аналитических или численных решений многомерных задач механики сплошной среды. На примерах расчетов двумерных задач газовой динамики показано, что программа, написанная на языке системы *Mathematica*, считает в 1000 раз медленнее, чем Фортран-программа. В связи с этим предлагается простая реализация интерфейса между Фортран-программами и системой *Mathematica*, которая позволяет в полной мере реализовать те обширные возможности, которые предоставляет широкий спектр функций компьютерной графики системы *Mathematica*. Приводятся листинги соответствующих программ на языках Фортран-90 и *Mathematica*, показывающие, как реализовать интерфейс.

Список литературы

1. **Wolfram S.** The Mathematica Book, 3rd Edition. Cambridge, UK: Cambridge University Press, 1996. 1403 p.
2. **Дьяконов В. П.** Системы символьной математики Mathematica 2 и Mathematica 3. М.: СК пресс, 1998. 318 с.
3. **Абрамовиц М., Стиган И.** Справочник по специальным функциям. М.: Наука, 1979. 830 с.
4. **Ganzha V. G., Vorozhtsov E. V.** Numerical Solutions for Partial Differential Equations: Problem Solving Using Mathematica. Boca Raton, New York: CRC Press. 1996. 347 p.
5. **Kiselev S. P., Vorozhtsov E. V., Fomin V. M.** Foundations of Fluid Mechanics with Applications: Problem Solving Using Mathematica, Boston: Birkhauser, 1999. 575 p.
6. **Лойцянский Л. Г.** Механика жидкости и газа. Изд. 10-е. М.: Дрофа, 2003. 840 с.
7. **Wolfram** Research, Mathematica 3.0 Standard Add-on Packages / ed. by E. Martin. Cambridge, UK: Wolfram Media/Cambridge University Press, 1996. 516 p.
8. **Chakravarthy S. R., Osher S.** A new class of high accuracy TVD schemes for hyperbolic conservation laws // AIAA Paper. 1985. N 85-0363. 11 p.
9. **Ворожцов Е. В.** Применение разложений Лагранжа—Бюрмана для численного интегрирования уравнений невязкого газа // Вычисл. методы и программирование. 2011. Т. 12. № 2. С. 112—125.
10. **Горелик А. М.** Программирование на современном Фортране. М.: Финансы и статистика, 2006. 352 с.

УДК 004.42

А. Е. Александров, д-р техн. наук, проф.,
А. А. Востриков, канд. техн. наук, доц.,
И. В. Шалыминов, аспирант,
Московский государственный университет
приборостроения и информатики
e-mail: femsystem@yandex.ru

Программная система "Прогноз" для анализа безопасности энергетических систем

Приведена структурная модель, используемая в программной системе (ПС) "Прогноз", позволяющая строить модель исследуемой конструкции по принципу влияния ее составных элементов на безопасность всей конструкции, а также показана возможность моделирования процессов зарождения и образования трещин для проведения вероятностного анализа безопасности энергетических систем. Представлен алгоритм расчета вероятности образования сквозных трещин, формируемых в результате слияния поврежденных ячеек. Приведен пример проведения анализа безопасности оборудования на основе ПС "Прогноз".

Ключевые слова: *вероятностный анализ безопасности, вероятностные методы механики разрушения, метод конечных элементов*

Введение

При эксплуатации энергетических систем, особенно объектов ядерной энергетики, одной из важнейших задач является задача по обеспечению заданного уровня безопасности. Сложность решения данной задачи связана с учетом большого числа переменных, имеющих, в том числе, и случайный характер. Большое значение при решении такого рода задач приобретают методы построения компактных и содержательных моделей, связанные с декомпозицией исследуемой конструкции на элементы, влияющие на ее безопасность. Построение таких структурных моделей создает методическую основу для реализации вычислительных моделей по прогнозированию уровня безопасности и его обоснованию.

Структурная модель в программной системе "Прогноз"

Структурная модель, используемая в программной системе (ПС) "Прогноз", строится по принципу последовательного влияния разрушения выделенных структурных элементов на безопасность всей исследуемой конструкции.

Наиболее значимым, с точки зрения безопасности, является процесс разрушения конструкции, приводящий к максимальному ущербу [1]. Анализ статистики разрушений показывает, что разрушения происходят в локализованных зонах, характеризующихся конструктивными или технологическими концентраторами напряжений. При формировании геометрической модели пользователь выделяет такие наиболее опасные локализованные зоны в отдельные структурные элементы, называемые конструктивными элементами. Безопасность анализируемой конструкции определяется в соответствии с логической схемой объединения конструктивных элементов, формирующих конструкцию в целом.

Для анализа процесса зарождения трещин конструктивный элемент разбивается на ячейки. Разрушение конструктивного элемента описывается на основе моделей зарождения и роста дефекта (или дефектов) в виде трещины и достижения им предельного состояния. Модель зарождения трещины, в свою очередь, базируется на понятии скалярной меры повреждений, которая трактуется в терминах образования зародышей макроскопических трещин [2]. В этом случае зарождение макротрещины происходит в объеме, который велик по сравнению с характерным размером микронеоднородности материала (размером зерна), но мал по сравнению с характером изменения функции напряжения в объеме исследуемой конструкции. Выполнив декомпозицию конструктивного элемента на ячейки, имеющие размер эталонного объема, можно построить структурную модель, представленную на рис. 1 и включающую следующие структурные элементы: исследуемая конструкция, конструктивный элемент, ячейка.

Рис. 1. Структурная модель исследуемой конструкции в ПС "Прогноз"

Для нахождения предельного состояния ячейки предварительно должны быть рассчитаны температурные и напряженно-деформированные поля в исследуемой конструкции. Для расчета этих полей использовался метод конечных элементов с базовыми восьмиузловыми призматическими элементами. Значения напряжений и температур в ячейках определялись в результате интерполяции по известным значениям напряжений и температур в узлах конечного элемента.

Сформированная таким образом структурная модель позволяет провести анализ безопасности оборудования при его эксплуатации.

Пример проведения анализа безопасности оборудования на основе ПС "Прогноз"

Ниже представлен пример анализа безопасности исследуемой конструкции на основе разработанной ПС "Прогноз". В качестве объекта исследования была рассмотрена конструкция коллектора парогенератора [3], используемая в реакторной установке ВВЭР-1000.

В соответствии с расчетной последовательностью были выполнены следующие этапы анализа:

1. *Формирование сценариев эксплуатации оборудования*. Сценарии эксплуатации для анализа безопасности формировались предварительно экспертами. В качестве примера на рис. 2 приведен сценарий эксплуатации блока, включающий 16 режимов, их сочетания в виде отдельных последовательностей (циклограмм), число повторений каждой из циклограмм и частота появления расчетного сценария.

2. *Проведение численных расчетов на основе МКЭ для нахождения температурного и напряженно-деформированного полей*. Для каждого из режимов были рассчитаны температурные и напряженно-деформированные поля, которые объединялись в соответствии с правилами формирования сценария эксплуатации.

Для удобства реализации этого этапа в ПС "Прогноз" разработаны специальные языковые средства, позволяющие автоматизировать процесс формирования заданных сценариев на основе предварительно рассчитанных полей температур и напряжений. В левой части оконного интерфейса, представленного на рис. 2, сгруппированы исходные режимы, имеющие соответствующие идентификаторы. Каждый из режимов характеризуется рассчитанными предварительно полями напряжений — в виде тензоров, и температур, которые хранятся в виде файлов заданной структуры. При объединении заданных режимов происходит их "сшивка" и образование циклограмм — последовательности режимов, повторяющейся в процессе эксплуатации заданное число раз. Формирование циклограмм показано на рис. 2 в средней части оконного интерфейса. Далее на основе сформированных циклограмм формируется цепочка режимов, моделирующая заданный сценарий эксплуатации, порядок формирования которой приведен в правой части интерфейса.

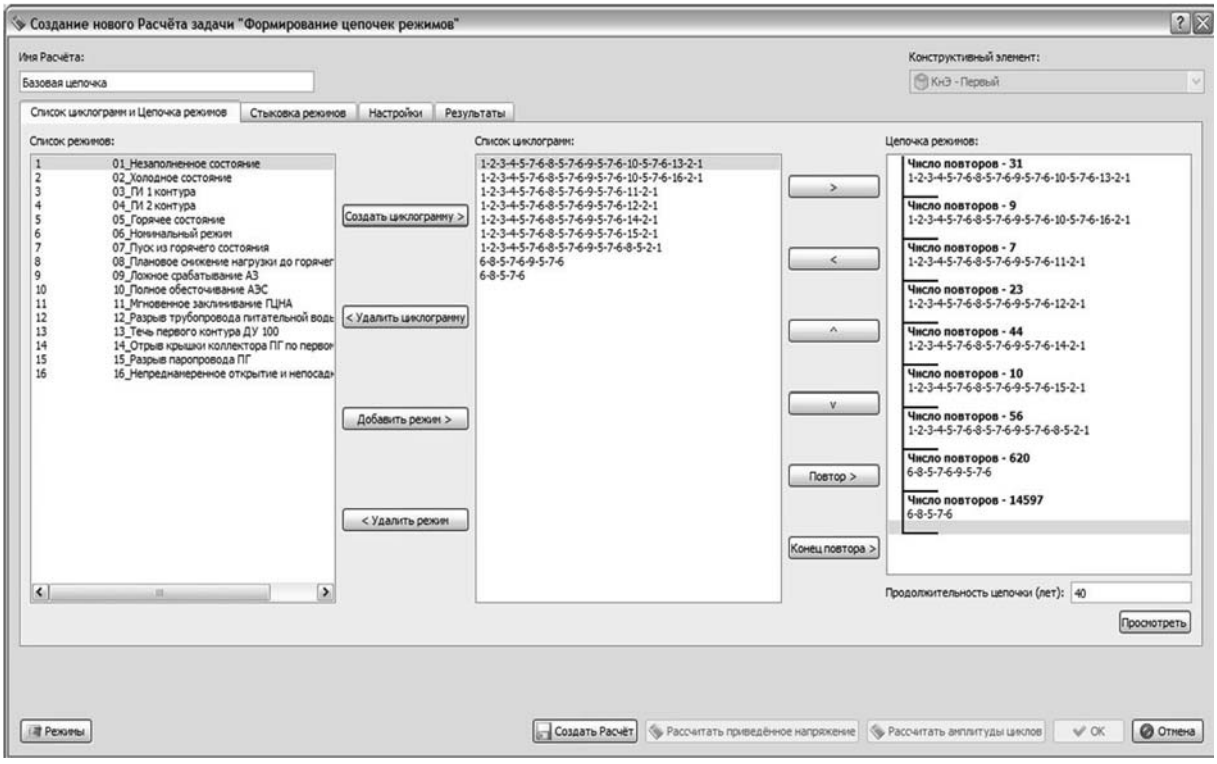


Рис. 2. Режимы эксплуатации, объединяемые в циклограммы, и формирование результирующей цепочки

3. *Формирование структурной модели исследуемой конструкции.* Формирование структурной модели исследуемой конструкции было выполнено в соответствии с правилами, описанными выше, при этом в качестве конструктивного элемента была выделена перемычка коллектора. Базовые геометрические объекты, с помощью которых может быть сформирована сама перемычка, представлены на рис. 3. При анализе напряженно-деформированного состояния была выделена наиболее напряженная зона в виде базового элемента *б*), изображенного на рис. 3. Для этого базового элемента была сгенерирована сеть ячеек с шагом 1 мм. Алгоритм генерации ячеек приведен в работе [4].

4. *Определение критериальных соотношений для предельных состояний ячейки.* Вероятность повреждения ячейки рассматривается как случайное событие, определяемое вероятностью достижения предельного состояния, когда повреждаемость ячейки достигает значения, равного единице. Использованная в ПС "Прогноз" модель расчета повреждаемости ячейки основана на методике, представленной в отраслевом стандарте [5], и учитывает повреждаемость ячейки от различных факторов, включая циклические усталостные напряжения при воздействии водной среды.

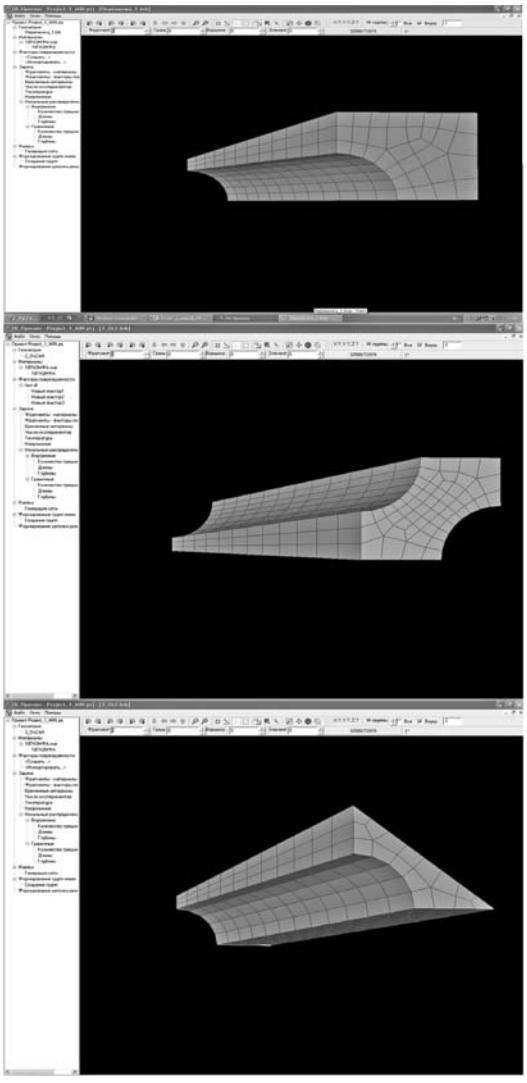
В соответствии с разработанной моделью [3] для каждой ячейки выполняется определенная последовательность расчетов:

1) формирование циклограмм компонент тензоров напряжений в соответствии с заданной последовательностью расчетных режимов;

2) формирование циклограмм главных напряжений по циклограммам компонент тензоров напряжений;

3) определение приведенных циклических напряжений и скоростей деформаций;

4) формирование расчетных циклов методом теней;

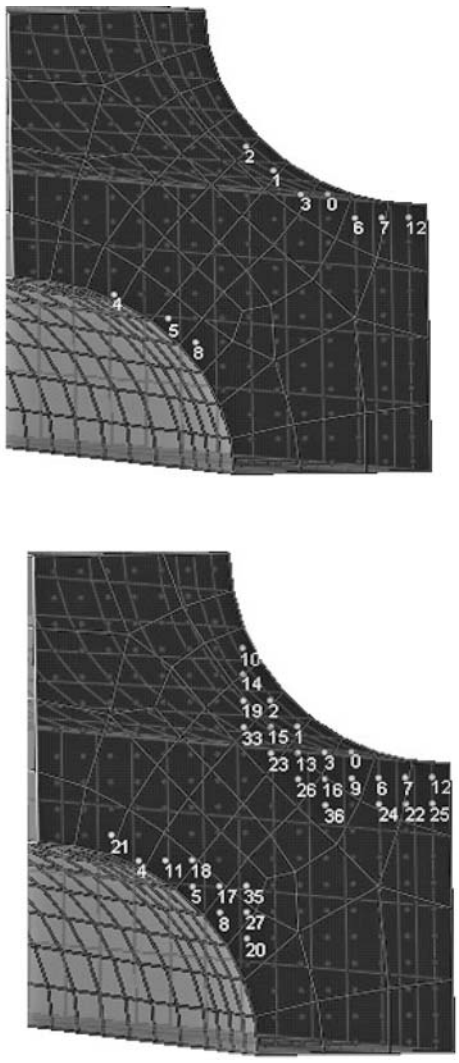


а)

б)

в)

Рис. 3. Базовые элементы, образующие перемычку коллектора



а)

б)

Рис. 4. Местоположение поврежденных ячеек:
а — на момент времени начала зарождения дефектов; *б* — на момент времени активного зарождения дефектов

5) определение для каждого расчетного цикла допустимого числа циклов по кривой усталости с учетом влияния водной среды;

6) определение суммарной повреждаемости в соответствии с заданной последовательностью расчетных режимов;

7) определение вероятности повреждаемости ячейки.

На рис. 4 представлены результаты расчетов, показывающих местоположение поврежденных ячеек в разные моменты времени работы коллектора парогенератора при выбранных условиях эксплуатации. Следует обратить внимание, что каждая поврежденная ячейка характеризуется вероятностью повреждаемости, которая зависит от времени, т. е. каждая из ячеек на данный момент времени имеет различное значение вероятности повреждаемости.

5. *Определение критериальных соотношений для предельных состояний конструктивного элемента.* Предельное состояние конструктивного элемента — перемычки коллектора, определялось моментом, когда на поверхности перемычки образуется сквозная поверхностная трещина. Рядом расположенные поврежденные ячейки, сливаясь, образуют трещину. Если трещина образуется от одной границы области перемычки до другой, то предполагалось, что в этом случае и образуется сквозная поверхностная трещина.

Для расчета вероятности образования трещин была разработана специальная структура данных — матрица повреждаемости, которая позволяет помечать поврежденные ячейки и затем объединять соседние поврежденные ячейки в трещины. Структурная организация матрицы повреждаемости соответствует структуре сети ячеек и представляет собой двухмерный массив, индексируемый по двум целочисленным координатам. Каждый элемент матрицы соответствует отдельной ячейке в соответствии с ее координатой. При повреждении ячейки значение матрицы повреждаемости становится равным 1, иначе равно 0 (рис. 5). Имитационное моделирование процесса повреждения ячейки осуществляется с помощью датчика случайных чисел путем сопоставления для каждой отдельной ячейки случайно сгенерированного числа со значением вероятности повреждаемости этой ячейки на заданный момент времени. Поврежденные ячейки объединяются в трещины с помощью разработанного алгоритма, который выполняет следующие действия:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
10	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
11	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
12	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Рис. 5. Структура матрицы повреждаемости

1) проводит просмотр элементов матрицы повреждаемости до тех пор, пока не обнаружит поврежденную ячейку;

2) заносит данную ячейку в новый фронт распространения трещины и обнуляет соответствующий элемент матрицы повреждаемости, чтобы не учитывать данную ячейку в дальнейшем поиске;

3) определяет всех поврежденных соседей данной ячейки, заносит их во фронт трещины, обнуляет соответствующие элементы в матрице повреждаемости и для каждой из них определяет всех поврежденных соседей;

4) продолжает работать до тех пор, пока все соседи последней ячейки, вошедшей во фронт, окажутся неповрежденными;

5) возвращается к шагу 1 и продолжает просмотр элементов матрицы, пока не обнаружит очередную поврежденную ячейку, которая станет началом нового фронта распространения трещины.

Из всех сформированных в результате работы алгоритма трещин выбирается сквозная, т. е. такая трещина, траектория которой содержит крайние ячейки границы перемычки.

После проведения всех вычислительных экспериментов для данного момента времени вычисляется вероятность (частота) образования сквозных трещин как отношение числа экспериментов, в которых сквозная трещина образовывалась, к общему числу экспериментов:

$$P = \frac{N_1}{N},$$

где P — вероятность (частота) образования сквозной трещины; N_1 — число экспериментов, в которых сквозная трещина образовывалась; N — общее число экспериментов.

Результатом работы алгоритма является распределение вероятностей образования сквозных трещин по моментам времени (рис. 6).

Анализ данного распределения позволяет выделить следующие периоды процесса зарождения и развития трещин в перемычке коллектора [3]:

I — инкубационный период — протекает от начала эксплуатации до появления первых зародышей макротрещин. Данный период определяет скорость роста повреждений ячеек;

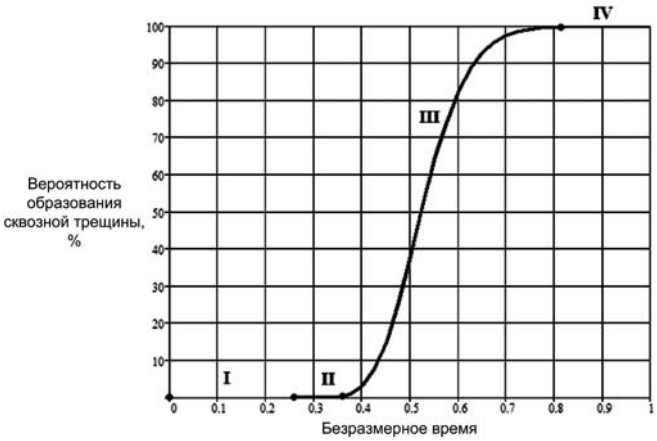


Рис. 6. Распределение вероятности образования сквозной трещины по моментам времени

II — период роста трещин — протекает от появления первых зародышей до момента первого образования сквозной трещины на поверхности перемычки;

III — период вероятностного образования сквозной трещины — протекает от момента образования первой сквозной трещины до момента достижения вероятностью образования сквозной трещины значения, равного единице. Данный период характеризуется моментом возможного образования хотя бы одной сквозной трещины;

IV — период детерминированного развития трещины характеризуется тем, что находящиеся на поверхности перемычки ячейки, образующие сквозную трещину, повреждены со стопроцентной вероятностью.

6. *Определение суммарной вероятности разрушения исследуемой конструкции — коллектора парогенератора*. Зная вероятности отказа каждого конструктивного элемента на заданный момент времени для сформированного сценария, можно определить вероятность отказа исследуемой конструкции. Для этого необходимо использовать структурную схему, созданную в начале анализа. Безотказная работа коллектора парогенератора может быть найдена по теореме умножения независимых событий — неотказов каждого из элементов. Тогда условная вероятность отказа (разрушения) всей конструкции $P_i(\tau)$ для i-го расчетного сценария определяется следующим соотношением:

$$P_i(\tau) = 1 - \prod_{j=1}^{M} (1 - P_{i,j}(\tau)),$$

где M — число конструктивных элементов; $P_{i,j}(\tau)$ — условная вероятность отказа (разрушения) j-го конструктивного элемента при условии реализации i-го расчетного сценария.

Если известна частота появления расчетного сценария f_i, то вероятность отказа (разрушения) всей конструкции за весь рассматриваемый период эксплуатации можно найти следующим образом:

$$P_\Sigma(\tau) = \sum_{i=1}^{L} f_i P_i(\tau),$$

где L — число рассматриваемых сценариев эксплуатации исследуемой конструкции.

Заключение

Разработанная содержательная структурная модель, основанная на декомпозиции исследуемой конструкции на элементы, влияющие на ее безопасность, позволяет провести анализ безопасности эксплуатации оборудования при различных сценариях, моделирующих работу оборудования в реальных условиях, при различных режимах, включая и аварийные.

Реализованные численные алгоритмы последовательного разрушения структурных элементов модели от ячейки (образование зародышей трещин), конструктивных элементов (образование сквозных трещин, приводящих к отказу конструктивных элементов) до исследуемой конструкции дают возможность, в свою очередь, проследить последовательность разрушения конструкции при ее эксплуатации и рассчитать безопасность ее работы.

Список литературы

1. **Анализ** риска и проблемы безопасности. В 4-х частях. Ч. 1. Основы анализа и регулирования безопасности. Науч. руковод. К. В. Фролов. М.: МГФ "Знание", 2006. 640 с.

2. **Болотин В. В.** Ресурс машин и конструкций. М.: Машиностроение, 1990. 448 с.

3. **Александров А. Е., Александров П. А., Востриков А. А.** Отчет по верификации программного комплекса "Прогноз" по расчету вероятности повреждаемости перемычек коллектора парогенератора РУ ВВЭР-1000. Рег. номер № 01201056086. М: МГУПИ. 2010. 167 с.

4. **Александров А. Е., Востриков А. А., Шалыминов И. В.** Алгоритм формирования геометрических объектов для анализа зарождения трещин // Программное и информационное обеспечение систем различного назначения на базе персональных ЭВМ: Межвузовский сб. науч. тр. М.: МГУПИ, 2009. Вып. 12. С. 146—152.

5. **Руководство** по расчету на прочность оборудования и трубопроводов реакторных установок РБМК и ВВЭР на стадии эксплуатации. РД ЭО 0330-01. ГП "Российский государственный концерн по производству электрической и тепловой энергии на атомных станциях" концерн "Рорэнергоатом". 2003. 137 с.

УДК 656.2:519.852

А. В. Горохов, аспирант,
Московский государственный университет
путей сообщения (МИИТ),
e-mail: agorokhov.miit@gmail.com

Алгоритмизация поиска источников хаотических колебаний в крупномасштабных энергосистемах

Исследована задача поиска источника колебательных процессов в электроэнергетической системе. Представлены два алгоритма оперативного решения задачи, основанные на поиске с выбором наилучшего направления и на параллельном рекурсивном обходе графа. На основе сравнительного анализа предложенных алгоритмов сделаны выводы о возможности их информационно-технологического применения.

Ключевые слова: *критериальная функция на графе, параллельные вычисления, синхронный генератор, электроэнергетическая система, хаотические колебания*

Введение

При неправильных настройках автоматических регуляторов в электроэнергетических системах могут возникать автоколебания, а также детерминированные колебательные процессы, сходные со случайными по внешнему виду и по статистическим характеристикам (в дальнейшем будем называть такие процессы хаотическими). Наличие таких процессов не только нежелательно с точки зрения стабильности напряжения и частоты, но и опасно в сочетании с возможными большими возмущениями. Хаотические колебания в нерегулируемых энергосистемах были исследованы в работах [1—4]. В работе [5] с помощью численного отображения Пуанкаре была показана возможность возникновения хаотических колебаний в одиночном регулируемом генераторе.

В данной статье предложен алгоритм выявления единственного источника таких колебаний в динамических системах и приведены результаты его работы на примере модели энергосистемы, насчитывающей более тысячи генераторов и несколько тысяч узлов электрической сети. Алгоритм основан на вычислении критериальной функции, характеризующей уровень колебаний. Полученные в ходе исследования результаты можно применять как при эксплуатации энергосистем, так и при разработке и использовании программных продуктов, моделирующих параллельную работу синхронных генераторов.

Постановка задачи

Пусть имеется граф электрической сети $G(\bar{I}, U, \Gamma)$, где $\bar{I}$ — множество вершин графа (узлов электрической системы); U — множество ребер (линий электропередачи, трансформаторов, элементов продольной компенсации), Γ: $U \to \bar{I} \times \bar{I}$ — отображение множества ребер на декартово произведение вершин, задающее граничные вершины каждого ребра. На множестве вершин определена действительная критериальная функция $F(I)$: $\bar{I} \to R$. Предполагается, что эта функция неотрицательная ограниченная и ее глобальный максимум достигается в той вершине, где расположен искомый источник больших колебаний:

$$I^* = \text{argmax } F(I) \text{ на множестве } \bar{I}.$$

Таким образом, поиск местоположения источника колебаний сводится к максимизации критериальной функции на конечном графе. Наиболее очевидным методом решения подобной задачи является полный перебор вершин графа. Такой подход вполне приемлем в процессе разработки и отладки математической модели, но, учитывая, что реальные энергосистемы насчитывают десятки тысяч географически разнесенных узлов, при поиске источника колебаний в условиях эксплуатации необходимо принимать во внимание затраты времени на сбор, передачу и обработку информации. Например, если опрос одного узла занимает 6 с, на сети даже из 1000 узлов полный перебор займет более 1,5 ч, что при быстром распространении колебаний по сети может быть недопустимо долго для принятия оперативных мер. В этих условиях необходимо применять существенно более "быстрые" алгоритмы, разработке которых посвящена статья.

Общая структура модели исследуемой энергосистемы

Модель, которая была использована для проведения исследований, основана на приведенной в трудах Американского общества инженеров-электриков тестовой модели и представляет собой систему дифференциально-алгебраических уравнений электрической сети и генераторов с регуляторами возбуждения и частоты вращения первичных двигателей [6]. Приведем некоторые численные характеристики исследуемой модели. В ее состав входит 5900 узлов, в том числе 1349 генераторов. В системе насчитывается 2680 отдельных динамических моделей и 10 669 дифференциальных переменных.

Модель представляет собой систему дифференциально-алгебраических уравнений следующего вида:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{z}),$$

$$\mathbf{g}(\mathbf{x}, \mathbf{z}) = 0,$$

где $\mathbf{x}$ — n-мерный вектор дифференциальных переменных; $\mathbf{z}$ — m-мерный вектор алгебраических переменных; $\mathbf{f}$ и $\mathbf{g}$ — вектор-функции размера n и m. Уравнения сети записаны с помощью метода узловых потенциалов:

$$\mathbf{Yv} = \mathbf{J}(\mathbf{x}, \mathbf{v}),$$

где $\mathbf{J}$ — комплексный вектор узловых токов; $\mathbf{v}$ — комплексный вектор узловых потенциалов; $\mathbf{Y}$ — комплексная матрица узловых проводимостей; $\mathbf{x}$ — вектор углов и частот вращения. Связь динамических моделей генераторов с уравнениями электрической сети осуществляется с помощью вектора узловых токов [6].

Сценарий возникновения колебаний состоит в обрыве цепи гибкой обратной связи в автоматическом регуляторе возбуждения. В результате этого повреждения теряется устойчивость положения равновесия системы генератор—турбина—регулятор возбуждения и развиваются колебания большой амплитуды. На рис. 1 приведена кривая напряжения генератора в относительных единицах (о. е.). Вид кривой свидетельствует либо о существовании очень длинного периода, либо о хаотическом виде колебаний. В то же время на кривой напряжения возбуждения имеет место периодический процесс с очень большой амплитудой и периодом 2,5 с (рис. 2). Сопоставление кривых позволяет сделать вывод, что колебания возбуждаются в контуре генератор — автоматический регулятор напряжения. При этом колебания становятся, по меньшей мере, квазихаотическими в результате взаимодействия с энергосистемой. Более подробно вопрос развития хаотических колебаний в генераторах исследован в работах [5, 7, 8].

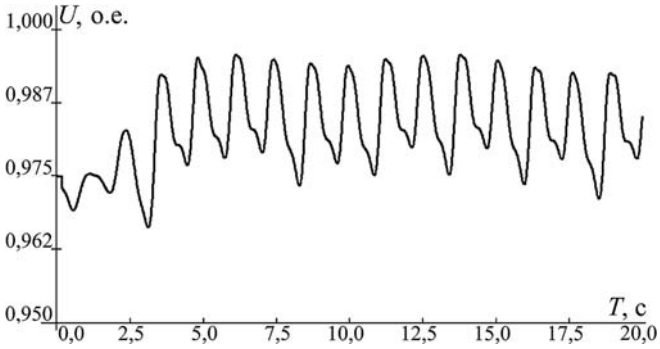


Рис. 1. Кривая напряжения

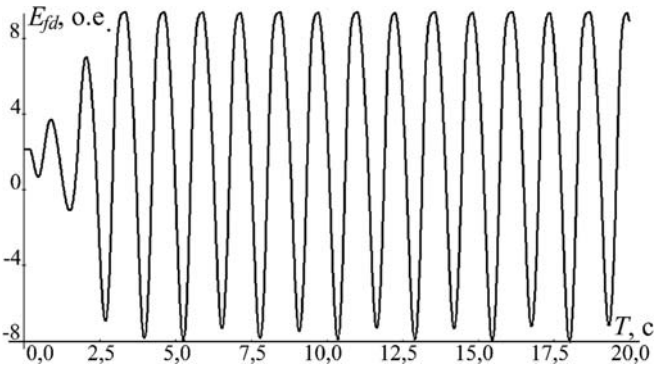


Рис. 2. Кривая напряжения возбуждения

Выбор критериальной функции

Критерий оценки поведения системы должен обладать следующими свойствами: 1) быть определен на всех узлах графа энергосистемы, не только на динамических; 2) допускать быстрое непосредственное измерение.

Наилучшим образом этим требованиям отвечают критерии, основанные на использовании модуля или фазы напряжения U. Хотя в настоящее время, благодаря спутниковым системам GPS и ГЛОНАСС, появилась возможность эффективного измерения фазовых сдвигов между напряжениями в удаленных узлах энергосистем [9], более надежным, быстродействующим и менее емким по объему информации является измерение максимальных значений напряжения. Так как хаотический процесс сопровождается значительными нерегулярными изменениями модуля напряжения на шинах генератора, для определения размаха колебаний требуется наблюдение в течение некоторого характеристического интервала времени.

На основании приведенных соображений в качестве критериальной функции была выбрана ΔU — разность между максимальным и минимальным значениями напряжения на генераторе с номером I в течение заданного интервала времени T_χ:

$$\Delta U(I, T_\chi) = \max_t U(I, t) - \min_t U(I, t), \quad t \in \lfloor t_x, t_x + T_\chi \rfloor. \qquad (1)$$

При хаотических колебаниях функция (1) ведет себя как случайная величина, вероятностные характеристики которой зависят от длины характеристического интервала T_χ.

Для хаотических процессов, протекающих в ограниченной области фазового пространства, функция $\Delta U(I, T_\chi)$ является ограниченной и монотонно неубывающей по T_χ как разность ограниченных монотонно неубывающей и монотонно невозрастающей по T_χ функций. Как ограниченная монотонно неубывающая функция $\Delta U(I, T_\chi)$ имеет ко-

нечный предел, т. е. стремится к константе с ростом T_χ. Следовательно, дисперсия $\Delta U(I, T_\chi)$ с увеличением длины характеристического интервала T_χ монотонно уменьшается, стремясь к нулю.

Автором была предложена следующая методика выбора характеристического интервала времени, достаточного для вычисления размаха колебаний: для оценки характеристического интервала строится зависимость дисперсии функции (1) от длины интервала измерения; в качестве длины характеристического интервала выбирается значение, для которого дисперсия не превышает допустимой дисперсии оценки. Допустимая дисперсия принимается равной квадрату среднего квадратического отклонения, присущего используемым приборам измерения напряжения.

Приведем пример применения указанного подхода. В целях определения характеристического интервала времени, достаточного для выявления размаха колебаний, найдем по выборке из 10 000 значений зависимость дисперсии размаха от длительности интервала в пределах от 2 до 10 с. Результаты расчета, приведенные в табл. 1, показывают, что уже при интервале 9 с дисперсия критериальной функции $\sigma^2_{\Delta U}$ определяется в пределах ошибки вычислений. Таким образом, для расчетов можно использовать интервал длиной 9 с.

Таблица 1

Дисперсия критериальной функции

T, с	2	4	6	8	10
$\sigma^2_{\Delta U}$	0,000304	0,00018	0,000135	0,000102	0,000006

Многоэкстремальность критериальной функции. Пути поиска и область влияния

Критериальная функция (1) может иметь множество локальных экстремумов на узлах, не являющихся источниками хаотических колебаний. Как следствие, любой алгоритм, в котором не заложена возможность поиска всех экстремумов и непосредственного сравнения всех локально-экстремальных значений критериальной функции, должен иметь способ определить, порожден ли найденный экстремум источником больших колебаний или нет. В качестве такого способа, основываясь на исследованиях, показавших, что переход в хаотический режим характеризуется резким увеличением амплитуды колебаний [7, 8], можно ввести нижний порог отбора $\Delta U^*_{\min}$ — минимальное значение критериальной функции для признания экстремума глобальным. Для модельной сети расчет показал, что ни в одном из ее локальных экстремумов значение ΔU не превышает 0,05 относительных единиц, при этом в глобальном экстремуме $\Delta U \approx 0{,}4$. Исходя из этого в вычислительных экспериментах для данной системы порог $\Delta U^*_{\min}$ был принят равным 0,1. В реальной эксплуатации имеет смысл аналогично выбирать порог отбора таким образом, чтобы он заведомо превышал максимально допустимое для нормального режима значение амплитуды колебаний.

В случае если найденный экстремум не является глобальным, будем случайным образом выбирать одну из еще не пройденных алгоритмом вершин и продолжать поиск, начиная с нее. Такой переход будем называть *скачком*. Учитывая конечность числа вершин графа, можно заключить, что за конечное число скачков либо будет обнаружен глобальный максимум на графе, либо будет установлено отсутствие вершин, удовлетворяющих порогу отбора. Последовательность узлов, которую проходит поиск от начального узла до искомого, будем называть *путем*. Путь, не содержащий скачков, будем называть *непрерывным*. Множество начальных узлов, для которых существуют непрерывные пути, будем называть *алгоритмической областью монотонности* искомого узла (в дальнейшем — область монотонности). Заметим, что для разных алгоритмов область монотонности может отличаться.

Алгоритм поиска по наилучшему направлению

Для решения задачи был разработан алгоритм, представляющий собой поиск в глубину [10] с выбором наилучшего направления. Исходными данными алгоритма являются номер начального узла I_0 и нижний порог отбора $\Delta U^*_{\min}$. Алгоритм содержит два основных шага.

Шаг 1. Обойти все смежные с текущим I_i неотмеченные узлы $I^i_1 \dots I^i_m$, отметив как пройденные их и текущий узел.

Шаг 2. Проверить, есть ли среди смежных узлов такие, что критериальная функция на них имеет большее значение, чем на текущем:

$$\exists j\colon \Delta U(I^i_j) > \Delta U(I_i).$$

A. Если такие узлы есть, выбрать в качестве текущего тот из них, который имеет наибольшее значение критериальной функции, и перейти к шагу 1:

$$\hat{I}_i = I^*\colon \Delta U(I^*) = \max_{I \in I^i_1 \dots I^i_j} \Delta U(I).$$

B. Если такие узлы отсутствуют:

i. если значение критериальной функции на текущем узле превышает нижний порог отбора

($\Delta U(I_i) \geqslant \Delta U^*_{\min}$) и на узле имеется устройство регулирования напряжения, считать, что это устройство и есть искомый источник колебаний;

ii. в противном случае из множества еще не пройденных узлов выбрать случайный узел I_k с устройством регулирования напряжения; если значение критериальной функции на нем меньше уже найденного ($\Delta U(I_k) < \Delta U(I_i)$), повторить случайный выбор; иначе выбрать узел I_k в качестве текущего и перейти к шагу 1.

Свойства алгоритма.

1. Алгоритм является монотонно неубывающим по критерию, что следует из шага 2Bii.

2. Алгоритм конечен, так как, как было показано выше, конечно число скачков.

Оценка времени выполнения алгоритма T_o зависит от способа его реализации:

- если обход соседних узлов в шаге 1 происходит последовательно *(поиск по наилучшему направлению с последовательным расчетом)*, то

$$T_o = O(N), \tag{2}$$

где N — количество вычислений критериальной функции;

- если модифицировать алгоритм таким образом, чтобы вычисление ΔU на соседних узлах происходило одновременно *(поиск по наилучшему направлению с параллельным расчетом)*, то

$$T_o = O(L), \tag{3}$$

где L — длина полученного пути.

Оценки связаны соотношением $N <= (q - 1)L$, где q — максимальная локальная степень вершины графа.

Рекурсивный алгоритм поиска с использованием параллельных вычислений

Так как в реальной системе поиск происходит на тысячах независимых узлов, вычисление критериальной функции может проходить одновременно на нескольких узлах. Таким образом, можно разработать алгоритм, с помощью которого за счет параллельных вычислений получают результат за меньшее число итераций. Для распараллеливания алгоритма необходимо его разбить на независимые подзадачи. В данном случае это можно обеспечить с помощью рекурсивной организации алгоритма следующим образом: для каждого очередного узла I_i, где i — номер шага алгоритма, определена функция $F_{\Delta U}(I_i,\ F_{\Delta U}(I_1^i)...F_{\Delta U}(I_n^i))$, где $I_1^i ... I_n^i$ — смежные с I_i узлы. Данная функция определяется следующим рекуррентным соотношением:

$$F_{\Delta U}(I_i,\ F_{\Delta U}(I_1^i,\ ...)...F_{\Delta U}(I_n^i,\ ...)) =$$
$$= \begin{cases} \max(\Delta U(I_i),\ F_{\Delta U}(I_1^i,\ ...)...F_{\Delta U}(I_n^i,\ ...)), \\ 0 \text{ для уже пройденных узлов.} \end{cases} \tag{4}$$

Таким образом, значением $F_{\Delta U}$ является максимальное значение ΔU на ветви поиска, состоящей из узла I_i, еще не пройденных узлов из числа смежных с ним, еще не пройденных узлов из числа смежных с данными узлами и т. д. Тогда для начального узла значением $F_{\Delta U}(I_0,\ ...)$ будет локальный максимум значения критериальной функции.

Алгоритм, использующий соотношение (4), может быть реализован через обмен сообщениями между узлами. Для каждого очередного узла I_i, кроме I_0, нужно выполнить следующие шаги:

1. Если текущий узел I_i уже был пройден одной из ветвей поиска, передать на предшествующий узел нулевое значение ΔU.

2. Измерить ΔU на узле I_i.

3. Если значение $\Delta U(I_i)$ меньше, чем на предшествующем, передать на предшествующий узел нулевое значение ΔU.

4. Если у узла есть смежные узлы, кроме предшествующего, начать следующий уровень рекурсии, приняв каждый из смежных узлов ($I_1^i ... I_n^i$) в качестве текущего с предшествующим узлом I_i.

5. После того, как завершится измерение $\Delta U(I_i)$ (шаг 2), и каждый из смежных узлов ($I_1^i ... I_n^i$) передаст на узел I_i полученное значение $F_{\Delta U}$ (шаг 6), выбрать среди найденных узлов такой, что

$$\hat{I}_i = I^* : \Delta U(I^*) = \max_{I \in I_i, I_1^* ... I_j^*} \Delta U(I),$$

где I_i^* — узел, на котором получено наибольшее значение ΔU для ветви поиска, начатой с узла I_i^0.

6. Передать на предшествующий узел полученное значение $F_{\Delta U}$ и номер узла.

Для начального узла I_0 необходимо выполнить следующие шаги:

1. Начать следующий уровень рекурсии, приняв каждый из смежных узлов ($I_1^0 ... I_n^0$) в качестве текущего с предшествующим узлом I_0.

2. Измерить ΔU на узле I_0.

3. После того, как завершится измерение $\Delta U(I_0)$ (шаг 2) и каждый из смежных узлов ($I_1^0 ... I_n^0$) передаст на узел I_0 полученное значение $F_{\Delta U}$ (шаг 6 ал-

горитма для I_i), выбрать среди найденных узлов такой, что

$$\hat{I}_0 = I^* : \Delta U(I^*) = \max_{I \in I_0, I_1^* \ldots I_j^*} \Delta U(I),$$

где I_i^* — узел, на котором получено наибольшее значение ΔU для ветви поиска, начатой с узла I_i^0;

4. По правилу, описанному для предыдущего алгоритма, либо считать найденный узел искомым, либо совершить случайный выбор узла.

Оценка времени выполнения алгоритма T_o зависит от способа его реализации:

- если алгоритм во всех случаях осуществляет выбор узла с максимальной ΔU только после завершения всех ветвей поиска *(рекурсивный поиск с ожиданием завершения всех ветвей)*, то

$$T_o = O(L_{\max}), \quad (5)$$

где $L_{\max}$ — длина наибольшего из найденных путей;

- можно модифицировать алгоритм следующим образом: если у текущего узла нет соседних, кроме исходного, и значение ΔU на узле превышает $\Delta U_{\min}^*$, считать текущий узел искомым и выполнить останов поиска (*рекурсивный поиск с остановом по первому обнаружению*); в таком случае

$$T_o = O(L_{\min}), \quad (6)$$

где $L_{\min}$ — длина наименьшего из путей, ведущих к искомому узлу.

Исследование представленных алгоритмов

Для сравнительного анализа алгоритмов на вычислительном кластере МИИТа был выполнен расчет путей для всех 5900 возможных начальных узлов. Для каждого из путей были получены значения критериев эффективности согласно (2), (3), (5), (6). В табл. 2 сведены средние значения критериев оценки времени T_o: $\overline{N}$, $\overline{L}$, $\overline{L}_{\max}$, $\overline{L}_{\min}$ и средние квадратичные отклонения σ_N, σ_L, $\sigma_{L_{\max}}$, $\sigma_{L_{\min}}$ по путям из всех узлов, по непрерывным путям из области монотонности соответствующего алгоритма, и по путям со скачками, т. е. не входящим в область монотонности соответствующего алгоритма.

Сравнение полученных результатов показывает следующее:

1. При поиске путем полного перебора требуется просмотр каждого узла, т. е. всего 5900 просмотров. Как видно, даже поиск по наилучшему направлению с последовательным расчетом, который обладает наименьшей эффективностью из всех исследуемых алгоритмов, занимает в среднем в 40 раз меньшее время, чем полный перебор.

Таблица 2

Сравнение последовательного и рекурсивного алгоритмов

Последовательный алгоритм			
Критерии	Все пути	Непрерывные пути	Пути со скачками
$\overline{N}$	153,74	57,735	159,957
σ_N	93,39	20,3	92,87
$\overline{L}$	16,856	10,329	17,279
σ_L	5,549	3,876	5,374
Область монотонности	359		
Рекурсивный алгоритм			
Критерии	Все пути	Непрерывные пути	Пути со скачками
$\overline{L}_{\max}$	24,587	19,232	24,984
$\sigma_{L\max}$	8,059	7,49	8,1
$\overline{L}_{\min}$	18,61	7,275	19,841
$\sigma_{L\min}$	7,055	2,55	6,735
Область монотонности	902		

2. Можно видеть, что для рекурсивного поиска с ожиданием завершения всех ветвей среднее значение критерия по путям из области монотонности в 2 раза больше, чем для последовательного поиска с параллельным расчетом; при этом среднее значение по всем путям в 1,5 раза больше для данного варианта рекурсивного поиска, чем для последовательного.

3. Для рекурсивного поиска с остановом по первому обнаружению среднее значение критерия по всем путям в 1,1 раза больше, чем для последовательного поиска с параллельным расчетом; несмотря на это, по путям из области монотонности данный вариант рекурсивного поиска занимает в 1,4 раза меньшее время. При этом стоить учитывать, что область монотонности для рекурсивных алгоритмов включает в 2,5 раза больше путей.

На основании результатов проведенного сравнения можно сделать вывод, что для начальных узлов из области монотонности (как было сказано выше, это наиболее часто встречающийся в реальной эксплуатации случай) целесообразно использовать рекурсивный поиск с остановом по первому обнаружению. В то же время для поиска в системе, начинающегося с любого узла, может быть применен и поиск в наилучшем направлении с параллельным расчетом.

Список литературы

1. **Chih-Wen Liu, Thorp J. S., Jin Lu, Thomas R. J., Hsiao-Dong Chiang.** Detection of transiently chaotic swings in power systems using real-time phasor measurements // Transactions of IEEE on Power Systems. N. Y.: Cornell University, ORA Corporation. Ithaca. August 1994. Vol. 9, N 3.

2. **Chin-Woo Tan, Varghese M., Varaiya P.** Bifurcation, chaos and voltage collaps in power systems // Proc. of IEEE. November 1995. Vol. 83, N 11.

3. **Kwatny H. G., Fischl R. F., Xwankpa C. O.** Local bifurcation in power system // Proc. of IEEE. November 1995. Vol. 83, N 11.

4. **Moiola J. L., Revel G., Alonso D., Itovich G., Robbio F. I.** On the use of a frequency method for classifying the oscillatory behavior in nonlinear control problems // Proc. of First Iberoamerican Meeting on Geometry, Mechanics and Control. Santiago de Compostela, June 23—27, 2008.

5. **Иванова А. П., Эпштейн Г. Л.** Опыт анализа хаотических колебаний в электроэнергетической системе // Труды Четвертой международной конференции "Средства математического моделирования", "Математические исследования". Санкт-Петербург. 2003. Том 10. С. 62—66.

6. **Kundur P.** Power System Stability and Control. New York: McGraw-Hill, 1994. 1196 с.

7. **Братусь А. С., Горохов А. В., Эпштейн Г. Л.** Поиск источников хаотических колебаний в энергосистемах // Мир транспорта. 2010. Т. 33, № 5. С. 12—19.

8. **Горохов А. В.** Анализ и локализация источника хаотических колебаний в сложной энергетической системе // Высокие технологии, фундаментальные исследования, экономика. Т. 2: Труды Двенадцатой международной научно-практической конференции "Фундаментальные и прикладные исследования, разработка и применение высоких технологий в промышленности". 08—10 декабря 2011 года, Санкт-Петербург / под ред. А. П. Кудинова. — СПб.: Изд-во Политехн. ун-та, 2011. С. 51—54.

9. **Phadke A., Thorp J., Adamiak M.** A New Measurement Technique for Tracking Voltage Phasor, Local System Frequency, and Rate of Change of Frequency // IEEE Trans. May 1983. Vol. PAS-102, N 5. P. 1025—1038.

10. **Рейнгольд Э., Нивергельт Ю., Део Н.** Комбинаторные алгоритмы. М.: Мир, 1980.

УДК 519.857

В. И. Струченков, д-р техн. наук, проф.,
e-mail: str1942@mail.ru,
МГТУ МИРЭА

Новые алгоритмы оптимизации в задачах обеспечения надежности сложных систем

Рассматривается возможность использования комбинированного алгоритма решения задачи оптимального распределения ресурса в задачах обеспечения надежной работы сложных систем.

Ключевые слова: *надежность, оптимизация, динамическое программирование, метод ветвей и границ*

Для решения прикладных задач дискретной оптимизации, несмотря на очевидный прогресс в области вычислительной техники, сохраняет актуальность вопрос разработки быстродействующих алгоритмов. Особенно это относится к задачам большой размерности, решение которых встроено в многократно повторяющийся цикл расчетов при проектировании сложных систем.

Наибольшие успехи в этом направлении достигнуты при создании комбинированных методов оптимизации [1—3]. В частности, для решения задачи оптимального распределения ресурса разработан алгоритм динамического программирования с использованием множеств Парето [4], в котором каждое состояние на всех этапах рассматривается как точка ветвления и для каждой ветви строятся прогностические оценки оптимума (нижняя и верхняя границы) [2, 3]. Новый алгоритм оптимального распределения ресурса является комбинацией динамического программирования и метода ветвей и границ. Его особенность состоит в том, что для построения двусторонних оценок оптимума применяется специальный алгоритм "спуска — подъема" [2], позволяющий многократно сократить время счета по сравнению с известными алгоритмами.

Математическая формулировка задачи, для решения которой был разработан новый алгоритм, состоит в следующем.

Найти минимум

$$\sum_{i=1}^{n} g_i(x_i) \qquad (1)$$

при ограничениях

$$\sum_{i=1}^{n} f_i(x_i) \leqslant R,\ x_i \in X_i,\ i = 1, 2, \ldots, n, \qquad (2)$$

где X_i — конечные множества, $f_i(x_i) \geqslant 0$, $g_i(x_i) \geqslant 0$ и $R > 0$. Целочисленность функций $f_i(x_i)$ и $g_i(x_i)$ необязательна. Предполагается, что множество допустимых решений не пусто.

В настоящей статье рассматриваются две задачи, которые формально не имеют отношения к распределению ресурса, но могут быть приведены к виду (1), (2), позволяющему использовать этот быстродействующий алгоритм.

Задача № 1. Оптимальное резервирование

Для повышения надежности сложных систем часто используют резервирование элементов (уз-

лов), т. е. в случае отказа одного элемента используется его резервная копия (дубликат). Если считать отказы каждого элемента и их дубликатов независимыми случайными событиями и обозначить p_i — вероятность отказа i-го элемента, то для двух экземпляров такого элемента вероятность отказа при дублировании равна p_i^2. Соответственно вероятность отказа при наличии x_i экземпляров равна $p_i^{x_i}$. А вероятность сохранения работоспособности элемента равна $1 - p_i^{x_i}$.

Соответственно для системы из N элементов вероятность сохранения работоспособности $P(x)$, которую можно считать мерой надежности, вычисляется по формуле

$$P(x) = \sum_{i=1}^{N} (1 - p_i^{x_i}). \qquad (3)$$

Резервирование требует затрат, поэтому могут быть ограничены суммарные затраты, суммарный вес, объем или какая-либо еще характеристика набора элементов или несколько таких характеристик. Соответственно имеем ограничение

$$\sum_{i=1}^{N} a_i x_i \leqslant R \qquad (4)$$

или несколько ограничений такого вида. Заданные величины $a_i > 0$ — это стоимость, вес, объем или энергопотребление одного экземпляра i-го элемента (узла). Необходимо выбрать такое резервирование для каждого элемента, т. е. такой вектор $x(x_1, x_2, ..., x_N)$, чтобы выполнялось ограничение (4) и была максимальная надежность $P(x)$ или минимальная вероятность отказа системы, т. е. $1 - P(x)$. Поскольку переменные x_i принимают целочисленные значения, задача может быть решена по методу динамического программирования. Основные понятия, используемые в динамическом программировании [5, 6], формализуются следующим образом: очередной шаг — это очередной K-й элемент (узел), состояние системы — это $Q_k = \sum_{i=1}^{K} a_i x_i$, шаговый переход — рассмотрение очередного возможного числа копий K-го элемента, путь — набор уже рассмотренных элементов с заданным числом копий каждого из них, траектория — набор всех элементов с заданным числом копий каждого из них. Если в одно и то же состояние приводят несколько путей, то в соответствии с принципом оптимальности Р. Беллмана остается путь, для которого значение целевой функции максимально.

Если на некотором шаге для состояния A целевая функция (мера надежности) больше, чем для состояния B, и при этом ограничивающая функция Q_k меньше, то состояние B бесперспективно и может быть отброшено вместе со всеми своими продолжениями. Действительно, в состоянии A ничто не мешает в дальнейшем принимать точно такие же решения относительно числа копий всех рассматриваемых далее элементов (узлов), что и в состоянии B, что и означает, что состояние B "отстало навсегда". Другими словами, на каждом шаге динамического программирования остается только паретовское множество состояний [4].

Наличие в целевой функции произведения вместо суммы существенного значения не имеет как при отбраковке путей, приводящих в одно и то же состояние, так и при отбраковке непаретовских точек.

В работе [1] рекомендовано решать задачу на минимум вероятности отказа систем по алгоритму Р. Беллмана. Однако данная задача может быть решена и с использованием множеств Парето, и с отбраковкой части паретовских точек по методу ветвей и границ, что дает существенное снижение времени счета [2]. С этой целью будем решать задачу на максимум вероятности сохранения работоспособности системы, но вместо целевой функции $P(x)$ (3) рассмотрим $\ln P(x)$. Такая замена дает ту же точку максимума из-за монотонности функции логарифм. Однако она позволяет перейти от произведения к сумме и преобразовать задачу к уже решенной задаче (1), (2) распределения ресурса. Действительно,

$$\ln P(x) = \sum_{i=1}^{K} \ln(1 - p_i^{x_i}).$$

Далее вместо задачи на максимум $\ln P(x)$ будем рассматривать задачу на минимум функции минус $\ln P(x)$ при ограничении (4) и получаем задачу (1), (2), в которой

$$f_i(x_i) = a_i x_i;\ g_i(x_i) = -\ln(1 - p_i^{x_i}),$$

где $f_i(x_i) > 0$, $g_i(x_i) > 0$ и $R > 0$; $x_i = 1, 2, ...$

Максимальные значения переменных x_i определяются перебором в процессе счета из условия (4).

По смыслу задачи значения переменных x_i обычно не превышают 3, что, однако, не делает бессмысленным разработку быстродействующих алгоритмов решения задачи, так как к этой же модели могут сводиться другие задачи, в которых эти значения могут быть существенно больше. Кроме того, как уже отмечалось, возможно наличие нескольких ограничений вида (4). В этом случае, как показано в работе [1], задача сводится к многократному решению задачи с одним ограничением.

Задача № 2. Выбор оптимальной комплектации

Рассмотрим ту же задачу обеспечения надежности системы из N элементов, но не путем резервирования, а за счет выбора наилучшей комплектации.

Итак, пусть имеется возможность выбора каждого элемента x_i системы из нескольких вариантов его изготовления (приобретения). Каждому варианту соответствует заданная вероятность отказа $p(x_i)$, стоимость (вес, объем, энергопотребление) или другая характеристика $f_i(x_i)$. Требуется подобрать такую комплектацию системы, чтобы вероятность ее отказа была минимальна, а суммарная стоимость не превышала заданной величины R.

Другими словами: найти минимум $P(x)$, где $x(x_1, x_2, ..., x_N)$ — неизвестный вектор, компоненты которого $x_i = 1, 2, ..., X_i$ удовлетворяют ограничению

$$\sum_{i=1}^{N} f_i(x_i) \leqslant R.$$

Целевая функция имеет вид

$$P(x) = 1 - \prod_{i=1}^{N} (1 - p_i(x_i)). \qquad (5)$$

Задача может быть решена по методу динамического программирования с использованием множеств Парето [4]. Если предположить, что все $p_i(x_i) \ll 1$ и пренебречь произведением вероятностей в формуле (5), то получим задачу (1), (2) с целевой функцией

$$P(x) = \sum_{i=1}^{N} p_i(x_i).$$

Эта задача может решаться, как и задача (1), (2), с использованием комбинированного алгоритма, в котором для определения нижней и верхней границ оптимума применяется новый алгоритм спуска — подъема.

Анализируя задачу (1), (2), видим, что перемена местами функций $f_i(x_i)$ и $g(x_{i*})$ не меняет математической сущности задачи. Поэтому рассмотренные задачи на минимум вероятности отказа системы при ограничении на ее стоимость (вес, объем и т. д.) могут быть переформулированы как задачи минимизации стоимости (веса, объема и т. д.) при обеспечении заданной надежности. В любом случае задача рассматривается как двухкритериальная и ведется пошаговое построение одних и тех же паретовских множеств с последующим удалением части бесперспективных паретовских точек.

Список литературы

1. **Алексеев О. Г.** Комплексное применение методов дискретной оптимизации. М.: Наука, 1987.
2. **Струченков В. И.** Устаревшие стереотипы и новые алгоритмы решения прикладных задач дискретной оптимизации // Информационные технологии. 2012. № 5. С. 20—29
3. **Struchenkov V. I.** Combined Algorithms of Optimal Resource Allocation // Applied Mathematics. Scientific Research Publishing. 2012. Vol. 3. N 1.
4. **Струченков В. И.** Динамическое программирование с использованием множеств Парето // Дискретный анализ и исследование операций. 2008. Т. 15. № 6.
5. **Беллман Р.** Динамическое программирование. М.: ИЛ, 1960.
6. **Беллман Р., Дрейфус С.** Прикладные задачи динамического программирования. М.: Наука, 1965.

Информация

ЖУРНАЛ В ЖУРНАЛЕ

НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ

№ 11
НОЯБРЬ
2012

УДК 004.032.26:004.048

А. В. Мышев, канд. физ.-мат. наук, доц.,
e-mail: mishev@iate.obninsk.ru,
Национальный исследовательский
ядерный университет МИФИ —
Обнинский институт атомной энергетики

Метрологическая теория динамики взаимодействущих объектов в информационном поле нейросети

В рамках информационной модели нейросети рассматривается метрологическая теория описания динамики взаимодействующих объектов в его информационном поле. Динамика процессов информационного взаимодействия объектов и среды такой нейросети является основным механизмом, который индуцирует информационное поле. Методология теории использует семиотический подход поиска решений ее основной проблемы: в чем и как проявляются закономерности взаимодействия, взаимовлияния и взаимосвязи сигнала и символа (образа) как физических и информационных сущностей?

Ключевые слова: *информационная динамика, информационная метрология, информационное поле, информационная и фрактальная связанность, информационный поток, информационная модель нейросети*

Введение

Метрологическая теория динамики взаимодействующих объектов в информационном поле искусственной нейросети является новой парадигмой в области разработки и реализации нейросетевых технологий для решения широкого класса практических и прикладных задач [1, 2]. В таком контексте необходимо отметить следующее: во-первых, основным элементом формальной нейросети (ФНС) в традиционном представлении является формальный нейрон, модель которого описывает зависимость выходного сигнала как взвешенной суммы входных сигналов. Такие модели, с одной стороны, формально задаются как цифровые фильтры, а их практическая реализация осуществляется в виде сумматора. А с другой — они никоим образом не позволяют описать и отразить как динамику информационных процессов ФНС и эволюцию ее состояний, так и характер и механизм информационного взаимодействия элементов сети. Во-вторых, если исходить из традиционных представлений о единой информационной основе нейросетей (как в нейробиологическом аспекте, так и кибернетическом, т. е. в символьном или логико-информационном), то ее образуют универсальный генетический код и универсальный алфавит.

Также известно из молекулярной биологии [3], что существует достаточно прямая зависимость между пространственной структурой белковых цепочек и генетическим кодом (символьной цепочкой), который они хранят. В свою очередь пространственная структура таких цепочек формируется за счет слабых взаимодействий между различными конфигурациями элементов, образующих цепочки. Белковые цепочки можно рассматривать как физические прототипы элементов различных типов памяти и как функциональные элементы интеллекта.

Функциональный потенциал интеллекта определяется пространственно-временной формой элементов его физического и биологического прототипов — белковых цепочек. А эта форма, в свою очередь, определяется последовательностью символов генетического кода конкретной белковой цепочки [4]. Связь между пространственной формой и функциями белковых цепочек реализуется посредством физико-энергетических и информационных механизмов, которые обеспечивают различные типы взаимодействия как между цепочками, так и между образующими элементами цепочки.

Если говорить о физической и информационной сущности "необозначенных" взаимодействий между белковыми цепочками (к каковым относятся нейронные структуры), то они являются более загадочными явлениями, чем взаимодействия в электромагнитных и гравитационных полях. Как возникают силы, порождающие различные типы энергетических и информационных взаимодействий между белковыми цепочками? Где они возникают? Почему они возникают? Как они распространяются? Как они действуют? На что они именно действуют? Является ли информация физической величиной?

В рамках обозначенной канвы вопросов будем использовать семиотический подход поиска ответов на некоторые из них, а именно: в чем и как проявляются закономерности динамики процессов информационного взаимодействия объектов и среды ФНС, которая (динамика) индуцирует информационное поле, как физических и информационных сущностей? Это необходимо, чтобы обозначить и/или показать характер их проявления в информационной динамике нейросетевых технологий как средствах искусственного интеллекта (ИИ) и атрибутов информационного поля нейросети, а также чтобы отразить и показать в чем же суть, смысл и содержание этих категорий в восприятии человеком информации, генерируемой посредством

обозначенных нейросетевых технологий. Построение логической схемы поиска решений обозначенных вопросов будем проводить в рамках формализма информационной модели нейросети [1].

Методология теории, с одной стороны, открывает возможности исследования физической сущности динамики процессов информационного взаимодействия в нейросетях в целях разработки интеллектуальных технологий измерений и вычислений и является новым развитием и продолжением работ в области квантовых измерений и вычислений [5—7]. А с другой стороны, открывает возможности создания способов математического описания информационной динамики процессов измерений и вычислений как динамики ограниченных слов, взаимодействующих с информационной средой в информационном поле [8, 9].

Методы информационной метрологии образуют такую систему логических регулятивов и правил, которые выступают в качестве интеллектуальных инструментов разработки моделей алгоритмов и процедур технологий измерений, обработки и анализа потоков информации с учетом динамической эволюции потока и его элементов в информационном поле. В обозначенных интеллектуальных системах, в информационной среде которых протекают динамические процессы измерений, обработки и анализа информации на логическом и физическом уровнях, присутствует и доминирует информационное поле, которое определяет характер информационной эволюции потока данных и взаимодействие его элементов. В каналах передачи и хранения информационных систем потоки данных логически структурированы и функционально организованы в пространственных и временных масштабах на уровне информационных квантов и образуют дискретные квантовые информационные пространства.

Элементами информационных квантов потока в каналах передачи являются символы алфавита источника (информационные кванты источника), цепочка которых образует посылку информации в канале. Множество таких посылок образуют поток информации. Посылки могут быть как постоянной, так и переменной длины. В практической и технической реализации длина посылки определяется программно-аппаратным пулингом взаимодействия источника и приемника информации.

В каналах хранения информационными квантами потока являются ячейки памяти (физические и виртуальные), наполнение и содержание которых также образует цепочку символов базового алфавита (битов) [10]. Потоки информации в каналах хранения и передачи представляют собой логически структурированные информационные объекты как на уровне всего потока, так и на уровне его элементов (посылок и ячеек памяти).

Информационная динамика потока проявляется как на уровне потока, так и на уровне его элементов. На уровне потока она проявляется в характере изменения количественной меры информации, которую несут его элементы, а на уровне посылок или ячеек памяти она проявляется двояко: в характере изменения символов в позициях ячейки или посылки; в характере изменения значений символов в позиции. Метамеханизм такой динамики можно описать в рамках парадигмы теории информационного поля, которое образует информационный поток.

Основные понятия и определения

В системе базовых понятий и определений онтологии предметной области информационного поля в качестве единицы измерения информации на уровне ее логического и физического прототипов выступает ***бит***. Как логический прототип информационной сущности бит определяется в виде символического образа конкретной логики, а как образ булевой логики обозначается в виде двух символов: 1 и 0. Физическими прототипами бита в каналах хранения информации выступают микрофизические системы — объекты, детектирование состояний которых идентифицирует и инициализирует символ логического прототипа. Например, в ОЗУ компьютера в качестве микрофизической системы—объекта может быть триггер или другой элемент, на жестких дисках — это область поверхности. Аналогично и на других носителях информации с другой физической средой с двумя состояниями значения шкалы физических состояний обозначенных систем—объектов будут соответствовать логическим прототипам: 1 или 0.

Булевы образы являются базовыми символами интеллектуальных информационных и вычислительных систем, которые комбинируются посредством исчисления высказываний исходя из некоторого множества простых признаков, являющихся двоичными переменными [11]. Физические прототипы бита в каналах передачи, а также и некоторых типах каналов хранения, является ***сигнал***. Что такое сигнал? Определений сигнала существует множество, но в силу логики изложения и целесообразности остановимся на следующем. ***Сигнал*** — это пространственно-временное изменение физической величины в физической среде.

В каналах информационных систем в качестве глобальных атрибутов информации на логическом и физическом уровне выступают информационные бинарные множества — это множества, элемента-

ми которых являются символы базового (булева) алфавита, посредством которых образуются, с одной стороны, логические структуры в информационном пространстве каналов памяти в виде сегментов, а с другой — потоки логически и функционально связанных данных в каналах передачи. Информационный объект системы рассматривается как бинарное множество, на котором определяются информационные пространства с заданной на них алгебраической и логической структурой [10].

Базовые понятия и определения онтологии предметной области обозначим и сформулируем следующим образом.

Информационный поток, с одной стороны, определяется как множество взаимодействующих цепочек символов конкретного алфавита переменной или/и фиксированной длины, которые образуют логически и параметрически связанную структуру данных (СД). А с другой — это множество массивов данных, которые объединены логической организацией данных в массиве, однородностью их отношений и содержания, а связанность массивов по параметрической переменной образует сложную регулярную СД. Модель информационного потока для первого определения задается в виде кортежа $<X, N_m>$, где X — множество цепочек символов алфавита N_m, которое образует поток, а N_m — множество бинарных цепочек длины m. Когда поток определен в виде сложной регулярной СД, то геометрия и топология потока описывается и формализуется в рамках информационной модели нейросети [1], матричным представлением такой сети будет матрица структурных чисел [12].

Информационная система координат задается как по "ширине", так и по "глубине" информационного потока и включает систему нумераций и информационную метрику. Схема построения информационной системы координат для потоков информации предполагает выполнение следующих шагов:

- определяется формальный язык описания информационного потока;
- определяются математическая структура, описывающая логическую организацию потока на уровне символов алфавита его модели и динамику цепочек взаимодействующих символов в потоке, внешняя и внутренняя системы нумераций;
- задается метрика на информационном пространстве потока;
- осуществляется выбор интегральных характеристик информационного потока: для первой модели в информационном пространстве $<X, N_m>$, а для второй — в дискретном квантовом информационном пространстве;
- проводится когнитивный анализ потока на основе интегральных характеристик в заданной информационной системе координат.

Система нумерации информационного потока включает нумерацию двух типов: внутренюю и внешнюю. Применительно к информационному потоку, представленному в информационной системе координат в виде двухмерной (а в общем случае многомерной) структуры, выделение двух типов нумераций заключается в следующем:

- *внешняя нумерация* отражает информационную параметрическую связь либо между символами алфавита N_m в информационном потоке по их позициям во всех словах потока, если поток определен в рамках первой модели (т. е. информационную динамику символов в позициях и обмен информацией между словами — связанность слов-посылок в потоке), либо между элементами регулярной СД в случае второй модели потока;
- *внутренняя нумерация* отражает структуру символов алфавита N_m относительно номеров их позиций в слове информационного потока — связанность символов в слове-посылке для первой модели и аналогично для второй модели, когда отражается информационная структура элементов регулярной СД относительно их номеров в матрице структурных чисел.

Информационная метрика определяет расстояние между элементами информационного потока x_i и x_{i+1} и вычисляется по формуле

$$d(x_i, x_{i+1}) = \sum_{i=1}^{L} w_i |a_i - b_i|, \qquad (1)$$

где L — длина символьных цепочек x_i и x_{i+1}; w_i — вес i-й позиции символьной цепочки; a_i и b_i i-е символы цепочек в соответствующих позициях X_i и X_{i+1}.

Параметрическая связь между элементами потока информации в применяемых методах информационной метрологии вводится и используется для структуризации элементов в потоке. Параметрические связи указывают направление переноса информации между отдельными элементами информационного потока.

Способы построения параметрической связи в потоках информации:

- *ассоциация* имеет место в том случае, если существует несколько связанных друг с другом элементов информационного потока, причем появление одного элемента закономерно влечет появление связанного с ним другого элемента.

Ассоциативные связи различают по типу:

1) ассоциации по смежности в информационном пространстве образуют параметрические свя-

зи между топологически близкими элементами, которые можно представить в виде единого информационного объекта;

2) ассоциации по смежности во времени образуют параметрические связи между близкими по времени или параметру (т. е. принадлежащие одному кванту времени или соответствующему ему информационному кванту) элементами потока, воспринимаемыми как единый информационный объект;

- *последовательность* определяется как упорядоченный набор связанных по параметрической переменной элементов информационного потока;
- *классификация* как способ построения параметрической связи отражает сгруппированные в соответствии с общими признаками элементы информационного потока.

Параметризация связей между элементами информационного потока также является процедурой построения логической организации в виде структуры данных и информационной привязки элементов потока к информационной системе координат, а также определяет связанность элементов между собой в этой системе.

Информационные и фрактальные центры связанности. В качестве интегральных характеристик информационного поля потока данных в информационной системе координат выбраны следующие атрибуты [9]:

- *информационные центры* являются "энергетическими" атрибутами информационного поля, посредством которых определяется "силовой" потенциал взаимодействия и описывается информационная динамика в потоке данных;
- *фрактальные (геометрические) центры* являются топологическими и геометрическими атрибутами информационного поля, посредством которых отражаются топологическая и геометрическая структуры потока данных в этом поле и описываются фрактальные свойства (степень нерегулярности) потока в метрическом информационном пространстве.

Основные положения теории

Информационный поток данных описывается и представляется в виде двухмерной структуры данных (например, в виде связанного списка или матрицы структурных чисел). На рис. 1 (см. третью сторону обложки) приведена геометрическая иллюстрация двухмерной структуры потока. В плоскости *XOY* эта структура определяется в виде решетки. Сетевой моделью потока на такой решетке является ориентированный граф, который однозначно описывает частичную упорядоченность информационного потока. Целочисленная ось *OX* задает направление и область определения параметрической переменной x (или дискретного квантового времени), относительно которой упорядочены все элементы потока или символы посылки. На целочисленной оси *OY*, на которой задается шкала изменения либо номеров позиций символов в каждой посылке потока, либо номеров информационных квантов в посылке, индексную переменную обозначим y. Смысл и содержание метрологии информационной динамики процессов взаимодействия в информационном поле нейросети состоит в том, чтобы выделить из потока данных наиболее информационно связанные элементы. Информационный потенциал в информационном пространстве потока определяет характер взаимодействия между элементами потока и взаимодействие элементов потока с информационной средой. Схему метрологии информационного взаимодействия элементов потока данных в информационном поле можно обозначить в виде следующего алгоритма:

- во-первых, определяется фрактальная и информационная связанность элементов в пределах столбца размерности M (связанность по вертикали — локальная связанность);
- во-вторых, определяется фрактальная связанность всех элементов матрицы (связанности по горизонтали — в направлении увеличения параметрической переменной) и выделяется топологическая инкарнация матрицы в виде отдельной строки размерности N;
- в-третьих, определяются статистические характеристики (математическое ожидание и дисперсия) для каждого столбца, множество значений которых образует топологическую инкарнацию по горизонтали (статистическая связанность);
- в-четвертых, из всех полученных инкарнаций выбирается та, которая удовлетворяет условию выбора. В качестве условия выбора используется положение о минимальности фрактальной связанности и максимальности информационной связанности.

Сетевая модель потока информации в информационной модели нейросети [1, 12] задается в виде некоторой двухмерной матрицы структурных чисел (в общем случае многомерной), элементы которой содержат информацию о вероятностных, топологических и геометрических характеристиках элементов потока. С одной стороны, такую матрицу можно интерпретировать как дискретную случайную функцию от параметрической переменной x в направлении оси *OX*. Тогда изменение переменной y по оси *OY* задает отношение порядка либо номеров позиций символов в цепочке относительно столбца матрицы, если посылка потока опреде-

ляется как цепочка взаимодействующих символов, либо номеров элементов одномерного массива относительно столбца матрицы, когда посылка информационного потока определена в виде сложной СД. С другой стороны, математический аппарат матриц структурных чисел для формализации и построения сетевых моделей информационных потоков со сложной логической организацией и структурой связей является расширением и развитием теории случайных функций для изучения динамики процессов взаимодействия элементов информационного потока в его информационном поле. В этом случае рассматривается динамика процессов как по вертикали относительно элементов соседних столбцов, так и по горизонтали относительно элементов строк. Основная посылка раскрытия смысла динамики процессов по вертикали состоит в том, чтобы определить и выделить те элементы соседних i-го и $(i + 1)$-го столбцов, начиная с первого, которые наиболее связаны между собой. Так как ненулевых элементов в столбцах может оказаться более одного, то связанность (смысл этого понятия будет более содержательным и понятным по мере изложения материала) рассчитывается относительно каждого ненулевого элемента i-го столбца во взаимодействии его со всеми ненулевыми элементами $(i + 1)$-го столбца. Таким образом, среди всех ненулевых элементов каждого столбца выделяем наиболее связанные (связанность по "вертикали"). В результате такой процедуры получаем новую матрицу, состоящую из столбцов, элементы которых наиболее связаны. В этом случае динамика процессов взаимодействия элементов в потоке определяется как динамика с локальным взаимодействием.

Динамика взаимодействия процессов по горизонтали проявляется в формировании строк матрицы, элементами которых являются элементы ранее сформированных столбцов, т. е. такая строка представляет собой топологическую инкарнацию на множестве ненулевых элементов новой матрицы (ранее образованной — предыдущий шаг) в виде строки. Цель такой процедуры и основная посылка заключается в том, чтобы выделить те возможные траектории динамики процессов информационного взаимодействия элементов матрицы, которые связывают все активные элементы первого и последнего столбцов потока. Последним шагом обозначенной процедуры является выделение из множества топологических инкарнаций (связанность по "горизонтали") такой, которая удовлетворяет условию выбора (минимальная фрактальная связанность и максимальная информационная связанность). Такую сетевую модель можно рассматривать как новое направление построения информационных фильтров на основе нейросетевых технологий.

Характеристики информационного потока

Фрактальная размерность точечных подмножеств матрицы как информационного прототипа нейросети определяется по следующей формуле:

$$d_f = \lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log \varepsilon}, \qquad (2)$$

где $N(\varepsilon)$ — число покрытий в пределах границ области значений по оси OY ненулевых элементов в строке или столбце матрицы информационного потока; ε — расстояние между элементами потока. Емкостная фрактальная размерность d_f вводится и определяется как мера нерегулярности топологической и геометрической структур потока в целом или его частей.

Для оценки информационной фрактальной размерности потока данных автором предложена и использовалась информационная B-энтропия, определяемая выражением

$$B(\varepsilon) = -\sum_{i=1}^{K} p_i \log \sum_{i=1}^{K} (1 - \rho_{ij}) p_j, \qquad (3)$$

где $1 \leqslant K \leqslant \max(N, M)$ — число элементов строки или столбца информационного потока или его подмножества, относительно которых оценивается информационная размерность; ρ_{ij} — рандомизированная метрика между i-м и j-м элементами строки или столбца в метрическом вероятностном пространстве (S^*, S_a, P, ρ), где S^* — пространство элементарных исходов; S_a — алгебра событий на S^*, P — вероятностная мера на S_a, $0 \leqslant \rho \leqslant 1$ — рандомизированная метрика.

Метрика ρ_{ij} отражает геометрические свойства между элементами потока в строке или столбце и определяется следующим выражением:

$$\rho_{ij} = \frac{|r_i - r_j|}{|r|}, \qquad (4)$$

где $|r_i - r_j|$ — это расстояние (геометрическое или информационное) между i-м и j-м элементами строки или столбца относительно начала; $|r|$ — модуль длины строки или столбца.

При малых ε информационная B-энтропия будет вести себя как

$$B(\varepsilon) = d_b \log(1/\varepsilon), \qquad (5)$$

тогда для малых ε информационную фрактальную размерность можно определить как

$$d_b = \lim_{\varepsilon \to 0} \frac{B(\varepsilon)}{\log(1/\varepsilon)} = \lim_{\varepsilon \to 0} \frac{\sum_{i=1}^{K} p_i \log \sum_{j=1}^{K} (1 - \rho_{ij}) p_j}{\log \varepsilon}. \quad (6)$$

Информационная связанность в *i*-й строке или столбце потока определяется следующим выражением:

$$I_i = \frac{\sum_{j=1}^{K} p_i r_j}{\sum_{j=1}^{K'} p_i}, \quad (7)$$

где p_{ij} — вероятностная мера *j*-го элемента потока в *i*-й строке или столбце; r_j — информационное расстояние *j*-го элемента в строке или столбце; вероятности p_{ij} могут быть нормированы как по строке или столбцу, так и по всему потоку ($K' = N$ или M, $K' = NM$).

Условие выбора информационного центра в *i*-м столбце или строке задается следующим образом:

$$I_i \to \max_i. \quad (8)$$

Фрактальная связанность в *k*-й строке или столбце вычисляется по следующей формуле:

$$D_k = \frac{\sum_{j=1}^{K} \varepsilon_j r_j}{\sum_{j=1}^{K} \varepsilon_i}, \quad (9)$$

где r_j — геометрическое расстояние *j*-го элемента относительно начала строки или столбца; $\varepsilon_j = |r_{j+1} - r_j|$.

Условием выбора фрактального центра будет

$$D_k \to \min_k. \quad (10)$$

Информационная и фрактальная связанности являются математическими атрибутами, посредством которых задается и описывается потенциал взаимодействия информационного поля потока данных (по аналогии с гравитационным потенциалом). Формулы (7) и (9) определяют аналитический вид потенциала поля. Потенциал разделяется на две составляющие: информационная составляющая (формула (7)) и геометрическая составляющая (формула (9)).

В качестве статистических характеристик для элементов строки и столбца используются математическое ожидание и дисперсия.

Математическое ожидание дискретного множества элементов в *i*-й строке или столбце потока определяется следующим выражением:

$$M_i[y] = \sum_{j=1}^{K} y_j p_{ij}, \quad (11)$$

где символ *y* обозначает переменную для значений элементов *i*-й строки или столбца потока, p_{ij} — вероятность элемента в потоке.

Дисперсия определяется формулой

$$D_i[y] = \sum_{j=1}^{K} (y_j - M_i)^2 p_{ij}. \quad (12)$$

В онтологии предметной области и методологии теории важное значение имеют понятия информационного и фрактального (геометрического) центра. Эти центры как математические атрибуты позволяют описать динамическую эволюцию потока данных в информационном поле нейросети с информационной и топологической точки зрения, как размытый информационный процесс в условиях модельной замкнутости, ограничений, обмена и неопределенности. А характер такой динамики в информационном поле потока данных с потенциалом поля, задаваемым формулами (7) и (9), можно определить (описать) как перемежаемость в размытой информационной среде.

Динамическая модель процесса взаимодействия объектов в информационном поле нейросети

Информационная модель ФНС формально, логически и структурно определяет и описывает ее (нейросеть) как локальную виртуальную информационную динамическую систему [1]. Пространственная структура элементов, топология связей, логическая и информационная организация в такой динамической системе образуют среду (физическую и виртуальную) формирования искусственной интеллектуальной нейросети с динамически изменяемой и реконфигурируемой архитектурой, базовыми элементами и компонентами которой на логическом и физическом уровнях являются:

- каналы передачи и хранения информации;
- процессорные элементы, элементы детектирования и восприятия.

Каналы передачи в такой сети на логическом и физическом уровнях образуют "нити" информационного взаимодействия между элементами, а именно: "транспортировка" и управление в сети. Каналы хранения образуют ту физическую среду, в которой, с одной стороны, реализуется модель активной виртуальной памяти, а с другой — формируется виртуальное информационное пространство, в ко-

тором индуцируется информационное поле. Процессорные элементы являются "генераторами" таких потенциалов информационных взаимодействий, посредством которых индуцируется и формируется информационное поле динамически изменяемого потока информации. Элементы детектирования и восприятия выполняют функции приема и отражения информации как на уровне сигналов, так и на уровне символов и более сложных образов.

Логическая схема динамической модели процессов взаимодействия в информационном поле нейросети, которое индуцирует и порождает информационный поток, представляющий последовательность цепочек взаимодействующих символов, в реальных измерительных и вычислительных системах может быть описана следующим образом.

Прежде всего, следует отметить, что в моделях алгоритмов и процедур информационных технологий приема, передачи, обработки, анализа информации интеллектуальных систем измерений, телекоммуникаций, детектирования, восприятия или функциональных и логических схем процессорных элементов вычислительных систем в качестве входного потока информации могут выступать несколько наборов данных, имеющие различную точность измерения, обусловленную информационной неопределенностью, точностью измерительных приборов и т. д. Каждый элемент данных имеет определенную точность, выраженную числом значащих позиций, и представлен в виде цепочки взаимодействующих символов, длина которой равна числу значащих позиций. Каждой позиции сопоставлен ее вес, а алгоритм назначения весов позициям может быть построен следующим образом: "младшие" позиции имеют больший вес, "старшие" позиции имеют меньший вес, сумма весов позиций должна быть нормирована на конкретную величину. В простейшем случае вес позиции обратно пропорционален номеру позиции при условии, что нумерация позиций идет, начиная с "младших" позиций. Информационная динамика процессов обработки и анализа в этом случае будет проявляться и отражаться в том, как происходит эволюция центров связанности информационного поля потока данных относительно позиций символов и относительно значений символов в позициях.

Логическая схема (или модель) алгоритмов и процедур исследования информационной динамики процессов взаимодействия в информационном поле потока для обозначенной его структуры включает следующие шаги:

— оценить динамику значений символов по позициям элементов данных для каждого набора данных;

— определить информационные и фрактальные центры относительно позиций для каждого элемента потока данных;

— определить информационные центры относительно значения символа в позиции для каждой позиции по всем элементам потока;

— получить числовые оценки разброса значений символов относительно центров;

— определить относительные частоты появления центров в каждой позиции символьной цепочки.

Так как элементы потока данных имеют различную точность, соответственно они представлены в виде символьных цепочек различной длины. Для того чтобы была возможность сравнивать элементы данных нескольких потоков или в рамках одного потока, необходимо привести группу взаимодействующих символьных цепочек к одной длине. Соответственно, более короткие символьные цепочки необходимо дополнить символами до длинных символьных цепочек. Чтобы добавляемые значения не вносили существенных изменений в структуру данных, необходимо "сместить" их веса за пределы символьной цепочки (в сторону информационной среды справа). На рис. 2 (см. третью сторону обложки) приведена геометрическая иллюстрация расположения символьных цепочек разной длины относительно друг друга и информационной среды.

Приведенные к одинаковой длине символьные цепочки образуют информационный поток, представляющий собой некоторую двухмерную матрицу из $N \cdot M$ элементов, которая имеет две системы нумерации. Следует отметить, что веса последних символов короткой цепочки меньше, чем веса аналогичных символов более длинной цепочки. Один индекс матрицы обозначает внешнюю нумерацию, а другой индекс — внутреннюю нумерацию. Внешняя нумерация определяет отношение порядка по индексу, выбранному в качестве параметра, и состоит в том, что столбцы матрицы содержат структурированные по параметру данные. Внутренняя нумерация для фиксированного внешнего индекса определяет информационную связанность значений элементов подмножества матрицы и отражает структуру данных столбца, не зависящих от параметра и обусловленных информационной неопределенностью.

Тогда решение обозначенной динамической задачи заключается в выделении из матрицы наиболее связанных элементов, которые образуют информационную траекторию потока. Для этого определяются фрактальная и информационная связанность элементов по столбцам и строкам, а затем рассчитываются их фрактальные и информационные центры.

Динамическая эволюция значений элементов информационного потока относительно позиций позволяет получить оценку регулярности информационной структуры потока данных, а также определить "плохие" данные (выпадающие из общей структуры). Такую динамическую модель процессов взаимодействия и эволюции информационного потока можно также рассматривать как прототип моделей дискретных информационных фильтров нового поколения.

О практической реализации динамической модели процессов взаимодействия объектов в информационном поле нейросети

В качестве полигона исследования и практической реализации динамической модели процессов взаимодействия объектов в информационном поле нейросети были выбраны потоки динамических данных спутниковых телеметрических измерений, передаваемых по информационному каналу системы GPS (см. таблицу), который сопрягался через аппаратный модуль системы с компьютером. В результате эксплуатации модуля в астрономической обсерватории Киевского университета обнаружилось, что модуль не гарантирует требуемой точности — девятая цифра из трех последних не была значащей. Обращение к службе технического сопровождения системы на предмет метрологической аттестации модуля положительного результата не дали — модуль прошел успешно аппаратную метрологическую проверку и образовалась тупиковая ситуация. Это обстоятельство мотивировало и инициировало постановку обратной задачи, решение которой в рамках научного сотрудничества с астрономической обсерваторией Киевского университета явилось объектом практической реализации разрабатываемой автором метрологической теории динамики процессов взаимодействия в информационном поле потока данных. Практическая суть задачи и способ ее решения заключаются в следующем.

Данные телеметрических измерений представляют поток динамических данных, в котором базовыми элементами являются: Ephemeris — предполагаемый момент наблюдения и Observation — зарегистрированный момент астрономического явления. Цепочки символов для моментов регистрации характеризуются различной точностью (элементы данных представлены различным числом значащих цифр — от 6 и 9 символов). Для каждого элемента из двух наборов данных приведены оценки точности для предполагаемых моментов наблюдений и моментов регистрации астрономических явлений. Решение задачи в обозначенной постановке состоит в том, чтобы, используя методы информационной метрологии, проверить соответствие точности регистрации моментов заявленным значениям, т. е. определить и измерить динамику значений символов в старших значащих позициях (7-, 8-, 9-я позиции) результатов регистрации моментов астрономических явлений и предполагаемых моментов наблюдений и тем самым проверить правильность и надежность передачи информации по каналу GPS.

Среди данных спутниковых телеметрических измерений практический интерес для проверки надежности передачи информации по каналу GPS представляют значения предполагаемых и зарегистрированных моментов времени астрономических явлений. Логическая схема и алгоритм решения обозначенной задачи на основе методов и технологий информационной метрологии включает выполнение следующих шагов:

- проверка заявленной точности регистрации моментов астрономических явлений;

Логическая организация потока данных спутниковых телеметрических измерений, передаваемых по каналу связи GPS (фрагмент)

Date	Ephemeris	Object	m	Event	Telescope	Observation
18.01.2005	21 50 54.	DM +17457	5.8	DD	NM	15 05 59.104
19.04.2005	20 08 33.	SAO 99185	7.9	DD	NM	20 08 28.878
12.05.2005	18 19 15.	SAO 78862	8.6	DD	RV	18 19 14.508
12.05.2005	18 39 31.	SAO 78873	7.8	DD	RV	18 39 30.484
12.05.2005	18 40 54.	SAO 78876	7.2	DD	RV	18 40 56.434
15.05.2005	19 13 59.	SAO 28603	8.9	DD	RV	19 13 58.095
15.05.2005	21 34 13.	SAO 98640	8	DD	RV	21 34 15.703
16.06.2005	19 44 20.	SAO 139071	7.8	DD	RV	19 44 19.346
12.07.2005	18 46 25.		7.4	DD	RV	18 46 22.571

Примечание: Date — дата наблюдения; Ephemeris — предполагаемый момент наблюдения, который дается с точностью до ±5"; Object — номер звезды за каталогом, m — звездная величина звезды, Event — явление (DD, DB — покрытие темным(светлым) лимбом Луны, RD, RB — покрытие темным, светлым лимбом Луны, DU — покрытие звезды Луной во время затмения); Telescope — RV телескоп рефрактор, NM — телескоп рефлектор; Observation — зарегистрированный момент явления.

- определение динамики значений в старших позициях моментов регистрации астрономических явлений для обоих наборов данных (предполагаемые моменты времени наблюдений и зарегистрированные моменты времени);
- определение связанности между соответствующими предполагаемыми и регистрируемыми моментами времени астрономических явлений;
- определение характеристик расхождения (несоответствия) предполагаемого и наблюдаемого моментов времени астрономических явлений;
- определение связанности между позициями моментов времени астрономических явлений;
- определение информационных, топологических и статистических характеристик предполагаемых и регистрируемых моментов астрономических явлений;
- проведение когнитивного анализа полученных результатов.

Решение исходной задачи на основе методов и технологий информационной метрологии динамики процессов взаимодействий в информационном поле потока данных заключается в том, чтобы получить оценку регулярности динамики символов цепочки в старших значащих позициях результатов регистрации астрономических явлений и определить число значащих позиций, в которых проявляется нерегулярность, и сопоставить ее с заявленной точностью регистрации моментов астрономических явлений.

Из таблицы данных спутниковых телеметрических измерений видно, что элементы набора предполагаемых моментов описываются шестью значащими цифрами при точности до +/− 5 с, а элементы набора наблюдаемых моментов — девятью значащими цифрами при точности измерения 7...10 мс. Исходные наборы данных логически структурированы и описаны в рамках первой модели информационного потока, элементами которого являются цепочки символов постоянной длины, равной числу значащих цифр элементов набора данных таблицы. По схеме, описанной в предыдущем разделе, для каждого символа цепочки определяется весовой коэффициент.

Алгоритм выполнения операций между символьными цепочками различной длины для обоих наборов также включает выполнение следующих шагов:

- более короткие символьные цепочки достраивались до более длинных за счет добавления символов из алфавита информационного потока;
- так как добавляемые символы могли быть любыми из алфавита информационного потока, а также для минимизации влияния добавляемых символов, их весовые коэффициенты "смещались" за пределы более длинной символьной цепочки в сторону информационной среды справа.

Схема анализа передачи потока данных спутниковых телеметрических измерений, передаваемых по каналу связи GPS, интерпретировала входной поток как совокупность трех наборов:

- набор значений предполагаемых моментов астрономических явлений;
- набор значений регистрируемых моментов астрономических явлений;
- набор значений разности (расхождения) моментов.

Значения моментов регистрации астрономических явлений представлены в виде символьных цепочек постоянной длины, равной числу значащих цифр. Нумерация позиций символов начинается со старших значащих цифр. Каждой позиции назначен вес в соответствии с правилом назначения весов, которое иллюстрирует рис. 2 (см. третью сторону обложки). Для каждого элемента (символьной цепочки) из трех наборов рассчитаны фрактальные и информационные центры, распределение которых в фрагменте по позициям представлены на рис. 3 (см. третью сторону обложки).

Результаты обработки и анализа передаваемого потока астрономических данных, имеющего структуру, показанную в таблице, по информационному каналу GPS доказали возможность практической реализации методов информационной метрологии для диагностики и контроля на надежность информационных каналов не аппаратными средствами, а с использованием нейросетевых технологий интеллектуального анализа. В качестве частных выводов результатов проведенной метрологической проверки передачи информации по каналу GPS для рассматриваемого потока данных можно выделить следующие:

- информационный канал передачи GPS не обеспечивает заявленную точность в 9-м знаковом разряде цепочки данных (рис. 3, таблица справа), здесь динамика процесса взаимодействия отсутствует, а проявляется феномен информационной перемежаемости в размытой среде;
- динамика процессов взаимодействия в информационном поле потока имеет ярко выраженный характер в 6-й, 7-й, 8-й позициях цепочек данных (рис. 3, таблица справа);
- технологии и средства передачи информации по каналу GPS либо не имеют средств защиты и контроля информации от всевозможных помех, либо проявляют свойства зашумления канала ложной информацией (обнаружение выхода информационных и фрактальных центров за граничные позиции символов цепочки данных — позиция 4, рис. 3, таблица справа, строка 11).

Были проведены также более глубокие исследования на предмет применения разработанных приложений и технологий для алгоритмов и процедур вычислительных технологий [13]. Результаты показали, что метрологические аспекты информационной динамики процессов взаимодействия в информационном поле как потока данных, так и среды вычислений для вычислительных технологий и информационных технологий обработки, передачи, контроля и анализа результатов измерений и вычислений обусловлены наличием следующих факторов. Во-первых, информационная динамика процессов взаимодействия в среде вычислений и технологиях обработки и анализа информации в каналах передачи и хранения систем измерений, телекоммуникаций, детектирования, восприятия и вычислений строго привязана к системе координат информационной привязки, чего в настоящее время не имеют ни одна реальная система измерений или наблюдений, а также вычислительные технологии. Во-вторых, необходимое условие контроля и диагностики информационной динамики процессов обработки и анализа информации в условиях ограничений, обмена, многофакторной неопределенности — это наличие интеллектуальных систем и технологий метрологической проверки и поверки измеряемых или вычисляемых потоков данных.

Заключение

Рассмотренная метрологическая теория исследования динамики процессов взаимодействия в информационном поле ФНС отчасти отражает и определяет новые основы формирования наших представлений и понятий об информационном поле нейросети, интеллектуальном потенциале взаимодействия в пространстве этого поля, о механизмах формирования гиппокампальной памяти с доминантной архитектурой, о принципах организации и реализации связи внимания с восприятием, памятью и когнитивными функциями, о синергетических процессах коллективного воздействия компонентов нейросети, о природе и формах проявления закономерностей взаимосвязи и взаимовлияния сигнала и символа (образа) как базовых атрибутов интеллектуальной системы и многом другом. Следует отметить, что в рамках методологии теории многие проблемы искусственного интеллекта формализуются и описываются на основе синергии методов и технологий нейробиологического и кибернетического подхода исследования динамики процессов взаимодействия в информационном поле нейросети, математические основы которых имеют междисциплинарный базис и в значительной степени инвариантны по отношению к средствам и способам отражения и представления предметной области.

Парадигма информационного поля ФНС во многом определяет, с одной стороны, новые подходы исследования и изучения эволюции нейросетей в динамическом и информационном аспекте, а с другой — в ней заложены новые принципы и схемы создания нейросетевых технологий в плане разработки и реализации систем искусственного интеллекта как в различных областях технических приложений, так и в системах виртуальной и когнитивной реальности информационных сетей нового поколения.

Это в свою очередь предполагает необходимость перехода к другому языку математического описания динамики процессов информационного взаимодействия и иному стилю восприятия информации и знаний в терминах информационного поля нейросети.

Список литературы

1. **Мышев А. В.** Информационная модель нейросети в технологиях вычислительного интеллекта и формах реализации компьютинга // Информационные технологии. 2012. № 1. С. 63—70.
2. **Мышев А. В.** Символы и сигналы в информационных моделях нейросети и нейрона // Сб. науч. тр. XIV Всероссийской научно-техн. конф. "Нейроинформатика—2012", 23—27 января 2012, Москва, Россия. Ч. 1. М.: НИЯУ МИФИ. 2012. С. 93—95.
3. **Альбертс Б., Брей Д., Льюис Дж.** Молекулярная биология клетки. В 3-х т. М.: Мир, 1994.
4. **Меклер Л. Б., Идилис Р. Г.** Общий стереохимический генетический код — путь к биотехнологиям и универсальной медицине 21-го века уже сегодня // Природа. 1995. № 5. С. 28—70.
5. **Кадомцев Б. Б.** Динамика и информация. М.: Изд. УФН, 2005. 398 с.
6. **Швингер Ю.** Квантовая кинематика и динамика. М.: Наука, 1992. 350 с.
7. **Валиев К. А.** Квантовые компьютеры и квантовые вычисления // УФН, 2005. 175 (1). С. 3—39.
8. **Мышев А. В.** Динамика информационных процессов в вычислительных технологиях компьютерного моделирования // Тр. ИСА РАН. М.: КРАСАНД. 2010. Т. 58. С. 137—147.
9. **Myshev A. V.** The metrological theory of information dynamics for intelligent technologies' processes of processing and analyzing information // Proc. of the 11th International Conference (18—20 May, Minsk, Republic of Belarus). Minsk: BSUIR, 2011. P. 370—374.
10. **Мышев А. В.** Модели активной памяти на бинарных полях в технологиях виртуализации каналов передачи и хранения информации // Программные продукты и системы. 2010. № 1. С. 54—58.
11. **Гренандер У.** Лекции по теории образов. Т. 1. М.: Мир, 1979. 383 с.
12. **Мышев А. В.** Теория компьютерного восприятия и технологии взаимодействия вычислительного интеллекта с виртуальной средой моделирования // Матер. 7-й междунар. научно-техн. конф. "Кибернетика и высокие технологии XXI века". Воронеж: ВТУ, 2006. С. 397—409.
13. **Мышев А. В., Иванов П. Г.** Информационная метрология в компьютерных технологиях обработки и анализа информации // Научная сессия МИФИ-2009. Тез. докл. в 3-х т. Т. 3. "Информационно-телекоммуникационные системы. Проблемы информационной безопасности в системах высшей школы. Экономика, инновации и управление". М.: МИФИ, 2009. С. 160—161.

УДК 004.932

К. А. Демиденков, аспирант,
e-mail: sdk@mail.by,
И. И. Мельников, аспирант,
И. А. Евсеенко, канд. техн. наук, доц.,
Белорусско-Российский университет, г. Могилев

Совершенствование способов определения плотности транспортного потока и его состава на базе искусственных нейронных сетей путем использования параллельных вычислений

Рассмотрена возможность применения искусственных нейронных сетей совместно с алгоритмами цифровой обработки видеоданных для создания автоматизированной системы определения плотности транспортного потока и его состава. Предложен способ совершенствования данной системы путем применения параллельных вычислений.

Ключевые слова: *нейронная сеть, параллельные вычисления, видеоизображение, цифровая обработка, плотность транспортного потока, видеопоток, автоматизированная система*

Введение

В последнее десятилетие автомобильный парк Республики Беларусь значительно увеличился. По данным Министерства транспорта и коммуникаций Республики Беларусь по состоянию на 01.01.2010 он насчитывал 3433 тыс. автотранспортных средств, из которых 83,6 % приходилось на автомобили [14]. За период с 2005 по 2009 гг. число автотранспортных средств возросло на 21 %, а уровень автомобилизации населения Беларуси повысился с 280 до 340 автотранспортных средств на 1 тыс. населения. Таким образом, увеличилась плотность транспортного потока. Однако пропускная способность дорог остается прежней, что приводит к образованию заторов и пробок. Во многих странах ЕС, Северной Америки и Юго-Восточной Азии, а также в Российской Федерации стоит подобная проблема, которая успешно решается путем разработки и внедрения интеллектуальных транспортных систем [1, 2] или автоматизированных систем управления дорожным движением (АСУДД) [4, 5, 7, 8]. Существует подобная система и в Республике Беларусь — АСУДД "АГАТ" [6]. Данные системы позволяют эффективно управлять дорожным движением, но нуждаются в непрерывном получении информации о плотности транспортного потока в реальном времени и ее идентификации (например, состав движения: грузовики, микроавтобусы, легковые автомобили и т. п.). Поэтому возникает необходимость обеспечения непрерывного наблюдения за движущимися объектами на различных участках дорог независимо от погодных условий и времени суток, что также является далеко не решенной проблемой [9].

Проблемы получения данных о составе транспортного потока и вычисления его характеристик

Сегодня в АСУДД для сбора информации активно применяются датчики (транспортные детекторы) различного типа: контактного, фотоэлектрического, ультразвукового, магнитного и т. д. [10]. Однако их использование может быть весьма дорогим и неэффективным из-за быстрого износа, узкого угла обзора, низкой помехоустойчивости [3, 10]. К тому же датчики и программное обеспечение к ним тесно связаны со всей АСУДД, что повышает трудоемкость при их установке и сопровождении. Поэтому в последнее время именно системы видеонаблюдения привлекают все больший интерес исследователей и разработчиков [3]. Это связано с тем, что подобные системы все более интенсивно внедряются в повседневную жизнь, включая транспорт, а видеокамеры становятся все более доступными. К тому же на базе систем видеонаблюдения можно разрабатывать независимые автоматизированные системы определения плотности транспортного потока и состава движения. Они могут использоваться отдельно от какой-либо АСУДД, что актуально для относительно небольших, но интенсивно растущих городов (в частности, областных центров Беларуси). Такие системы уже используются в Российской Федерации [13] и Украине [11], но в Республике Беларусь подобных систем нет, хотя необходимость в них существует в областных центрах.

Также следует подробнее рассмотреть уже упомянутую выше проблему непрерывного наблюдения за движущимися объектами на различных участках дорог независимо от погодных условий и времени суток. Анализ систем обработки видеоизображения и распознавания объектов на нем показал, что процесс обработки и анализа видеоизображения идентичен для всех подобных систем [12]. Он состоит из нескольких этапов:

- выделение переднего плана;

- выделение и классификация движущихся объектов;
- распознавание и описание движения найденных объектов.

На каждом этапе можно использовать различные алгоритмы, обладающие определенными достоинствами и недостатками. Например, на этапе выделения переднего плана можно использовать методы временной разности, оптического потока, вычитания фона и др. [12]. Методы вычитания фона достаточно эффективны, но ресурсоемки и несколько латентны в обновлении модели фона, алгоритмы оптического потока чувствительны к шуму. На последнем этапе, когда необходимо распознать, какое транспортное средство попало в зону наблюдения (и транспортное ли это средство вообще), может потребоваться применение искусственных нейронных сетей (далее нейронных сетей) разной топологии в зависимости от того, какие алгоритмы были использованы на предыдущих этапах и какого класса транспортное средство распознается. Проблема заключается в том, что при различных условиях внешней среды (например, меняющейся освещенности) необходимо использовать различные алгоритмы цифровой обработки видеоизображения.

Основная нагрузка при этом возлагается на методы этапа выделения переднего плана. От того, насколько точно выбранный метод позволяет отделить точки изображения, принадлежащие движущимся объектам, от точек статического фона, во многом зависит качество работы методов последующих этапов, а значит, и всей системы в целом [12]. Именно на методы первого этапа большое влияние оказывает внешняя среда, и они, как правило, самые ресурсоемкие во всей цепочке методов обработки видеоизображения и обнаружения движущихся объектов [12].

Проблему можно решать путем повышения устойчивости алгоритмов обработки видеоизображения за счет совместного использования различных их типов [12], однако это не решает ее полностью, так как ресурсоемкость этих алгоритмов увеличивает время обработки видеоизображения.

Еще одна проблема связана с этапом распознавания движущихся объектов. На данном этапе обычно используются нейронные сети, однако трудно создать и обучить нейронную сеть, которая бы с одинаково высокой точностью идентифицировала объекты самых разных классов. Например, что за движущийся объект обнаружен — транспортное средство или человек, если обнаружено транспортное средство, то какого оно класса (легковой автомобиль, грузовой автомобиль, автобус и т. п.) и т. д. Это увеличит ресурсоемкость системы и снизит ее быстродействие.

Учитывая значительный прогресс в области многоядерных и многопроцессорных систем, можно предложить новые пути решения этих проблем.

Таким образом, можно выделить следующие актуальные и тесно взаимосвязанные проблемы, стоящие перед разработчиками автоматизированной системы определения плотности транспортного потока и его состава.

1. В Республике Беларусь увеличивается число автомобилей, что приводит к увеличению интенсивности движения транспортного потока и более частому образованию заторов на дорогах крупных городов. Необходимо создавать АСУДД для эффективного управления им.

2. Существующие системы управления дорожным движением далеко не всегда используют эффективные средства для сбора информации о плотности транспортного потока и его составе, что вызывает интерес к использованию иных более эффективных средств — систем видеонаблюдения. К тому же не всегда необходимо разворачивать всю АСУДД. Достаточно обойтись использованием автоматизированной системы определения плотности транспортного потока и его состава на базе системы видеонаблюдения. На сегодняшний день подобных систем в Республике Беларусь нет.

3. Разрабатывая автоматизированную систему определения плотности транспортного потока и его состава на базе систем видеонаблюдения, требуется не только ориентироваться на уже существующие системы за рубежом, но и искать пути совершенствования их в целях обеспечения непрерывного наблюдения за движущимися объектами независимо от времени суток и погодных условий, снижения ресурсоемкости и увеличения быстродействия.

Автоматизированная система определения плотности транспортного потока и его состава на базе нейронных сетей

Разрабатываемая система состоит из трех модулей (рис. 1): детектор движущихся объектов, идентификатор транспортных средств, калькулятор плотности транспортного потока. В детекторе движущихся объектов происходит выделение переднего плана, выделение и классификация движущихся объектов. В идентификаторе транспортных средств происходит распознавание движущихся объектов на базе нейронных сетей. Нейронные сети уже успешно применяются в различных АСУДД для управления движением (например, адаптивные светофоры) и распознавания номерных знаков транспортных средств нарушителей правил дорожного движения. Калькулятор плотности транспортного потока рассчитывает плотность, используя инфор-

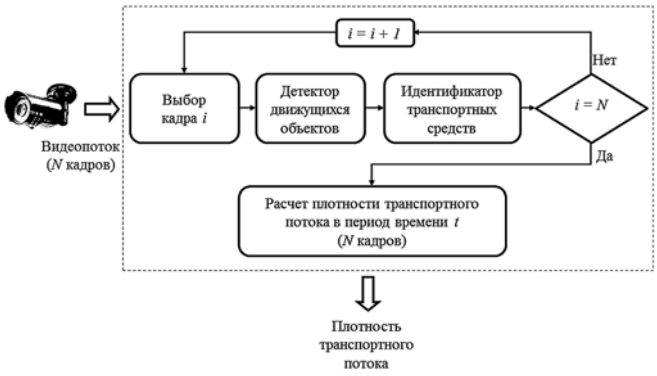


Рис. 1. Общая модель разрабатываемой системы

мацию о транспортных средствах, успешно распознанных идентификатором транспортных средств.

Далее будут рассматриваться только первые два модуля, которые представляют наибольший интерес, а под общей схемой работы системы будет подразумеваться общая схема работы именно этих двух модулей. Общая схема работы системы представлена на рис. 2.

Из рис. 2 видно, что видеокамера наблюдения передает данные о дорожной обстановке, которые последовательно проходят все этапы обработки, затем происходит распознавание выделенных движущихся объектов. При этом этапы выделения движущихся объектов и их классификации осуществляются детектором движущихся объектов, этап идентификации транспортного средства (другими словами, этап распознавания движущихся объектов) — идентификатором транспортных средств. И детектор движущихся объектов, и идентификатор транспортных средств представляют собой программные модули.

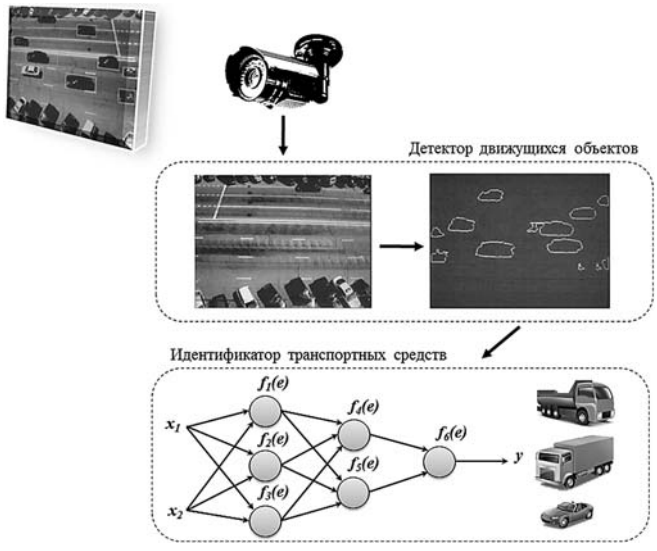


Рис. 2. Общая схема работы системы

Детектор оперирует набором алгоритмов, позволяющих отделить передний план от фона и выделить движущиеся объекты, например в виде контуров, рассчитав определенные характеристики каждого контура, представляющие собой числовые параметры (например, длина и ширина контура). Далее характеристики каждого контура передаются в идентификатор транспортных средств, в основе которого лежит заранее обученная нейронная сеть. Число входов нейронной сети равно числу заранее определенных характеристик контура (например, на рис. 2 показано два входа x_1 и x_2). Число выходов зависит от поставленной задачи. Если мы хотим определить, является ли выделенный движущийся объект транспортным средством, то достаточно одного выхода (на рис. 2 показан как y). Если же нам нужно определить класс транспортного средства, то необходимо столько выходов, сколько классов может опознать нейронная сеть. Сама нейронная сеть строится из отдельных искусственных нейронов, объединенных в слои (например, на рис. 2 их три), каждый из которых обладает определенной активационной функцией $f(e)$. Однако у такой схемы есть явные недостатки, о которых уже говорилось выше.

Способы улучшения автоматизированной системы определения плотности транспортного потока и его состава

Чтобы повысить быстродействие системы и степень независимости ее от изменяющихся факторов внешней среды, можно воспользоваться современными средствами распараллеливания процессов на базе многоядерных процессоров или видеокарт [15], так как развитие средств многопоточного и многопроцессорного программирования позволяет быстро и дешево организовывать высокопроизводительные и одновременно универсальные вычислительные мощности, заменяя, например, специализированные видеопроцессоры современными универсальными видеокартами, которые обладают достаточно высоким потенциалом в области параллельных вычислений.

Увеличить степень независимости системы от внешней среды можно путем увеличения устойчивости алгоритмов выделения переднего плана за счет совместного использования различных их типов, о чем уже говорилось выше. Однако это не увеличит быстродействие системы, а снизит ее. Ускорить работу на данном этапе можно путем разделения кадров видеоизображения на фрагменты, например по числу полос, за которыми ведется наблюдение (рис. 3). Каждый фрагмент будет обрабатываться отдельным процессом или потоком.

Нерешенной остается проблема, которая возникает на этапе распознавания движущихся объектов. В случае, когда число транспортных средств, попадающих в зону наблюдения за короткий промежуток времени, велико, нейронной сети приходится быстро распознавать огромное число движущихся объектов, это приведет к снижению быстродействия системы и все большему потреблению ресурсов. Также возникает проблема с обучением такой сети. Оно может занять достаточно долгое время.

Есть возможность ускорить работу системы путем применения распараллеливания с использованием многопроцессорных систем. Можно поступить следующим образом: распределить задачу на множестве нейронных сетей. Для этого каждую нейронную сеть нужно обучить отличать ограниченный набор движущихся объектов. Условно говоря, каждая нейронная сеть станет "экспертом" по своей группе объектов (легковые автомобили, грузовые автомобили, автобусы и т. д.). Каждая такая сеть будет функционировать в рамках отдельного процесса или потока.

Данный подход уже успешно используется для статических изображений, в частности, для ускорения процесса распознавания иероглифов [16] (рис. 4).

Распределением данных между процессами занимается особый выделенный процесс, который условно можно назвать главным. После распределения данных каждый процесс выполняет работу над своим блоком данных и затем отсылает результат главному процессу, который обобщает все принятые данные и выносит решение о результате распознавания. В качестве результата распознавания главный процесс должен выбрать результат той нейронной сети, которая наиболее уверена в своем ответе. В данной схеме каждая отдельная нейронная сеть будет иметь гораздо более простую структуру, чем одна универсальная нейронная сеть, ее можно будет быстрее обучить, она будет быстрее давать результат высокой точности.

Общая схема работы системы, при которой процесс цифровой обработки видеоизображения и распознавания объектов распараллелен, показан на рис. 5. Кадр, приходящий с видеокамеры, разбивается главным процессом или потоком, например на два фрагмента. Эти фрагменты передаются двум постоянно работающим процессам (потокам), которые параллельно осуществляют цифровую обработку видеоизображения. На выходе каждого из этих процессов (потоков) будет множество объектов с определенным набором характеристик. Для каждого объекта будет порожден новый процесс (поток) для его распознавания. Он, в свою очередь, распараллелит задачу распознавания объекта на

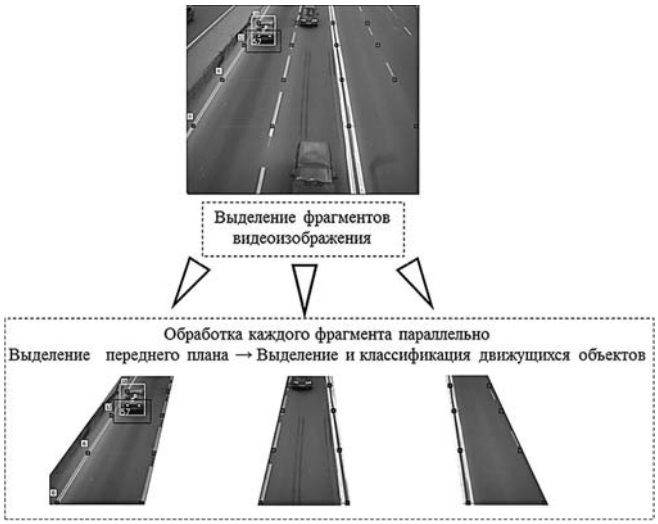


Рис. 3. Разбиение кадра на фрагменты для ускорения цифровой обработки видеоизображения

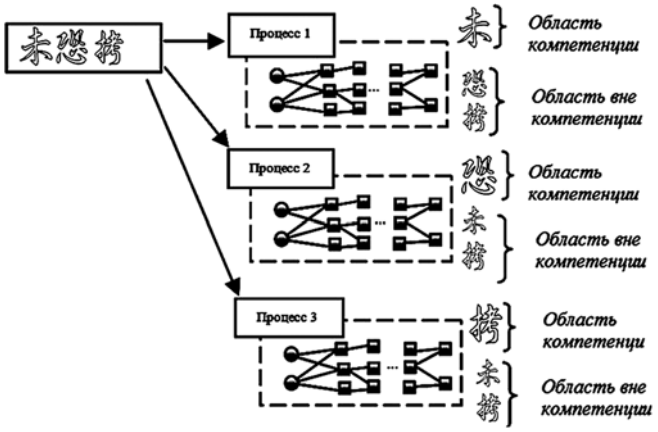


Рис. 4. Распределение образов по нейронным сетям путем распараллеливания на примере задачи распознавания трех иероглифов

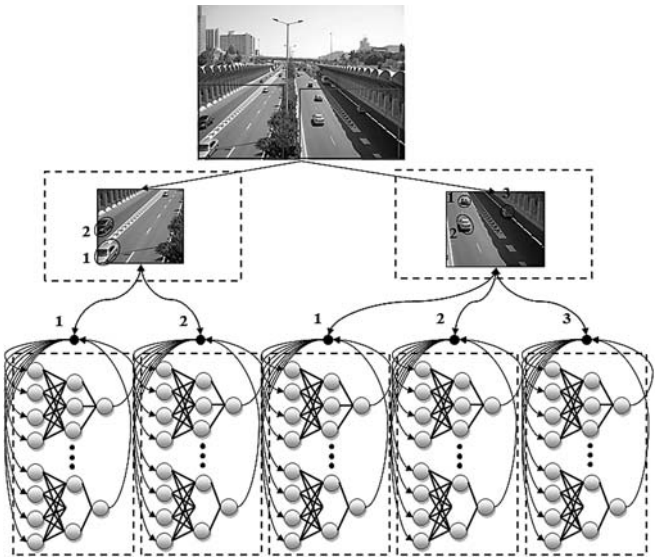


Рис. 5. Усовершенствованная схема работы разрабатываемой системы

множестве нейронных сетей-экспертов, что приведет к порождению новых параллельных процессов (потоков). Каждая нейронная сеть, проанализировав характеристики выделенного объекта, вернет процессу (потоку), который распределил задачу, значение вероятности того, что он принадлежит к транспортному средству определенного класса. Этот процесс (поток) по совокупности информации от всех нейронных сетей примет решение о том, какого класса является идентифицируемое транспортное средство, и передаст данные об этом процессу (потоку), его породившему, т. е. процессу (потоку) обработки видеоизображения.

Заключение

В статье рассмотрена возможность применения искусственных нейронных сетей совместно с алгоритмами цифровой обработки видеоданных для создания автоматизированной системы определения плотности транспортного потока и его состава. Предложен способ совершенствования данной системы путем применения параллельных вычислений.

Разрабатываемая система позволит увеличить эффективность управления дорожным движением в растущих городах. При этом она будет обладать низкой трудоемкостью по установке и сопровождению. Также возможно использование данной системы как модуля в рамках существующих АСУДД.

Список литературы

1. **About ITS** [Электронный ресурс] / RITA U. S. Department of Transportation. URL: http://www.its.dot.gov/its_program/about_ts.htm (дата обращения 29.09.2011).
2. **Ezell S.** Explaining International IT Application Leadership: Intelligent Transportation Systems. Washington: ITIF, 2010. 54 p.
3. **Ozkurt C., Camci F.** Automatic Traffic Density Estimation and Vehicle Classification for Traffic Surveillance Systems Using Neural Networks // Mathematical and Computational Applications. ASR, Manisa, Turkey, 2009. Vol. 14. N 3. P. 187—196.
4. **Автоматизированная** система управления дорожным движением [Электронный ресурс]. Институт системотехники. URL: http://www.omsis.ru/main.php?id=31 (дата доступа 10.09.2011 г.).
5. **Автоматизированная** система управления дорожным движением (АСУДД) [Электронный ресурс]. ГК "Спецтехника". URL: http://www.kb-spectech.ru/projectsl.html (дата доступа 23.09.2011 г.).
6. **Автоматизированная** система управления дорожным движением "Агат" (АСУДД "Агат") [Электронный ресурс]. АГАТ — системы управления. URL: http://www.agat.by/products/transport/dd/asudd-agat/ (дата доступа 25.03.2012 г.).
7. **Автоматизированная** система управления дорожным движением "Город" [Электронный ресурс]. Techno: группа компаний. URL: http://transport.grouptechno.ru/solutions/asudd-city/ (дата доступа 24.09.2011 г.).
8. **Автоматизированная** система управления дорожным движением "Магистраль" [Электронный ресурс]. Techno: группа компаний. URL: http://transport.grouptechno.ru/solutions/asudd-magistral/ (дата доступа 24.09.2011 г.).
9. **Галиакберов Р. А.** Алгоритмы отслеживания объектов в видеопотоках на параллельных вычислительных системах: Автореф. дис. маг. тех. наук: 230105.65. Донецк: Донецкий национальный технический университет. 2011. 30 с.
10. **Кожевников В. И., Вытяжков Д. В., Толмачев В. В.** и др. Автоматизированная система управления дорожным движением // Вестник Самарского государственного университета. 1995. № 1 (6). С. 110—118.
11. **КОМКОН** Traffic Control Equipment [Электронный ресурс]. КОМКОН НПП "Система + Сервис". URL: http://komkon.ua/programs/tce.html (дата доступа 26.03.2012 г.).
12. **Лукьяница А. А., Шишкин А. Г.** Цифровая обработка видеоизображений. М.: "Ай-Эс-Эс Пресс", 2009. 518 с.
13. **Система** распознавания номеров и автоматической фото- и видеофиксации нарушений правил дорожного движения "VOCORD Traffic" [Электронный ресурс]. VOCORD Системы видеонаблюдения и аудиорегистрации. URL: http://www.vocord.ru/218/ (дата доступа 26.03.2011 г.).
14. **Состояние** окружающей среды Республики Беларусь: нац. доклад / М-во природ. ресур. и окружающей среды Республики Беларусь, Гос. науч. учр-ние "Ин-т природопользования Нац. академ. наук Беларуси". Минск: Белтаможсервис, 2010. 150 с.
15. **Что** такое CUDA? [Электронный ресурс]. CUDA Zone nVidia. URL: http://www.nvidia.ru/object/what_is_cuda_new_ru.html (дата обращения 01.15.2012 г.).
16. **Ясинский Ф. Н., Мочалов А. С.** Распознавание большого количества образов при помощи нейронных сетей с использованием многопроцессорных систем // Вестник Ивановского государственного энергетического университета. 2011. № 2. С. 85—87.

УДК 004.032.26

Н. С. Томашевич, вед. специалист,
e-mail: tom-n@mail.ru,
ООО "Александрит"

Алгоритм построения признаковой нейронной сети для задач обработки изображений на примере задачи распознавания букв

Предложен новый алгоритм построения нейронной сети для задач обработки изображений. Отличительной особенностью предлагаемого метода является построение признакового первого слоя в нейронной сети, т. е. каждый нейрон первого слоя реагирует на какой-то один признак входного изображения. Остальные слои сети группируют выходы нейронов первого слоя оптимальным для распознавания классов образом. Кроме того, предлагаемый алгоритм позволяет получить 100 %-ное распознавание на обучающей выборке. Показана эффективность данной методики для многоклассовых задач распознавания, таких как задача распознавания букв.

Ключевые слова: *нейронные сети, обработка изображений, распознавание букв*

Введение

В этой работе рассматривается задача распознавания печатных букв инвариантно к используемому шрифту и стилю написания.

Задача распознавания букв решается уже давно и с разной степенью эффективности [1—6]. На наш взгляд, одним из основных недостатков используемых методов (вероятностных нейронных сетей, нечеткой логики, скрытых Марковских моделей, клеточных нейронных сетей и т. д.) является использование в сети (модели) сразу всего изображения буквы.

Например, в обучающей выборке мы имеем два вида буквы "А" на рис. 1, *а*, *б*. Буква на рис. 1, *в* получена из букв на рис. 1, *а*, *б* объединением правой и левой частей и используется в тестовой выборке. В этом случае ни один из традиционных методов не распознает букву "А" из тестовой выборки, поскольку все метрики близости покажут достаточно большое расстояние от *в*) и до *а*) и до *б*).

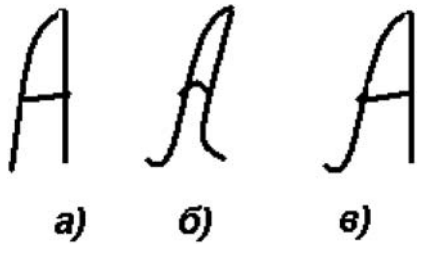


Рис. 1. Примеры написания буквы "А"

Выходом в данной ситуации является извлечение признаков и использование для распознавания буквы их комбинации. Например, если в данном случае мы обучим один нейрон сети реагировать на левую палочку буквы "А", второй — на правую, а третий — на среднюю, а потом сгруппируем выходы этих нейронов, то получим очень высокоэффективный классификатор.

Далее в работе рассматривается методика построения такой нейронной сети, названной в соответствии с принципом работы *признаковой*.

Предварительное построение первого слоя сети

Пусть обучающая выборка состоит из P примеров. Каждый пример представляет собой серое прямоугольное изображение размера $(2M + 1) \times (2N + 1)$, где $(2M + 1)$ — ширина изображения, $(2N + 1)$ — высота. Число классов образов обозначим через K.

Первый слой нейронной сети полностью оправдывает название сети — признаковая. Собственно только первый слой в нашей сети и является признаковым, назначение остальных слоев — это комбинация признаков для получения полного описания заданных классов образов.

Функция нейронов первого слоя сети — реакция на некоторый участок входного изображения, содержащий определенный признак. Этим признаком может быть прямая или загнутая линия, угол, цвет фона, направление градиента яркости и т. д. Внешний вид признака, на который должен реагировать нейрон, выбирается из кусочков изображений обучающей выборки.

Перед построением сети пользователь должен определить параметры нейронов 1-го слоя. Для этого он задает возможные варианты расположений, размеров и масштабов входных окон (рис. 2), используемые функции активации для нейронов 1-го слоя. Пусть у нас будет Q вариантов входных окон: $(2M_1 + 1) \times (2N_1 + 1)$, ..., $(2M_Q + 1) \times (2N_Q + 1)$ с центрами в точках (x_i, y_i), $x_i = [-M...M]$, $y_i = [-N...N]$, $i = [1...Q]$ различной высоты/ширины/масштаба, с разными функциями активации.

В качестве функции активации пользователь может использовать любую функцию, которая при реакции на входное окно выдает +1 (есть признак) и 0 (нет признака). Для примера в данной работе мы

Рис. 2. Возможные варианты расположений и размеров входных окон на примере задачи распознавания букв

рассмотрим в качестве функции активации радиальную базисную функцию (RBF). В этом случае веса нейрона устанавливаются равными значению серого цвета в соответствующей точке входного окна. Каждое входное окно является центром многомерного RBF-шара с некоторым, заданным пользователем, радиусом R. Радиус R определяет, насколько точно кусочек входного изображения должен соответствовать признаку, хранимому в нейроне. Чем меньше R, тем больше они должны быть похожи. Отметим, что в данной задаче распознаваемые изображения перед использованием нормализовались по яркости.

Имея P примеров в обучающей выборке, мы получим $P \times Q$ нейронов в 1-м слое.

Выход i-го нейрона 1-го слоя при использовании RBF-функции вычисляется следующим образом:

$$g_{i1} = \sqrt{\sum_{xx = x - M_q}^{x + M_q} \sum_{yy = y - N_q}^{y + N_q} \left(c_{i(xx,\, yy)} - v_{(xx,\, yy)}\right)^2};$$

$$y_{i1} = \begin{cases} 1 \text{ при } g_{i1} < R; \\ 0 \text{ при } g_{i1} \geqslant R, \end{cases} \qquad (1)$$

где $c_{i(xx,\, yy)}$ — значение веса i-го нейрона в точке (xx, yy); $v_{(xx,\, yy)}$ — значение цвета в точке (xx, yy) во входном окне; q — выбранный для i-го нейрона вариант размера/масштаба окна; g_{i1} — значение активации i-го нейрона 1-го слоя сети; y_{i1} — выходной сигнал i-го нейрона 1-го слоя сети, принимает два значения: 1 — нейрон сработал, и 0 — нейрон не сработал.

Для задачи распознавания букв при размере символ: высота $h = 25$, ширина 23 рекомендуется использовать значение Q не менее 20 и число примеров P для каждой буквы не менее 30 (при достаточной вариативности внешнего вида символов). Таким образом, на начальном этапе в первом слое сети мы имеем достаточно большое число нейронов (признаков). В дальнейшем какие-то из них окажутся полезными — они останутся в конечном варианте сети, а какие-то бесполезными — они будут удалены.

Функционалы для выбора нейронов

Для каждого i-го нейрона любого слоя сети определим два функционала: $\Phi_{(i,\, k)}$ и $\Omega_{(i,\, k)}$ следующим образом:

$$\Phi_{(i,\, k)} = \frac{\sum\limits_{\text{по примерам } k\text{-го класса}} y_i}{P_k}; \qquad (2)$$

$$\Omega_{(i,\, k)} = \frac{\sum\limits_{\text{по примерам НЕ } k\text{-го класса}} y_i}{P - P_k}, \qquad (3)$$

где y_i — выход i-го нейрона: 1 в случае срабатывания и 0 в противном случае; P_k — число примеров в k-м классе; $\Phi_{(i,\, k)}$ — функционал соответствия i-го нейрона k-му классу; $\Omega_{(i,\, k)}$ — функционал соответствия i-го нейрона всем классам, кроме k-го.

Как видно из формул (2), (3), $\Phi_{(i,\, k)}$ и $\Omega_{(i,\, k)}$ принимают значения от 0 до 1. Чем ближе значение $\Phi_{(i,\, k)}$ к 1, тем на большее число примеров k-го класса реагирует i-й нейрон. Чем ближе значение $\Omega_{(i,\, k)}$ к 1, тем на большее число примеров НЕ k-го класса реагирует нейрон. Нашим идеалом является такой нейрон, для которого $\Phi_{(i,\, k)} = 1$ и $\Omega_{(i,\, k)} = 0$.

Для нейронов начальных слоев мы при достаточном размере обучающей выборки, естественно, не сможем получить таких значений для $\Phi_{(i,\, k)}$ и $\Omega_{(i,\, k)}$, и для них мы будем лишь максимизировать $\Phi_{(i,\, k)}$ и минимизировать $\Omega_{(i,\, k)}$. И лишь для выходного нейрона, соответствующего решению по k-му классу, мы получим $\Phi_{(i,\, k)} = 1$ и $\Omega_{(i,\, k)} = 0$.

Введем еще один функционал — обобщенный функционал соответствия i-го нейрона k-му классу:

$$\Delta_{(i,\, k)} = \frac{(P - P_k) \sum\limits_{\text{по примерам } k\text{-го класса}} y_i}{P_k\left(1 + \sum\limits_{\text{по примерам НЕ } k\text{-го класса}} y_i\right)}. \qquad (4)$$

При построении нейронов начальных слоев будем максимизировать значение $\Delta_{(i,\, k)}$.

Алгоритм построения нейрона 2-го слоя сети

Первый слой у нас — признаковый, т. е. каждый нейрон 1-го слоя реагирует (выдает 1 на выходе) на определенный признак в конкретной точке (x, y). Во 2-м слое мы хотим получить более обобщенное реагирование на конкретный признак.

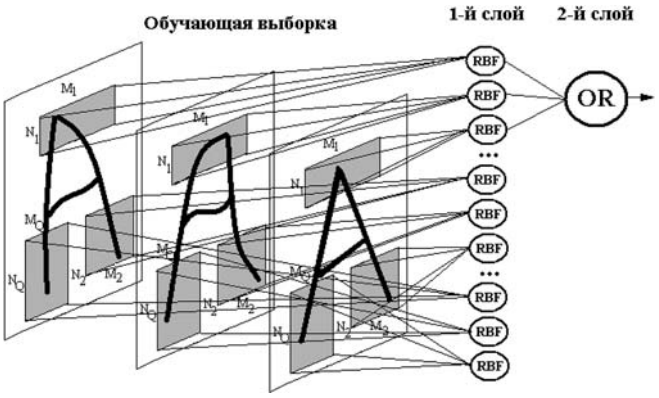


Рис. 3. Иллюстрация методики построения нейрона 2-го слоя сети

Например, для задачи распознавания буквы "А" (рис. 3), в 1-м слое построим множество RBF-нейронов, реагирующих на разные виды ее верхушки, разные виды ножек и т. д. Во 2-м слое нужно построить один нейрон, который будет срабатывать при наличии на входном изображении не какой-то конкретной верхушки, а любой верхушки, похожей на имеющиеся в обучающей выборке для буквы "А" (рис. 3).

Итак, из первого слоя выбираем $i1$-й нейрон с максимальным значением $\Delta_{(i1,\,k)}$ для какого-либо k-го класса. Создаем $i2$-й нейрон во втором слое и связываем его с $i1$-м нейроном первого слоя. Выход нейрона 2-го слоя вычисляется следующим образом:

$$g_{i2} = \sum_{i1=1}^{N} y_{i1};$$
$$y_{i2} = \begin{cases} 1 \text{ при } g_{i2} > 1; \\ 0 \text{ при } g_{i2} = 0, \end{cases} \qquad (5)$$

где N — число входов $i2$-го нейрона; y_{i1} — выход $i1$-го нейрона первого слоя; g_{i2} — значение активации $i2$-го нейрона; y_{i2} — выход $i2$-го нейрона.

Фактически нейрон 2-го слоя реализует логическую функцию ИЛИ над связанными с ним нейронами 1-го слоя. Его выход равен 1 в том случае, если сработал хотя бы один нейрон 1-го слоя. На начальном этапе выход нейрона 2-го слоя равен выходу соединенного с ним нейрона 1-го слоя.

Пусть $i1$-му нейрону 1-го слоя соответствует q-й вариант входного окна с координатами (x, y). Если $\Phi_{(i1,\,k)} < 1$, то далее мы рассматриваем только нейроны с q-м видом входного окна и координатами (x, y) и среди них ищем такой $j1$-й нейрон, чтобы $\Phi_{(i2,\,k)} = \Phi_{(i1,\,k)} + \Phi_{(j1,\,k)}$ было максимально.

Продолжаем присоединять к $i2$-му нейрону 2-го слоя новые нейроны 1-го слоя до тех пор, пока не получим $\Phi_{(i2,\,k)} = 1$. Это значит, что $i2$-й нейрон срабатывает на все примеры k-го класса. То есть $i2$-й нейрон будет реагировать на то изображение, у которого в некоторой окрестности точки (x, y) найдется признак, похожий хотя бы на один из признаков связанных с ним нейронов 1-го слоя.

Алгоритм построения нейрона 3-го слоя сети

Итак, нейрон 1-го слоя реагирует на конкретный признак, например, верхушку буквы "А" шрифта Arial. Нейрон 2-го слоя реагирует на обобщенный признак — верхушку буквы "А" любого шрифта и стиля. Нейрон 3-го слоя будет реагировать на группу признаков: верхушку, ножки буквы "А", палочку посередине. Таким образом, срабатывание нейрона 3-го слоя означает наличие на входе сети объекта соответствующего класса.

Выход нейрона 3-го слоя вычисляется следующим образом:

$$g_{i3} = \sum_{i2=1}^{N} y_{i2};$$
$$y_{i3} = \begin{cases} 1 \text{ при } g_{i3} = N; \\ 0 \text{ при } g_{i3} < N, \end{cases} \qquad (6)$$

где N — число входов $i3$-го нейрона; y_{i2} — выход $i2$-го нейрона второго слоя; g_{i3} — значение активации $i3$-го нейрона; y_{i3} — выход $i3$-го нейрона.

То есть нейрон 3-го слоя реализует логическую функцию И над связанными с ним нейронами 2-го слоя. Его выход равен 1 только в том случае, если сработали все связанные с ним нейроны 2-го слоя.

Как только построен первый нейрон 2-го слоя, мы переходим к созданию первого нейрона 3-го слоя. Естественно, он будет относиться к тому же классу, что и нейрон 2-го слоя. На начальном этапе выход нейрона 3-го слоя равен выходу соединенного с ним нейрона 2-го слоя. То есть нейрон 3-го слоя реагирует только на один какой-то признак распознаваемого класса (например, на среднюю палочку буквы "А" для всех примеров данного класса). При этом попадутся другие буквы ("Е", "Н", "В"), на которые будет срабатывать этот нейрон. Наша задача — построить следующий нейрон 2-го слоя, который будет реагировать на другой признак распознаваемого класса (например, на верхушку буквы "А"), которого нет у букв "Е", "Н", "В".

Для этого оставим для дальнейшего рассмотрения только те примеры обучающей выборки, на которые уже реагирует наш созданный $i3$-й нейрон, или, другими словами, выход $i3$-го нейрона на эти примеры равен 1. То есть в оставшейся для рассмотрения обучающей выборке будут все примеры

класса "А" и какое-то число распознанных примеров некоторых других классов ("Е", "Н", "В").

По уже описанному выше алгоритму создаем второй $i2$-й нейрон во втором слое. Нейроны 1-го слоя присоединяются к нему также в соответствии с максимизацией функционала $\Delta_{(i1,\,k)}$. Но, еще раз напомним, рассматриваются только те примеры, на которые сработал ранее построенный нейрон 3-го слоя.

В соответствии с данным алгоритмом продолжаем добавлять нейроны во 2-й слой сети, связывая их с нейроном $i3$ до тех пор, пока не будут распознаны все примеры k-го класса, т. е. пока не получим $\Phi_{(i3,\,k)} = 1$.

Построение сети для K классов образов

В задаче распознавания K классов образов при $K > 1$ мы строим сеть для первого класса описанным выше способом. Для второго и остальных классов мы стараемся использовать уже построенные ранее нейроны 2-го слоя. Например, если у нас во 2-м слое уже есть нейрон, срабатывающий на вертикальную палочку слева, то он будет использоваться для множества нейронов 3-го слоя, отвечающих за классы "г", "н", "и", "п" и т. д.

Такая методика позволяет очень существенно повысить качество обобщения и эффективно использовать уже построенные нейроны. Кроме того, для многоклассовых задач, таких как распознавание букв, такая методика просто незаменима при увеличении числа используемых букв, поскольку в этом случае размер сети увеличивается минимально, а время просчета не меняется вообще, так как обсчитываются не все нейроны сети (рис. 4). Как видно из рис. 4, обсчитываются все нейроны 1-го слоя, некоторые нейроны 2-го слоя, и один нейрон 3-го слоя.

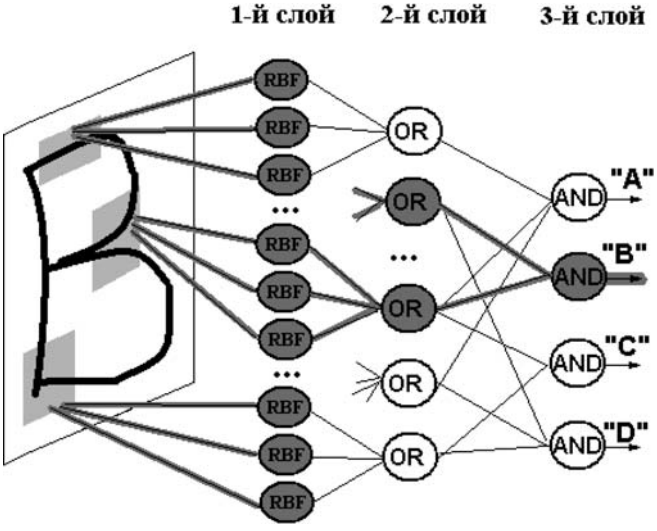


Рис. 4. Пример прохождения сигнала по нейронам сети при распознавании входного изображения

По окончании построения сети неиспользованные в 1-м слое нейроны удаляются.

Эксперименты

Эксперименты на задаче распознавания печатных букв проводились следующим образом. Нейронная сеть строилась для уже выделенных и нормализованных по яркости символов, взятых с отсканированного текста. Все они были смасштабированы к размеру 25 × 23. Пример такого символа приведен на рис. 2.

Сперва были сформированы входные окна в разных частях исходного изображения (см. рис. 2). Затем было взято небольшое число символов для каждого класса и построена нейронная сеть. Далее на отсканированных текстах были выделены нераспознанные буквы. Они были добавлены в обучающую выборку, сеть опять обучена. Так продолжалось до достижения 100 %-ного распознавания всех отсканированных текстов (30 страниц). Для каждой буквы в обучающей выборке потребовалось от 20 до 100 примеров. При этом из рассмотрения были выкинуты символы, не распознаваемые человеческим глазом (из-за качества изображения).

Список литературы

1. **Cannon M., Fugate M., Hush Don R., and Scovel C.** Selecting a restoration technique to minimize OCR error // IEEE Transactions on Neural Networks. May 2003. Vol. 14, N 3.
2. **Cannon M., Hochberg J., and Kelly P.** Quality assessment and restoration of typewritten document images // Int. J. Document Anal. Recognition. 1999. Vol. 2, N 2/3. P. 80—89.
3. **Tang X., Gap X., Liu J., and Zhang H.** A spatial-temporal approach for video caption detection and recognition // IEEE Transactions on Neural Networks. July 2002. Vol. 13. N 4.
4. **Weinman J. J., Learned-Miller E., Hanson A. R.** Scene text recognition using similarity and a lexicon with sparse belief propagation // IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Special Issue on Probabilistic Graphical Models. 2009. Vol. 31. N 10. P. 1733—1746.
5. **Kae A., Smith D. A., Learned-Miller E.** Learning on the fly: a font-free approach toward multilingual OCR // International Journal on Document Analysis and Recognition, Sp04. 2012. Vol. 14 (3). P. 289—301.
6. **Ho T. K., Nagy G.** OCR with no shape training // 15th International Conference on Pattern Recognition, Barcelona, Spain. 2000. Vol. 4. P. 4027.

CONTENTS

A problem of search for sources of chaotic oscillation in power systems is formulated. Two algorithms solving the problem are presented, one based on best-first search, other on a parallelized recursive graph traversal. Based on comparative analysis of the algorithms, possibility of their practical application is discussed.

Keywords: criterion function, synchronous generator, power system, parallel computing

Possibility of using combined algorithm of optimal resource allocation for maintenance systems reliability is considered.

Keywords: reliability, optimization, dynamic programming, branch and bound method.

Within the bounds of the information model of neural networks of metrological theory describing of dynamics interaction processes in its information field is considered. The dynamics of information processes of the interaction of objects and environments such neural networks is the main mechanism, which induces an information field. The methodology of theory uses the semiotic approach to find solutions of basic problem: wherein and how manifest laws of interaction, mutual influence and interaction signal and symbol (pattern), both physical and information entities?

Keywords: information dynamics, metrology information, the information field, information and fractal connectivity, information flow, information model of a neural network

This paper discribes availability of artificial neural networks in common with digital video processing algorithms to make an automatic traffic density estimation and vehicle classification system. They suggest a new approach to improve the system using parallel computing.

Keywords: a neural network, parallel calculations, a video image, numeral handling, density of a transport flow, a video stream, the automated system

New algorithm of neural network construction for image processing problems is proposed. The main difference of the proposed method is a construction of featured first layer in neural network. That is each neuron of first layer respond to one feature of input image. Other layers group outputs of first layer neurons in optimum way to class recognition. Moreover the proposed algorithm allows to receive 100 % on learning sample without quality worsening on test samlpe. Efficiency of this method for multi-class recognition problems is shown on the example of letter recognition task.

Keywords: neural networks, image processing, letter recognition

Адрес редакции:

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**

E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Технический редактор *Е. В. Конова*.

Корректор *Е.В. Комиссарова*.

Сдано в набор 07.09.2012. Подписано в печать 22.10.2012. Формат 60×88 1/8. Бумага офсетная.

Усл. печ. л. 8,86. Заказ IT1112. Цена договорная.

Журнал зарегистрирован в Министерстве Российской Федерации по делам печати, телерадиовещания и средств массовых коммуникаций.

Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".

105120, г. Москва, ул. Нижняя Сыромятническая, д. 5/7, стр. 2, офис 2.