МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. М. В. ЛОМОНОСОВА

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И
КИБЕРНЕТИКИ

СБОРНИК СТАТЕЙ СТУДЕНТОВ И АСПИРАНТОВ

Выпуск 1

Под редакцией
проф. С.А. Ложкина

2002

УДК 517.929+517.957+517.929+681.3.06
ББК 22.19
П 75

Печатается по решению РИСО
факультета вычислительной математики и кибернетики
МГУ им. М.В. Ломоносова

Рецензенты:
акад. Ильин В.А., проф. Ложкин С.А., проф. Бенинг В.Е.

Сборник статей студентов и аспирантов. Факультет ВМиК МГУ им. М.В. Ломоносова. Вып. 1/ Под ред. проф. С.А. Ложкина. — 2002. — 92 с.
ISBN 5-89407-142-9

ISBN 5-89407-142-9

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

В 2001 году на факультете Вычислительной математики и кибернетики Московского государственного университета им. М.В. Ломоносова был вновь образован Совет молодых ученых. Именно благодаря инициативе Совета и появился на свет этот сборник работ студентов и аспирантов факультета ВМиК.

В сборник вошли работы молодых ученых, представляющих различные научные направления и различные кафедры нашего факультета. Это и фундаментальные математические исследования, и исследования прикладных математических задач. В сборник вошли также работы, относящиеся к современным проблемам алгоритмизации и программирования, есть работа, где математический аппарат применяется к исследованию социологических вопросов.

Хотелось бы отметить научный уровень работ, публикуемых в данном сборнике, и широкий диапазон проблем, который они охватывают. Все это свидетельствует о том, что у нового молодого поколения воспитанников факультета ВМиК есть стремление к исследовательской, научной работе, и о том, что у многих научных школ факультета есть достойное молодое пополнение.

Это первый сборник работ студентов и аспирантов факультета ВМиК. Хочется выразить надежду, что он станет регулярным изданием, которое откроет путь в науку для новых воспитанников нашего факультета.

Зам. декана факультета ВМиК
по научной работе
профессор *С.А. Ложкин*

О ВНЕСЕНИИ ИЗМЕНЕНИЙ ВО ВСТРОЕННУЮ СИСТЕМУ ПРИ НАРУШЕНИИ ДИРЕКТИВНЫХ СРОКОВ ЗАДАЧ

Балашов В.В.

§ 1. Введение

Процесс разработки встроенной вычислительной системы (ВсВС) включает в себя, в частности, выбор аппаратной платформы ВсВС, разработку вычислительных задач, которые будут выполняться на ВсВС, и составление расписания (планирование) выполнения задач.

Исторически для решения проблемы планирования задач на ВсВС используются статические алгоритмы планирования, строящие статическое циклическое расписание выполнения задач. Использование статического планирования позволяет применять на ВсВС простые управляющие алгоритмы, которые выполняют активацию вычислительных задач в соответствии с заранее заданным фиксированным расписанием.

С использованием фиксированных алгоритмов планирования связан ряд проблем [1, 2], в том числе:

- большая вычислительная сложность алгоритмов планирования;
- плохая устойчивость составленного расписания к изменению параметров набора задач. При увеличении времени выполнения отдельных задач или при добавлении новых задач в набор задач часто требуется повторный синтез расписания;
- необходимость разбиения крупных вычислительных задач на более мелкие для снижения сложности размещения задач в циклах статического расписания;
- низкая эффективность при планировании задач с взаимно простыми периодами или периодами, имеющими сравнительно малое наибольшее общее кратное.

Для того чтобы избежать этих проблем, используется динамическое планирование задач (ДПЗ) на ВсВС, осуществляемое задачей-планировщиком в ходе работы ВсВС.

Будем рассматривать ВсВС жесткого реального времени с ДПЗ. На такой ВсВС все задачи должны выполняться строго в пределах директивных сроков (ДС).

При разработке ВсВС с ДПЗ встает проблема доказательства того, что для заданных набора задач (НЗ), параметров производительности ВсВС и схемы ДПЗ при любом прогоне ВсВС все задачи выполняются в пределах ДС (т.е. НЗ обладает свойством *гарантированной планируемости* на данной ВсВС). Эта проблема решается посредством проверки выполнимости для данных ВсВС и НЗ формул достаточных условий гарантированной планируемости задач (ГПЗ) [2, 3]. Для определенных архитектур ВсВС эти условия также являются необходимыми.

В случае, если для данных ВсВС и НЗ условия ГПЗ не выполняются, разработчик ВсВС должен принять решение о внесении изменений в параметры НЗ и/или о повышении производительности ВсВС с тем, чтобы НЗ стал обладать свойством гарантированной планируемости на ВсВС.

Ручной подбор изменений параметров НЗ и производительности ВсВС является трудоемкой задачей. Актуальна задача разработки методов автоматизации выбора изменений указанных параметров, приводящих к выполнению условий ГПЗ.

В данной работе рассматривается подход к автоматической генерации рекомендаций по внесению изменений в параметры НЗ и производительности ВсВС, основанный на формулах условий ГПЗ.

§ 2. Математическая модель планируемости задач

2.1. Архитектура встроенной системы и параметры набора задач. Определим архитектуру ВсВС и структуру НЗ, для которых будет исследоваться свойство ГПЗ. Будем рассматривать распределенную ВсВС, состоящую из центральной вычислительной машины (ЦВМ) и нескольких периферийных вычислительных машин (ПВМ), подключенных к ЦВМ через общую шину. ПВМ обмениваются по шине данными с ЦВМ. Все обмены производятся по инициативе ЦВМ, которая является мастером шины. Из параметров производительности ВсВС будем рассматривать вычислительную мощность процессора ЦВМ и пропускную способность общей шины.

Распределенные ВсВС с указанной архитектурой широко применяются в качестве бортовых управляющих систем самолетов. В качестве ЦВМ выступает бортовая цифровая вычислительная машина (БЦВМ), а в качестве ПВМ — устройства авионики, являющиеся специализированными вычислителями. Примером протокола, в соответствии с которым производятся обмены по шине, является MIL-STD-1553B.

На ЦВМ выполняется набор задач $\{Z_k\}$, $k = 1, \ldots, N$, обладающих следующими параметрами: T_k (период задачи), D_k (директивный срок задачи), P_k (приоритет задачи), C_k (максимальное время выполнения задачи) и E_k (максимальное время, в течение которого задача непрерывно осуществляет обмен по шине). В C_k входит время переключения процессора на выполнение Z_k с более низкоприоритетной задачи, а также время на обратное переключение.

На ЦВМ используется схема динамического планирования задач с фиксированными приоритетами и вытеснением. Каждой задаче сопоставляется уникальный приоритет P_k. Чем больше значение P_k, тем выше приоритет задачи. Задачи в НЗ упорядочены по приоритетам: $P_k < P_l$ при $k < l$. Сопоставление приоритетов задачам будем считать заданным заранее и неизменным.

Задача Z_k $(k = 1, \ldots, N)$ выполняется периодически и становится готовой к выполнению в моменты времени $n * T_k$, $k = 0, 1, 2, \ldots$. Планировщик поддерживает очередь готовых к выполнению *экземпляров задач* (ЭЗ). В момент наступления готовности задачи Z_k в очередь на выполнение ставится новый ЭЗ Z_k. В данной работе рассматриваются НЗ, в которых для некоторых k верно $D_k > T_k$. Для таких k при функционировании ВсВС допустимо нахождение в очереди на выполнение более одного ЭЗ Z_k. Постановку ЭЗ в очередь на выполнение будем называть *активацией* ЭЗ. Очередной ЭЗ Z_k начинает выполнение, как только процессор освобождается от выполнения ЭЗ с более высоким приоритетом и предыдущих ЭЗ Z_k. При активации задачи с более высоким, чем у Z_k, приоритетом ЭЗ Z_k вытесняется этим ЭЗ и продолжает выполнение только после того, как процессор освобождается от выполнения более высокоприоритетных ЭЗ. Если некоторый ЭЗ Z_k осуществляет обмен по шине, он не может быть вытеснен до завершения обмена.

Обозначим за R_k наихудшее время отклика задачи Z_k, т.е. максимальную по всем возможным прогонам ВсВС длительность интервала времени от начала очередного периода задачи Z_k до завершения выполнения ЭЗ Z_k, активированного в начале этого периода. Требования жесткого реального времени

задаются неравенством: $R_k \leqslant D_k$.

ЭЗ обмениваются данными через общее поле параметров, из которого каждый ЭЗ в начале выполнения считывает все входные данные и в которое перед завершением записывает результаты вычислений. Такая организация обмена данными реализована в нескольких операционных системах для ВсВС, в частности в ОС Chimera [4].

2.2. Формулы условий гарантированной планируемости. Рассмотрим архитектуру ВсВС и набор задач, приведенные в разделе 2.1. Пусть стоит задача: определить, обладает ли НЗ $\{Z_k\}$ свойством гарантированной планируемости на данной ВсВС. Эта задача может быть решена при помощи проверки истинности для данного НЗ и ВсВС достаточных условий ГПЗ [5]. Для ВсВС и НЗ, удовлетворяющих требованиям раздела 2.1, эти условия также являются необходимыми.

Формулы условий ГПЗ для ВсВС с ДПЗ могут быть записаны в общем виде:

$$R_k(\Psi) \leqslant D_k, \quad k = 1 \ldots N, \tag{1}$$

где Ψ есть набор параметров $\{T_k, P_k, C_k, E_k\}$, $k = 1, \ldots, N$.

Рассмотрим формулы для вычисления значений R_k для НЗ, в которых $D_k \leqslant T_k$, $k = 1, \ldots, N$. При выполнении таких НЗ, каждый ЭЗ Z_k должен завершить выполнение до наступления следующего периода Z_k.

Для таких НЗ максимальное время отклика R_k задачи Z_k при выполнении НЗ в пределах директивных сроков определяется значением C_k и максимальным влиянием более высокоприоритетных ЭЗ, вытесняющих ЭЗ Z_k. В R_k входит также максимальное время, в течение которого ЭЗ Z_k не может вытеснить ЭЗ с более низким приоритетом в связи с тем, что этот ЭЗ выполняет обмен по шине. Это время называется временем блокировки ЭЗ Z_k. Обозначим его B_k. Из определения B_k следует формула:

$$B_k = \max_{n<k} E_n.$$

Обозначим за I_k максимальное суммарное время выполнения экземпляров задач с приоритетом, большим, чем P_k, в интервале от активации ЭЗ Z_k до завершения выполнения этого ЭЗ. Максимальное время от активации ЭЗ Z_k до завершения выполнения этого ЭЗ определяется соотношением

$$R_k = C_k + I_k + B_k,$$

в котором

$$I_k = \sum_{j>k} \left\lceil \frac{R_k}{T_j} \right\rceil * C_j. \tag{2}$$

Для вычисления R_k можно использовать следующее рекуррентное соотношение [3]:

$$R_k\,(n+1) = B_k + C_k + \sum_{j>k} \left\lceil \frac{R_k\,(n)}{T_j} \right\rceil * C_j. \tag{3}$$

Данное соотношение решается итеративно с начальным приближением $R_k\,(0) = C_k$. Итеративный процесс сходится к наименьшему значению R_k, удовлетворяющему (2), что доказано в [5].

Пусть для некоторых задач Z_m верно, что $D_m > T_m$. В таком случае при вычислении R_m необходимо учитывать влияние ЭЗ Z_m, не успевших завершить выполнение до активации рассматриваемого ЭЗ Z_m. Для учета этого влияния вводится понятие интервала занятости (ИЗ) уровня m, в течение которого выполнились q экземпляров задачи Z_m (обозначим такой ИЗ за W_{mq}); W_{mq} есть интервал времени прогона ВсВС, удовлетворяющий следующим условиям:

1. W_{mq} начинается в момент активации ЭЗ Z_m, при условии, что в этот момент предыдущий экземпляр Z_m уже завершил выполнение. Данный ЭЗ Z_m получает индекс 0 относительно W_{mq};
2. W_{mq} заканчивается в момент завершения ЭЗ Z_m с индексом q относительно W_{mq};
3. в каждой точке W_{mq} как минимум один ЭЗ Z_m присутствует в очереди на выполнение или выполняется.

Обозначим за L_{mq} максимальную длительность ИЗ W_{mq} по всем возможным прогонам ВсВС. Для L_{mq} справедливо следующее соотношение [3]:

$$L_{mq} = B_m + (q+1) * C_m + \sum_{j>m} \left\lceil \frac{L_{mq}}{T_j} \right\rceil * C_j. \tag{4}$$

Значение L_{mq} в (4) вычисляется итеративно аналогично тому, как вычисляется R_m в (3). Максимальное время отклика q-го экземпляра задачи Z_m равно $(L_{mq} - q * T_m)$.

Наихудшее (максимальное) время отклика задачи Z_m рассчитывается по следующей формуле:

$$R_m = \max_{q=0,1,\ldots Q} \left(L_{mq} - q * T_m\right), \tag{5}$$

где Q есть наименьшее значение q, при котором $L_{mq} < (q+1)*$ T_i. Экземпляр задачи Z_m с индексом Q относительно W_{mQ} завершает выполнение до активации следующего ЭЗ Z_m и не влияет на время выполнения последующих ЭЗ Z_m.

На практике можно ограничиться значениями q, для которых L_{mq} не превышает директивный срок D_n для всех $n < m$. Если для некоторого $n < m$ выполняется неравенство $L_{mq} > D_n$, то найдется ЭЗ Z_n, который не выполнится в пределах своего директивного срока D_n, поскольку процессор будет занят выполнением экземпляров более высокоприоритетной задачи Z_m, и (или) некоторые ЭЗ Z_m будут находиться в очереди на выполнение.

§ 3. Задача автоматической генерации рекомендаций

Пусть на некотором этапе разработки ВсВС с ДПЗ определен набор задач $\{Z_k\}$ и значения его параметров T_k, D_k, P_k, E_k и C_k. На этом этапе при помощи условий ГПЗ возможно оценить, являются ли задачи из данного набора задач выполнимыми в пределах директивных сроков при любом прогоне ВсВС.

В случае если условия ГПЗ не выполняются, перед разработчиками ВсВС встает задача определения изменений параметров НЗ и производительности ВсВС, после внесения которых НЗ стал бы обладать свойством гарантированной планируемости на ВсВС. Актуальна задача автоматизации поиска требуемых изменений этих параметров. В настоящем разделе представляется формальная постановка этой задачи, основанная на формулах условий ГПЗ.

В соответствии с (1) формулы условий ГПЗ имеют вид

$$\left\{D_k - R_k\left(\Psi\right) \geqslant 0\right\}, \quad k = 1, \ldots, N, \tag{6}$$

где Ψ есть набор параметров $\{T_k, P_k, C_k, E_k\}$, $k = 1, \ldots, N$.

Пусть некоторые из условий ГПЗ не выполняются. Это означает, что существует прогон ВсВС, при котором некоторые ЭЗ не выполняются в рамках директивных сроков. Обозначим за Ω вектор, содержащий значения $\{T_k, D_k, P_k, E_k, C_k\}$ для всех $k = 1, \ldots, N$. Обозначим за $\overrightarrow{G}\left(\Omega\right)$ вектор значений $\{D_k - R_k\left(\Psi\right)\}$, $k = 1, \ldots, N$, в котором все положительные

компоненты заменены нулями, а отрицательные — своими абсолютными значениями. Тогда выполнение условий ГПЗ (6) эквивалентно выполнению соотношения

$$\overrightarrow{G}(\Omega) = \overrightarrow{0}. \tag{7}$$

Для достижения выполнимости свойства ГПЗ для набора задач на ВсВС необходимо изменить Ω таким образом, чтобы стало выполняться условие 7.

Анализ свойств функции $\overrightarrow{G}(\Omega)$ показывает, что при следующих изменениях компонентов вектора Ω значения компонентов $\overrightarrow{G}(\Omega)$ не возрастают:

1. увеличение значения T_k или D_k для некоторого k;
2. уменьшение значения C_k или E_k для некоторого k.

Несложно доказать, что при достаточно больших T_k и D_k и достаточно малых C_k и E_k все компоненты $\overrightarrow{G}(\Omega)$ становятся равными нулю.

Пусть разработчик ВсВС может варьировать параметры набора задач и производительности ВсВС в определенных границах. Направление изменения параметров соответствует пунктам 1 и 2. Обозначим границы изменения для T_k, D_k, P_k, E_k, C_k соответственно за T_k^*, D_k^*, E_k^*, C_k^*. Граница уменьшения C_k определяется возможностями оптимизации кода задачи или повышения производительности процессора ЦВМ. Граница увеличения T_k может определяться допустимым временем устаревания вычисляемых задачей Z_k данных. Граница уменьшения E_k определяется возможностями повышения производительности общей шины, связывающей ЦВМ и ПВМ.

Исходные значения параметров обозначим за T_k^0, D_k^0, E_k^0, C_k^0. Вектор, состоящий из этих значений для $k = 1, \ldots, N$, обозначим за Ω^0. Вектор, полученный из Ω^0 заменой значений T_k^0, D_k^0, E_k^0, C_k^0 на T_k^*, D_k^*, E_k^*, C_k^*, обозначим за Ω^*. Для общности введем обозначения: $P_k^0 = P_k^* = P_k$.

Задача автоматической генерации рекомендаций по изменению параметров набора задач и производительности ВсВС с целью достижения гарантированной планируемости НЗ на ВсВС формулируется следующим образом.

Исходные данные:

1. рассматриваемая ВсВС имеет архитектуру и схему планирования задач, описанные в разделе 2.1;
2. для набора задач $\{Z_k\}$ известно значение Ω^0;

3. значения параметров T_k, D_k, E_k, C_k можно изменять в пределах от T_k^0, D_k^0, E_k^0, C_k^0 до T_k^*, D_k^*, E_k^*, C_k^*.

Требуется:

1. определить, является ли НЗ $\{Z_k\}$ гарантированно планируемым на данной ВсВС;
2. если НЗ не является гарантированно планируемым, найти в границах, заданных Ω^0 и Ω^*, вектор Ω' параметров НЗ и производительности ВсВС, при которых НЗ становится гарантированно планируемым, т.е. $\overrightarrow{G}(\Omega) = \overrightarrow{0}$. Искомый вектор Ω' должен лежать на границе области гарантированной планируемости, и при изменении Ω' в направлении Ω^0 набор задач должен терять свойство гарантируемой планируемости.

Требование нахождения Ω' на границе области гарантированной планируемости (ОГП) введено для того, чтобы отсечь возможное тривиальное решение $\Omega' = \Omega^*$. Дополнительное требование к Ω' означает, что приблизить Ω' к Ω^0 путем изменения отдельных компонентов в сторону Ω^0 невозможно. Это отсекает тривиальный способ улучшения решения.

При постановке задачи и ее решении могут учитываться зависимости между компонентами Ω, следующие из свойств параметров ВсВС и НЗ. Вместо задания параметра E_k для каждой задачи можно задавать максимальную длину M_k сообщения, передаваемого этой задачей по шине, а в качестве параметра производительности ВсВС ввести пропускную способность BW общей шины. Значения E_k вычисляются по формуле $E_k = M_k/BW$.

Пусть найдено значение Ω', являющееся решением задачи, Ω' определяет изменения параметров НЗ и производительности ВсВС. После внесения этих изменений в разрабатываемую ВсВС и НЗ отсутствие при любом прогоне ВсВС превышения директивных сроков задачами из НЗ является математически доказуемым фактом. Таким образом, вектор Ω' может рассматриваться как рекомендация разработчикам ВсВС и набора задач по изменению параметров НЗ и ВсВС с целью достижения гарантированной планируемости задач.

Если решение не существует (это равносильно выполнению условия $\overrightarrow{G}(\Omega^*) \neq \overrightarrow{0}$), то внесения изменений в предложенных разработчиком пределах недостаточно для достижения гарантированной планируемости задач из НЗ $\{Z_k\}$ на данной ВсВС.

§ 4. Подходы к решению задачи

Задача, поставленная в §3, является достаточно общей. В разделах данного параграфа рассматриваются уточнения постановки задачи и подходы к решению с учетом этих уточнений.

В каждом подходе в области ГПЗ в границах, заданных от Ω^* к Ω^0, выбирается начальное приближение. Затем по алгоритму, выбор которого определяется исходя из уточнения постановки задачи, осуществляется переход от начального приближения к решению Ω', причем на каждом шагу перехода компоненты Ω изменяются в сторону Ω^0. Значение Ω', которое находят приведенные алгоритмы, удовлетворяет требованиям граничности и локальной неулучшаемости (см. постановку задачи) по построению алгоритмов.

4.1. Простейший подход. Пусть Ω_i — компонент вектора Ω. Зафиксируем некоторое натуральное N. Значение $\left(\Omega_i^0 - \Omega_i^*\right)/N$ будем называть элементарным приращением Ω_i.

Пусть с точки зрения разработки ВсВС и НЗ внесение элементарного приращения в каждый компонент Ω имеет одинаковую стоимость. Исходя из этого можно искать граничную точку области ГПЗ на отрезке прямой от Ω^* к Ω^0, что соответствует одинаковым относительным приращениям для каждого Ω_i (обозначим вектор координат найденной граничной точки за Ω''). Поиск можно осуществлять, например, методом деления отрезка пополам.

Вектор Ω'' может допускать покомпонентное приближение к Ω^0 для отдельных компонентов. Переход от Ω'' к некоторому Ω' может осуществляться как путем последовательного, по одному, приближения этих компонентов в сторону Ω^0 до достижения точки, в которой дальнейшее приближение невозможно. Из вида функции $\overrightarrow{G}(\Omega)$ следует, что одного прохода по всем таким компонентам Ω'' достаточно для нахождения Ω'.

Можно использовать более сложный переход от Ω'' к некоторому Ω', при котором для компонентов Ω'', допускающих приближение к Ω^0, осуществляется одновременное приближение к Ω^0 с равными относительными приращениями. Когда дальнейшее приближение невозможно, одновременное приближение продолжается для оставшихся компонентов, допускающих приближение к Ω^0, и так далее.

4.2. Подход с упорядочением по нежелательности изменений. Обозначим число компонентов Ω за M. Пусть, исходя из различной стоимости внесения элементарного приращения в компоненты Ω, каждому компоненту Ω сопоставлено уникальное число от 1 до M — номер в порядке убывания стоимости внесения элементарного приращения.

Выберем некоторое начальное приближение Ω^{**}, лежащее в области ГПЗ, в границах, заданных Ω^0 и Ω^*. Исходя из указанного выше упорядочения, будем осуществлять покомпонентное приближение от Ω^{**} к Ω^0, начиная с компонента с номером 1. По достижении границы области ГПЗ, продолжим приближение за счет компонента с номером 2, не выходя из области ГПЗ, и т.д. Для заданного начального приближения такой подход дает для компонента с номером 1 минимальный модуль разности с соответствующим компонентом Ω^0, для компонента с номером 2 (при заданном изменении компонента с номером 1) — также минимальный модуль разности с соответствующим компонентом Ω^0, и так далее.

4.3. Подход с учетом стоимости изменений. Сопоставим компоненту Ω_i для каждого i от 1 до M неотрицательное число F_i — стоимость внесения элементарного приращения для данного компонента (определение элементарного приращения см. в разделе 4.1). Для заданного Ω и Ω^0 суммарная стоимость внесения изменений в ВсВС и НЗ для перехода от Ω^0 к Ω вычисляется по следующей формуле:

$$F(\Omega) = \sum_{1}^{M} \left|\Omega^0 - \Omega\right| * N * F_i.$$

Пусть необходимо найти вектор Ω из области ГПЗ, на котором достигается минимум $F(\Omega)$. Задача в такой постановке является достаточно общей задачей оптимизации. Чтобы построить эффективный метод ее решения, необходимо произвести дальнейший анализ свойств функции $\overrightarrow{G}(\Omega)$. Такой анализ не рассматривается в данной статье, но является одним из направлений будущих исследований.

§ 5. Апробация подходов

Для апробации подходов из разделов 4.1 (простейший подход) и 4.2 (подход с упорядочением по нежелательности изменений) была построена программная реализация соответству-

ющих алгоритмов. Результаты их работы приведены в настоящем параграфе.

С целью повышения эффективности реализации, рассматривались целочисленные параметры встроенной системы и набора задач. Это не снижает практической применимости подходов, поскольку при разработке ВсС и НЗ чаще всего используются целочисленные параметры, что обеспечивается выбором единиц измерения и точности.

Рассматриваемые НЗ содержат как задачи, обменивающиеся данными по шине (в их название входит слово "обмен"), так и не обращающиеся к шине. Время выполнения задачи, обращающейся к шине, складывается из времени обмена по шине (Bt) и максимального времени обработки данных (Cc). Время обмена по шине вычисляется по формуле $Bt = L/BW$, где BW — пропускная способность шины, а L — длина передаваемого по шине сообщения. Каждая задача может передавать (принимать или отправлять) по шине только одно сообщение.

При оценке эффективности подходов может использоваться значение *загруженности процессора* U, которая вычисляется по формуле:

$$U = \sum_{i=1}^{N} \frac{Cc_i + Bt_i}{T_i} \cdot 100\%.$$

При $U > 100\%$ набор задач не может быть гарантированно планируемым. Чем выше U для гарантированно планируемого НЗ теоретического максимума в 100%, тем эффективнее используется процессорное время и, с этой точки зрения, лучше найденное решение Ω'.

В таблицах используются следующие обозначения, не вводившиеся выше:

Cc^0, Cc^*, Cc' — исходное, граничное и результирующее (соответствующее найденному решению) значения максимального времени обработки задачей данных;

T^0, T^*, T' — исходное, граничное и результирующее значения периода задачи;

BW^0, BW^*, BW' — исходное, граничное и результирующее значения пропускной способности шины ВсВС;

Bt^0, Bt' — время, затраченное задачей на передачу сообщения по шине при исходных и результирующих значениях параметров НЗ и производительности ВсВС;

R^0, R' — время отклика задачи при исходных и результирующих значениях параметров НЗ и производительности ВсВС;

U^0, U' — загруженность процессора при исходных и результирующих значениях параметров НЗ и производительности ВсВС.

5.1. Апробация простейшего подхода. Обозначим за U'' загруженность процессора в точке Ω'', лежащей на отрезке прямой от Ω^0 до Ω^*(см. описание подхода).

Из данных табл. 3 можно сделать вывод о том, что переход от U'' к U' является существенным с точки зрения повышения эффективности использования процессора. На отдельных тестах значение $(U' - U'')$ превышает 10%. На данном тесте особенно велико изменение параметра T_6: 43 единицы от $T_6'' = 443$ до $T_6' = 400$.

5.2. Апробация подхода с упорядочением по нежелательности изменений. При апробации в качестве начального приближения был выбран вектор Ω^*. Рассматривались следующие варианты упорядочения по убыванию нежелательности изменений:

1. $T_1, Cc_1, T_2, Cc_2, \ldots, T_N, Cc_N, BW$;
2. $T_N, Cc_N, T_{N-1}, Cc_{N-1}, \ldots, T_1, Cc_1, BW$.

Приведенные результаты тестов показывают, что для нескольких параметров, стоящих первыми в соответствии с упорядочением, можно добиться относительно небольшой разности между соответствующими им компонентами Ω' и Ω^0. При этом для остальных параметров эта разность велика, и часто компонент Ω' равен компоненту Ω^*. Это связано с приближением к границе области ГПЗ за счет первых параметров.

Для того чтобы избежать указанного эффекта для параметра, не являющегося одним из первых, его значение в начальном приближении следует задавать ближе к значению из Ω^0.

Для отдельных тестов, результаты которых не приводятся по причине ограниченного объема статьи, подход с упорядочением по нежелательности изменений дает существенно более низкую загруженность процессора, чем простейший подход (на одном из тестов это соответственно $75,93\%$ и $97,08\%$).

§ 6. Применение подходов в рамках среды Диана

Разрабатываемая в лаборатории вычислительных комплексов ВМиК МГУ среда имитационного моделирования ДИАНА включает в себя средства моделирования функционирования распределенных ВсВС [6]. В состав возможностей среды ДИАНА входит оценка времени выполнения последовательных

Табл. 1. Пропускная способность шины BcBC

BW^0	BW^*	BW'
5	10	9

Табл. 3. Загруженность процессора

U^0	U''	U'
192.63%	92.19%	96.28%

Табл. 2. Параметры набора задач

Задача	P	Cc^0	Cc^*	Cc'	T^0	T^*	T'	L	Bt^0	Bt'	D	R^0	R'
Обмен1	9	20	5	6	100	130	126	35	7	4	50	36	15
Обмен2	8	20	5	6	100	130	126	40	8	5	50	64	26
Обмен3	7	20	10	11	90	120	105	45	9	5	60	93	45
Обмен4	6	20	10	11	120	140	126	45	9	5	60	>1000	53
Задача5	5	35	25	26	100	130	126	0	0	0	100	>1000	79
Задача6	4	35	30	31	400	420	400	0	0	0	600	>1000	126
Задача7	3	40	30	31	400	450	400	0	0	0	400	>1000	236
Задача8	2	30	20	21	200	250	242	0	0	0	400	>1000	336
Задача9	1	25	15	16	200	250	233	0	0	0	400	>1000	399

Табл. 4. Пропускная способность шины BcBC, упорядочение 1

BW^0	BW^*	BW'
5	10	10

Табл. 6. Загруженность процессора, упорядочение 1

U^0	U'
192.63%	96.67%

Табл. 5. Параметры набора задач, упорядочение 1

Задача	P	Cc^0	Cc^*	Cc'	T^0	T^*	T'	L	Bt^0	Bt'	D	R^0	R'
Обмен1	9	20	5	11	100	130	100	35	7	4	50	36	20
Обмен2	8	20	5	5	100	130	130	40	8	4	50	64	29
Обмен3	7	20	10	10	90	120	109	45	9	5	60	93	44
Обмен4	6	20	10	10	120	140	130	45	9	5	60	>1000	54
Задача5	5	35	25	25	100	130	130	0	0	0	100	>1000	79
Задача6	4	35	30	30	400	420	400	0	0	0	600	>1000	188
Задача7	3	40	30	30	400	450	400	0	0	0	400	>1000	248
Задача8	2	30	20	20	200	250	217	0	0	0	400	>1000	347
Задача9	1	25	15	15	200	250	250	0	0	0	400	>1000	400

Табл. 7. Пропускная способность шины BcBC, упорядочение 2

BW^0	BW^*	BW'
5	10	10

Табл. 9. Загруженность процессора, упорядочение 2

U^0	U'
192.63%	98.92%

Табл. 8. Параметры набора задач, упорядочение 2

Задача	P	Cc^0	Cc^*	Cc'	T^0	T^*	T'	L	Bt^0	Bt'	D	R^0	R'
Обмен1	9	20	5	5	100	130	130	35	7	4	50	36	14
Обмен2	8	20	5	5	100	130	130	40	8	4	50	64	23
Обмен3	7	20	10	10	90	120	98	45	9	5	60	93	38
Обмен4	6	20	10	10	120	140	130	45	9	5	60	>1000	48
Задача5	5	35	25	25	100	130	130	0	0	0	100	>1000	73
Задача6	4	35	30	30	400	420	400	0	0	0	600	>1000	118
Задача7	3	40	30	30	400	450	400	0	0	0	400	>1000	221
Задача8	2	30	20	23	200	250	200	0	0	0	400	>1000	244
Задача9	1	25	15	25	200	250	200	0	0	0	400	>1000	365

программ на вычислителях различных архитектур без использования эмуляции на уровне машинных команд [7], а также оценка времени передачи сообщений через коммуникационную среду.

При помощи среды ДИАНА по заданному программному коду задачи с достаточно простой структурой графа управления (например, задачи цифровой обработки сигналов) может быть получена оценка наихудшего (максимального) времени выполнения этой задачи на данном вычислителе. Значения наихудшего времени выполнения задач могут использоваться в качестве исходных данных для модуля генерации рекомендаций.

Использование среды ДИАНА вместе с модулем генерации рекомендаций позволит определить наличие свойства гарантированной планируемости у набора задач с известным исходным кодом *без использования реальной аппаратной платформы*, на которой должны выполняться задачи. Это позволит на ранних этапах разработки ВсВС оценить возможность использования различных аппаратных платформ в составе ВсВС для выполнения заданного НЗ. Кроме того, разработчик сможет определить, в какой мере необходимо оптимизировать задачи из НЗ для того, чтобы при выполнении НЗ на ВсВС гарантированно не возникало превышения задачами директивных сроков.

§ 7. Применимость подходов для более широкого класса встроенных систем

В настоящее время автором проводится исследование применимости предлагаемого подхода для распределенных ВсВС, в состав которых входят несколько ЦВМ. Результаты исследований позволяют говорить о возможности использования схожих подходов для распределенных ВсВС, в которых узлы соединены каналами точка-точка или шинами с разделением доступа по времени.

§ 8. Заключение

В работе рассмотрена задача автоматической генерации рекомендаций по количественному изменению параметров набора задач (НЗ) и производительности распределенной ВсВС с целью достижения гарантированной выполнимости задач на ВсВС в пределах директивных сроков. Представлена формальная постановка задачи, основанная на формулах гарантированной планируемости задач. Предложены и рассмотрены на

примере подходы к решению задачи. Эти подходы к решению данной задачи являются новыми.

Полученные результаты могут применяться в процессе разработки ВсВС для оценки возможности использования различных аппаратных платформ в составе ВсВС для выполнения заданного НЗ. Кроме того, разработчик ВсВС сможет определить, в какой мере необходимо оптимизировать задачи из НЗ, чтобы при их выполнении на ВсВС гарантированно не возникало превышения директивных сроков.

В перспективе планируется осуществить адаптацию подходов для наиболее широкого класса архитектур ВсВС.

ЛИТЕРАТУРА

1. Locke C. Software Architecture for Hard Real-Time Applications: Cyclic Executives vs. Fixed Priority Schedulers. Real-time Systems. V 4 1992.
2. Bate I.J. Scheduling and Timing Analysis for Safety Critical Real-Time Applications. PhD Thesis, University of York, 1998.
3. Tindell K., Burns A., Wellings A.J. An Extendible Approach for Analysing Fixed-Priority Hard Real-Time Tasks. Journal of Real-Time Systems. 6(2) 1994.
4. Stewart D.B., Volpe R.A., Khosla P.K. Design of Dynamically Reconfigurable Real-time Software Using Port-Based Objects. IEEE Transactions on software engineering. **23**, No 12 1997.
5. Sjodin M., Hansson H. Improved Response Time Analysis Calculations. Proceedings of the 19-th IEEE Real-Time Systems Symposium. 1998.
6. Bakhmurov A., Kapitonova A., Smeliansky R. DYANA: An Environment for Embedded System Design and Analysis. Proceedings of 32-nd Annual Simulation Simposium. 1999.
7. Балашов В. В., Капитонова А.П., Костенко В.А., Смелянский Р.Л., Ющенко Н.В. Метод и средства оценки времени выполнения оптимизированных программ. М.: Программирование, 2000.

УДК 517.929

О НЕКОТОРЫХ ГИБРИДНЫХ СИСТЕМАХ

Вишневская Е. А.

Рассмотрим следующую систему интегральных уравнений при $t \in \Delta = [0, T]$:

$$x(t) = x_0 + \int_0^t f(s, x(s), y(s))ds, \tag{1}$$

$$y(t) = y_0 + \int_0^T g(t, s, x(s), y(s))ds, \tag{2}$$

где $x \in R^k$, $y \in R^l$, функции $f(s, x, y)$, $g(t, s, x, y)$ предполагаются непрерывными на $\Delta \times R^k \times R^l$ и $\Delta \times \Delta \times R^k \times R^l$ соответственно. Подобные системы возникают при изучении так называемых гибридных систем обыкновенных дифференциальных и интегральных уравнений, рассматриваемых, например, в [1, с. 151].

Для системы (1), (2) ставится вопрос о существовании и единственности решений $z(t) = \{x(t), y(t)\}^T$ в пространстве $L_2(\Delta, R^n)$ n-мерных векторных функций, определенных на отрезке Δ и суммируемых там вместе с квадратом модуля, где $n = k + l$. Дополнительно относительно функций $f(s, x, y)$ и $g(t, s, x, y)$ требуется выполнение следующих двух условий:

1^0. k-мерная векторная функция $f(s, x, y)$ удовлетворяет условию Липшица вида:

$$|f(s, x', y') - f(s, x'', y'')| \leqslant a_1(s)|x' - x''| + a_2(s)|y' - y''|$$

при $\forall x', x'' \in R^k$, $\forall y', y'' \in R^l$, $\forall s \in \Delta$, где $|\cdot|$ означает длину вектора в соответствующем действительном арифметическом пространстве (R^k или R^l), скалярные функции $a_1(s)$, $a_2(s)$ неотрицательны и непрерывны на отрезке Δ;

2^0. l-мерная векторная функция $g(t,s,x,y)$ допускает представление вида:

$$g(t,s,x,y) = K(t,s)h(s,x,y), \tag{3}$$

где $K(t,s)$ — матричная функция размерности $l \times l$, зависящая от аргументов непрерывным образом на $\Delta \times \Delta$, $h(s,x,y)$ — l-мерная векторная функция, непрерывная по совокупности переменных и удовлетворяющая условию Липшица вида

$$|h(s,x',y') - h(s,x'',y'')| \leqslant a_3(s)|x' - x''| + a_4(s)|y' - y''|, \tag{4}$$

при $\forall x', x'' \in R^k$, $\forall y', y'' \in R^l$, $\forall s \in \Delta$, где $a_3(s)$, $a_4(s)$ — скалярные неотрицательные функции, непрерывные на отрезке Δ.

Учитывая представление (3) и условие (4), получаем, что функция $g(t,s,x,y)$ удовлетворяет условию Липшица вида

$$|g(t,s,x',y') - g(t,s,x'',y'')| \leqslant b_1(t,s)|x' - x''| + b_2(t,s)|y' - y''|,$$

при $\forall x', x'' \in R^k$, $\forall y', y'' \in R^l$, $\forall t, s \in \Delta$, где

$$b_1(t,s) = \|K(t,s)\|a_3(s), \tag{5}$$

$$b_2(t,s) = \|K(t,s)\|a_4(s), \tag{6}$$

$$\|K(t,s)\| = \max_{\xi \in R^n, |\xi|=1} |K(t,s)\xi|.$$

Таким образом, скалярные функции $b_1(t,s)$, $b_2(t,s)$ неотрицательны и зависят от аргументов непрерывным образом на $\Delta \times \Delta$.

С учетом представления (3) для функции $g(t,s,x,y)$ система (1), (2) принимает вид

$$\begin{cases} x(t) = x_0 + \displaystyle\int_0^t f(s, x(s), y(s))ds, \\ y(t) = y_0 + \displaystyle\int_0^T K(t,s)h(s, x(s), y(s))ds, \end{cases}$$

которую мы и будем рассматривать в дальнейшем. Для получения результата используется подход, предложенный в статье [2, с. 249] и основанный на применении обобщенной векторной нормы для построения обобщенного инвариантного множества. Обобщенная векторная норма для рассматриваемой

задачи вводится следующим образом:

$$\|z(\cdot)\|_{\mathfrak{L}_2} = \begin{pmatrix} \|x(\cdot)\|_2 \\ \|y(\cdot)\|_2 \end{pmatrix}, \tag{7}$$

где

$$\|u(\cdot)\|_2 = \left(\int_0^T |u(s)|^2 ds \right)^{\frac{1}{2}},$$

$u(\cdot)$ — векторная функция соответствующей размерности, т.е. каждому $w \in L_2(\Delta, R^n)$ поставлен в соответствие неотрицательный вектор $\|w\|_{\mathfrak{L}_2} \in R^2$. Пространство $L_2(\Delta, R^n)$ n-мерных векторных функций, определенных на отрезке Δ и суммируемых там вместе с квадратом модуля, с обобщенной векторной нормой, введенной таким образом (см. (7)), обозначим как $\mathfrak{L}_2$.

С учетом представления (3) для функции $g(t, s, x, y)$ введем следующие операторы:

$$\Phi_1 z(\cdot) = \left\{ x_0 + \int_0^t f(s, x(s), y(s)) ds, \ t \in \Delta \right\} : L_2(\Delta, R^n) \mapsto L_2(\Delta, R^k), \tag{8}$$

$$\Phi_2 z(\cdot) = \left\{ y_0 + \int_0^T K(t, s) h(s, x(s), y(s)) ds, \ t \in \Delta \right\} : L_2(\Delta, R^n) \mapsto L_2(\Delta, R^l), \tag{9}$$

$$\Phi z(\cdot) = \begin{pmatrix} \Phi_1 z(\cdot) \\ \Phi_2 z(\cdot) \end{pmatrix} : \ L_2(\Delta, R^n) \mapsto L_2(\Delta, R^n) \tag{10}$$

и рассмотрим операторное уравнение

$$z(\cdot) = \Phi z(\cdot) \tag{11}$$

в $\mathfrak{L}_2$.

Желая применить теорему 1 из [2, с. 250], найдем обобщенный шар (ограниченное замкнутое выпуклое множество) из пространства $\mathfrak{L}_2$, который оператор Φ преобразует в себя. Для этого (см. [2, с. 250]) предпочтительно иметь оценку для $\forall z'(\cdot), z''(\cdot) \in L_2(\Delta, R^n)$:

$$\|\Phi z'(\cdot) - \Phi z''(\cdot)\|_{\mathfrak{L}_2} \prec S \|z'(\cdot) - z''(\cdot)\|_{\mathfrak{L}_2}, \tag{12}$$

где S — неотрицательная матрица (т.е. матрица, все элементы которой неотрицательны) размерности 2×2, а символ "$\prec$"

означает, что элементы вектора слева меньше либо равны соответствующих им элементов вектора справа. Пусть

$$\|\Phi\Theta\|_{\mathfrak{L}_2} \prec \eta, \tag{13}$$

где Θ — нулевая функция, а η — двумерный вектор. Из (12), (13) для $\forall z(\cdot) \in L_2(\Delta, R^n)$ следует неравенство

$$\|\Phi z(\cdot)\|_{\mathfrak{L}_2} \prec S\|z(\cdot)\|_{\mathfrak{L}_2} + \eta.$$

Если S является a-матрицей, то существует матрица $(I-S)^{-1}$ и она является неотрицательной (т.е. все элементы матрицы неотрицательны). Здесь I — единичная матрица.

Определение. *Матрица $D = (d_{ij})_{i,j=1}^n$ называется a-матрицей, если она неотрицательна и у матрицы $(I - D)$ последовательные главные миноры положительны:*

$$1 - d_{11} > 0, \quad \begin{vmatrix} 1-d_{11} & -d_{12} \\ -d_{21} & 1-d_{22} \end{vmatrix} > 0, \dots$$

$$\dots, \begin{vmatrix} 1-d_{11} & -d_{12} & \dots & -d_{1n} \\ -d_{21} & 1-d_{22} & \dots & -d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -d_{n1} & -d_{n2} & \dots & 1-d_{nn} \end{vmatrix} > 0. \tag{14}$$

Обозначим через $r(D)$ спектральный радиус матрицы D. Тогда условие (14) будет равносильно неравенству $r(D) < 1$, что равносильно условию

$$D^m \xrightarrow[m\to\infty]{} 0. \tag{15}$$

Так как по формуле Гельфанда

$$r(D) = \lim_{m\to\infty} \sqrt[m]{\|D^m\|},$$

то при некотором m_0 для $\forall m > m_0$ верно $\|D^m\|^{\frac{1}{m}} \leqslant q \in (r(D), 1)$ и $\|D^m\| \leqslant q^m$, откуда при $m \to \infty$ и следует (15). Продолжая рассмотрение нашей задачи, фиксируем некоторый положительный вектор $\rho \in R^2$ такой, что $(I-S)^{-1}\eta \prec \rho$. Тогда $S\rho + \eta \prec \rho$ и обобщенный шар

$$F = \{u(\cdot) \in L_2(\Delta, R^n) : \|u(\cdot)\|_{\mathfrak{L}_2} \prec \rho\} \tag{16}$$

и будет искомым для оператора Φ. Действительно,

$$\|\Phi z(\cdot)\|_{\mathfrak{L}_2} \prec S\|z(\cdot)\|_{\mathfrak{L}_2} + \eta \prec S\rho + \eta \prec \rho.$$

Займемся построением матрицы S. Для $\forall z'(\cdot), z''(\cdot) \in L_2(\Delta, R^n)$ имеем

$$\|\Phi z'(\cdot) - \Phi z''(\cdot)\|_{\mathfrak{L}_2} = \begin{pmatrix} \|\Phi_1 z'(\cdot) - \Phi_1 z''(\cdot)\|_2 \\ \|\Phi_2 z'(\cdot) - \Phi_2 z''(\cdot)\|_2 \end{pmatrix} =$$

$$= \begin{pmatrix} \left\| \int_0^t f(s, z'(s))ds - \int_0^t f(s, z''(s))ds \right\|_2 \\ \left\| \int_0^T K(t,s)h(s, z'(s))ds - \int_0^T K(t,s)h(s, z''(s))ds \right\|_2 \end{pmatrix} =$$

$$= \begin{pmatrix} \left(\int_0^T \left| \int_0^t \left[f(s, z'(s)) - f(s, z''(s)) \right] ds \right|^2 dt \right)^{\frac{1}{2}} \\ \left(\int_0^T \left| \int_0^T K(t,s) \left[h(s, z'(s)) - h(s, z''(s)) \right] ds \right|^2 dt \right)^{\frac{1}{2}} \end{pmatrix} \prec$$

$$\prec \begin{pmatrix} \left(\int_0^T \left(\int_0^t |f(s, z'(s)) - f(s, z''(s))| ds \right)^2 dt \right)^{\frac{1}{2}} \\ \left(\int_0^T \left(\int_0^T \|K(t,s)\| \cdot |h(s, z'(s)) - h(s, z''(s))| ds \right)^2 dt \right)^{\frac{1}{2}} \end{pmatrix} \prec$$

$$\prec \begin{pmatrix} \left(\int_0^T \left(\int_0^t a_1(s)|x'(s) - x''(s)|ds + \right. \right. \\ \left. \left. + \int_0^t a_2(s)|y'(s) - y''(s)|ds \right)^2 dt \right)^{\frac{1}{2}} \\ \left(\int_0^T \left(\int_0^T b_1(t,s)|x'(s) - x''(s)|ds + \right. \right. \\ \left. \left. + \int_0^T b_2(t,s)|y'(s) - y''(s)|ds \right)^2 dt \right)^{\frac{1}{2}} \end{pmatrix} \prec$$

(далее используем неравенство

$$\left(\int_0^T (|f_1(s)| + |f_2(s)|)^2 ds \right)^{\frac{1}{2}} \leqslant \left(\int_0^T |f_1(s)|^2 ds \right)^{\frac{1}{2}} + \left(\int_0^T |f_2(s)|^2 ds \right)^{\frac{1}{2}},$$

справедливое для элементов $f_1(s)$, $f_2(s)$ из $L_2(\Delta, R^k)$ и

$L_2(\Delta, R^l)$ соответственно)

$$\prec \begin{pmatrix} \left(\int_0^T \left(\int_0^t a_1(s)|x'(s)-x''(s)|ds\right)^2 dt\right)^{\frac{1}{2}} + \\ \qquad + \left(\int_0^T \left(\int_0^t a_2(s)|y'(s)-y''(s)|ds\right)^2 dt\right)^{\frac{1}{2}} \\ \left(\int_0^T \left(\int_0^T b_1(t,s)|x'(s)-x''(s)|ds\right)^2 dt\right)^{\frac{1}{2}} + \\ \qquad + \left(\int_0^T \left(\int_0^T b_2(t,s)|y'(s)-y''(s)|ds\right)^2 dt\right)^{\frac{1}{2}} \end{pmatrix} \prec$$

(теперь применим неравенство Коши—Буняковского)

$$\prec \begin{pmatrix} \left(\int_0^T \left(\int_0^t a_1^2(s)\,ds \int_0^t |x'(s)-x''(s)|^2 ds\right)dt\right)^{\frac{1}{2}} + \\ \qquad + \left(\int_0^T \left(\int_0^t a_2^2(s)\,ds \int_0^t |y'(s)-y''(s)|^2 ds\right)dt\right)^{\frac{1}{2}} \\ \left(\int_0^T \left(\int_0^T b_1^2(t,s)\,ds \int_0^T |x'(s)-x''(s)|^2 ds\right)dt\right)^{\frac{1}{2}} + \\ \qquad + \left(\int_0^T \left(\int_0^T b_2^2(t,s)\,ds \int_0^T |y'(s)-y''(s)|^2 ds\right)dt\right)^{\frac{1}{2}} \end{pmatrix} \prec$$

$$\prec \begin{pmatrix} \left(\int_0^T \int_0^t a_1^2(s)\,dsdt\right)^{\frac{1}{2}} \left(\int_0^T |x'(s)-x''(s)|^2 ds\right)^{\frac{1}{2}} + \\ \qquad + \left(\int_0^T \int_0^t a_2^2(s)\,dsdt\right)^{\frac{1}{2}} \left(\int_0^T |y'(s)-y''(s)|^2 ds\right)^{\frac{1}{2}} \\ \left(\int_0^T \int_0^T b_1^2(t,s)\,dsdt\right)^{\frac{1}{2}} \left(\int_0^T |x'(s)-x''(s)|^2 ds\right)^{\frac{1}{2}} + \\ \qquad + \left(\int_0^T \int_0^T b_2^2(t,s)\,dsdt\right)^{\frac{1}{2}} \left(\int_0^T |y'(s)-y''(s)|^2 ds\right)^{\frac{1}{2}} \end{pmatrix} =$$

$$= S\|z'(\cdot) - z''(\cdot)\|_{\mathfrak{L}_2},$$

где через S обозначена матрица (см. также (5), (6))

$$S = \begin{pmatrix} \left(\int_0^T \left(\int_0^t a_1^2(s)ds\right)dt\right)^{\frac{1}{2}}, & \left(\int_0^T \left(\int_0^t a_2^2(s)ds\right)dt\right)^{\frac{1}{2}} \\ \left(\int_0^T \int_0^T b_1^2(t,s)\,dsdt\right)^{\frac{1}{2}}, & \left(\int_0^T \int_0^T b_2^2(t,s)\,dsdt\right)^{\frac{1}{2}} \end{pmatrix}. \quad (17)$$

Таким образом, неравенство (12) получено. Имеем далее (см. (8), (9))

$$\|\Phi\Theta\|_{\mathfrak{L}_2} = \begin{pmatrix} \|\phi_1\theta\|_2 \\ \|\phi_2\theta\|_2 \end{pmatrix} = \begin{pmatrix} \left\|\int_0^t f(s,0)ds + x_0\right\|_2 \\ \left\|\int_0^T K(t,s)h(s,0)ds + y_0\right\|_2 \end{pmatrix} \prec$$

$$\prec \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \eta,$$

где c_1, c_2 — достаточно большие константы.

С помощью известных результатов нетрудно показать, что нелинейный оператор Φ (см. (8)–(10)) является вполне непрерывным оператором в пространстве $\mathfrak{L}_2$, т.е. непрерывным и компактным в $\mathfrak{L}_2$. (Свойство компактности оператора, действующего из некоторого пространства E_1 в E_2, означает, что он преобразует ограниченные множества пространства E_1 в предкомпактные множества из E_2.)

По теореме 1 из [2, с. 250], если S (см. (17)) является a-матрицей, то оператор Φ имеет в обобщенном шаре F (см. (16)) по крайней мере одну неподвижную точку, т.е. существует функция $z(\cdot) \in F$, которая является решением уравнения (11).

Замечание. Непосредственно из определения a-матрицы можно получить, что *для того чтобы некоторая неотрицательная матрица* $\Lambda = \begin{pmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{pmatrix}$ *являлась a-матрицей, необходимо и достаточно выполнение неравенств*:

$$\alpha_1 < 1, \quad (1-\alpha_1)(1-\beta_2) - \alpha_2\beta_1 > 0. \quad (18)$$

Также отметим, что *неравенства* (18) *можно заменить следующими*:

$$\alpha_1 < 1, \quad \beta_2 < 1, \quad (1-\alpha_1)(1-\beta_2) - \alpha_2\beta_1 > 0.$$

Нами доказана следующая теорема, обобщающая теорему 2 из [1, с. 155] на $\mathfrak{L}_2$.

Теорема 1. *Пусть выполнены следующие требования*

1. *Функции* $f(t,x,y)$, $g(t,s,x,y)$ *в* (1), (2) *являются непрерывными на* $\Delta\times R^k\times R^l$ *и* $\Delta\times\Delta\times R^k\times R^l$ *соответственно.*
2. *Функции* $f(t,x,y)$, $g(t,s,x,y)$ *удовлетворяют предположениям* 1^0, 2^0.
3. *Для элементов матрицы* $S=\begin{pmatrix}\alpha_1 & \alpha_2\\ \beta_1 & \beta_2\end{pmatrix}$, *где*

$$\alpha_i=\left(\int_0^T\left(\int_0^t a_i^2(s)ds\right)dt\right)^{\frac{1}{2}},\quad \beta_i=\left(\int_0^T\int_0^T b_i^2(t,s)\,dsdt\right)^{\frac{1}{2}},$$
$$i=1,2,$$
$$b_1(t,s)=\|K(t,s)\|a_3(s),\qquad b_2(t,s)=\|K(t,s)\|a_4(s),$$

выполнены неравенства:

$$\alpha_1<1,\quad (1-\alpha_1)(1-\beta_2)-\alpha_2\beta_1>0.$$

Тогда существует решение уравнения (11) *в* $\mathfrak{L}_2$, *которое одновременно будет решением задачи* (1), (2) *в* $L_2(\Delta,R^n)$.

Теорема утверждает существование решения. Но при наших условиях теоремы решение будет единственным. Покажем это. Пусть есть два различных решения $z_1(\cdot)$, $z_2(\cdot)$, т.е., $\|z_1(\cdot)-z_2(\cdot)\|_{\mathfrak{L}_2}\neq 0$ и $z_1(\cdot)=\Phi z_1(\cdot)$, $z_2(\cdot)=\Phi z_2(\cdot)$. Тогда в силу того, что S (см. (17)) — a-матрица, неравенства (12) и условия (15) получаем

$$\begin{aligned}\|z_1(\cdot)-z_2(\cdot)\|_{\mathfrak{L}_2}&=\|\Phi z_1(\cdot)-\Phi z_2(\cdot)\|_{\mathfrak{L}_2}\prec\\ &\prec S\|z_1(\cdot)-z_2(\cdot)\|_{\mathfrak{L}_2}\prec\\ &\prec S^2\|z_1(\cdot)-z_2(\cdot)\|_{\mathfrak{L}_2}\prec\\ &\prec\ldots\prec\\ &\prec S^m\|z_1(\cdot)-z_2(\cdot)\|_{\mathfrak{L}_2}\xrightarrow[m\to\infty]{}0,\end{aligned}$$

что противоречит предположению о существовании двух различных решений уравнения (11). Этот результат отражает

Теорема 2. *Пусть выполнены условия* 1 – 3 *предыдущей теоремы. Тогда существует единственное решение уравнения* (11) *в* $\mathfrak{L}_2$, *которое одновременно будет решением задачи* (1), (2) *в* $L_2(\Delta,R^n)$. *Его можно получить методом последовательных приближений, начиная с* $\forall z_0(\cdot)\in L_2(\Delta,R^n)$, *и оно*

удовлетворяет оценке:

$$\|z(\cdot)\|_{\mathfrak{L}_2} \prec (I - S)^{-1} \|\Phi\Theta\|_{\mathfrak{L}_2},$$

где Θ — *нулевая функция.*

ЛИТЕРАТУРА

1. Никольский М.С., Франко Х. *О гибридных системах*// Вестник Российского университета дружбы народов. Серия Математика. **6**. 1999. С. 151–158.
2. Перов А.И., Кибенко А.В. *Об одном общем методе исследования краевых задач*// Известия АН СССР. Серия математическая. **30**. 1966. С. 249–264.

УДК 517.957

О НОВЫХ ПОДХОДАХ К ПРОБЛЕМЕ УПРАВЛЕНИЯ ХАОСОМ

А.В. Дернов

§ 1. Введение

Во многих нелинейных динамических системах наблюдается явление самоорганизации [1]. Под этим понимается наличие решений, образующих некие структуры в фазовом пространстве: неподвижные точки, замкнутые траектории, инвариантные торы, а также другие, более сложные геометрические образования. Если эти структуры имеют простой вид, например, устойчивый фокус или устойчивый предельный цикл, мы говорим, что в системе присутствует регулярное поведение. Наличие устойчивого инвариантного тора, если его размерность невелика, можно тоже иногда отнести к регулярному поведению [2]. Однако, существует момент, когда поведение системы уже нельзя назвать регулярным (например, если в фазовом пространстве присутствует аттрактор, содержащий внутри себя плотное множество неустойчивых циклов). В этом случае можно говорить о присутствии в системе динамического хаоса.

Изучение динамического хаоса началось с открытия явления турбулентности в течении жидкости. Следующим этапом было открытие знаменитой системы Лоренца [3] путем редукции уравнения Навье—Стокса для атмосферных потоков к системе трех обыкновенных дифференциальных уравнений. К настоящему моменту хаос найден в огромном количестве динамических систем. Он наблюдается как в диссипативных системах, обладающих аттракторами, так и в консервативных, фазовый объем которых остается постоянным. По своей физической сути такие системы могут быть самыми различными:

движение небесных тел, биллиарды, динамика популяций, химическая кинетика, гидродинамика, экономические процессы.

Со временем было осознано, что присутствие хаоса является необходимым условием работы сложных систем. В связи с этим, в теории управления возник новый раздел — управление хаосом. Первоначально под управлением хаосом понималась задача: как, переходя с одной траектории хаотического аттрактора на другую путем малых возмущений, попасть в окрестность заданной неустойчивой периодической траектории [4]. Затем стала решаться задача: как, находясь в окрестности неустойчивой периодической траектории, приблизиться к ней на сколь угодно малое расстояние [5]. Сейчас проблема управления хаосом включает в себя задачу получения информации о других неустойчивых структурах в фазовом пространстве: инвариантных кривых отображений, инвариантных торах потоков. Подходы к решению этих задач предлагаются в настоящей работе.

§ 2. Описание методики

В работе [5] были предложены методы локализации и стабилизации неустойчивых неподвижных точек и циклов отображений, систем обыкновенных дифференциальных уравнений, а также уравнений с запаздывающим аргументом. Каждый из этих методов сводится к построению в пространстве большей размерности некоторой динамической системы, для которой заданная неустойчивая периодическая траектория исходной хаотической системы является проекцией ее асимптотически орбитально устойчивой периодической траектории.

Однако следует заметить, что в окрестности периодической траектории потоку можно взаимно-однозначно поставить в соответствие отображение: для этого достаточно рассмотреть сечение Пуанкаре. При этом орбитальная устойчивость цикла будет соответствовать устойчивости неподвижной точки отображения, а мультипликаторы этого цикла будут собственными значениями отображения. Таким образом, задачи стабилизации цикла и точки сводятся друг к другу. Рассуждая аналогично, можно рассечь плоскостью инвариантный тор и получить соответствующую ему инвариантную замкнутую кривую отображения. Поэтому, чтобы стабилизировать неустойчивый инвариантный двумерный тор потока, достаточно стабилизировать инвариантную кривую отображения. Задача стабилизации инвариантных кривых пока не решена, однако можно в определеннном смысле приблизиться к ее решению. Для этого нужно рассматривать инвариантную кривую, как неподвиж-

ную точку некоторого отображения бесконечномерного пространства замкнутых кривых в себя. После этого можно аппроксимировать данное пространство конечномерным и, повысив его размерность, стабилизировать неподвижную точку методом, предложенным в [5]. В результате мы получим некое приближение исходной неустойчивой структуры, что может представлять важную информацию. Стоит отметить, что метод стабилизации неподвижной точки отображения является робастным, а также довольно прост в программной реализации.

§ 3. Применение методики для стабилизации неустойчивых циклов

Предложенный выше подход был успешно применен для стабилизации неустойчивых циклов в нескольких динамических системах. Переход от потоков к отображениям осуществлялся численно с помощью сечений Пуанкаре. Впервые был стабилизирован седловой цикл в системе Лоренца [3], а также несколько циклов в системе Чо [6], что существенно приблизило к пониманию структуры возникающего там аттрактора. Рассмотрим подробнее эти два примера.

Метод, предложенный в [5], применялся в следующей реализации. Пусть известно, что гладкое отображение

$$f(z) : R^n \to R^n$$

имеет неподвижную точку $z^0 = (z_1^0, \dots, z_n^0)$, которая неустойчива. Рассмотрим расширенное отображение

$$F(z,q) : R^{n+1} \to R^{n+1},$$
$$F(z,q) = \begin{pmatrix} f(z) + \varepsilon q \\ f_1(z_1, \dots, z_n) - z_1 + \beta q \end{pmatrix},$$
$$\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)^T, \quad \beta, q \in R.$$

Отображение $F(z,q)$ имеет неподвижную точку $(z^0, 0)$. Она будет асимптотически устойчива, если все собственные значения матрицы Якоби

$$\left(\frac{\partial F(z,q)}{\partial (z,q)}\right)_{z=z^0, q=0}$$

лежат строго внутри единичного круга. Чтобы все собственные значения этой матрицы равнялись нулю, необходимо и

достаточно выполнения $n+1$ условий:

$$\sum_i T_i^l = 0, \quad l = 1, \ldots, n+1,$$

сумма берется по всем главным минорам T_i^l порядка l. Для удовлетворения этих условий нужно найти ε и β, решив систему $n+1$ линейных уравнений. Мы не можем вычислить производные отображения $f(z)$ в точке z^0, поскольку не знаем эту точку. Однако, если нам каким-либо образом удалось попасть в ее достаточно малую окрестность, тогда в силу гладкости отображения $f(z)$ значения производных, вычисленные в точке из этой окрестности будут близки к значениям производных в точке z^0. Собственные значения матрицы Якоби полученной расширенной системы в точке z^0 хоть реально и не будут нулями, но будут лежать в единичном круге, что является достаточным условием асимптотической устойчивости. А это означает, что точка $(z^0, 0)$ может быть найдена с любой точностью итерационным процессом

$$(z_{n+1}, q_{n+1}) = F(z_n, q_n).$$

Заметим, что отображение Пуанкаре находится численно, значит, для нахождения производных нужно использовать конечные разности. Разностный шаг дифференцирования нужно согласовывать с шагом по времени в численном методе интегрирования исходных дифференциальных уравнений. Для стабилизации циклов в системах Лоренца и Чо использовалась двухточечная аппроксимация производной, имеющая второй порядок точности, а для интегрирования дифференциальных уравнений — метод Рунге—Кутта четвертого порядка. Для данных систем опытным путем было установлено, что метод работает наилучшим образом при одинаковом порядке этих разностных шагов.

Выше было отмечено, что для стабилизации неподвижной точки данным методом нужно попасть в достаточно малую ее окрестность. Это тоже довольно сложная задача. Рассмотрим ее решение в двух конкретных случаях. В системе Лоренца (рис. 1) аттрактор устроен так, что траектория “часто” подходит довольно близко к седловому циклу, имеющему довольно сильное сжатие вдоль устойчивого многообразия и довольно слабое растяжение вдоль неустойчивого. Последовательно осуществлялись попытки “зацепиться” за этот цикл, которые в конце-концов привели к успеху. На рис. 2 показан цикл, стабилизированный данным методом.

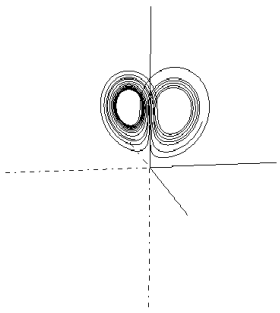

Рис. 1

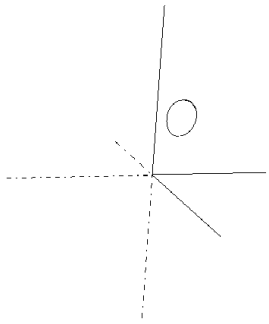

Рис. 2

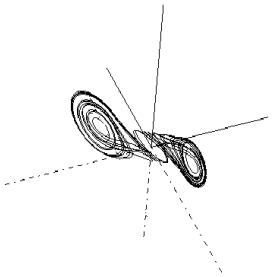

Рис. 3

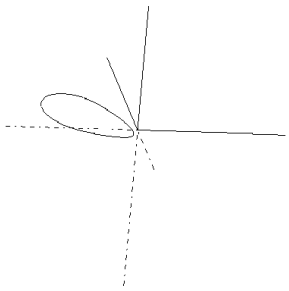

Рис. 4

В системе Чо стабилизировать циклы было несколько проще. Там наблюдается сценарий перехода к хаосу путем бесконечного числа бифуркаций удвоений периода. В конце сценария два симметрично расположенных аттрактора “сливаются”, образуя известный двухспиральный аттрактор Чо (рис. 3). Стабилизация циклов осуществлялась следующим образом. Если цикл асимптотически орбитально устойчив, то к нему можно приблизиться на сколь угодно малое расстояние и построить расширенную матрицу по алгоритму, указанному выше. Если сделать это в момент, близкий к бифуркации удвоения периода, то мы можем “удерживать” этот цикл с помощью расширенной системы еще и некоторое время после перехода значения параметра через точку бифуркации. Затем, стабилизировав цикл, уже ставший неустойчивым, можно пересчитать параметры расширенной системы и продвинуться еще дальше за точку бифуркации в сторону усложнения поведения системы. Данный метод позволил путем малых шагов по парамет-

ру продвигаться до момента, когда система образовывала уже двухспиральный аттрактор (рис. 4), т.е. стабилизировать неустойчивый цикл во время всего развития сценария перехода к хаосу.

§ 4. Некоторые подходы к стабилизации неустойчивых инвариантных кривых отображений

Как было сказано выше, инвариантную кривую можно рассматривать как неподвижную точку отображения некого пространства кривых в себя. Приближенное решение такой задачи можно получить, аппроксимировав это пространство конечномерным.

Рассмотрим пример, когда это можно сделать. Пусть кривые данного пространства плоские, замкнутые и ограничивают "вездную" область, т.е. можно выбрать точку, идущий из которой любой луч пересекает кривую только один раз. Предположим, что отображение не выводит нас из этого класса. Тогда, введя полярные координаты (ρ, ϕ), можно рассматривать это пространство как пространство периодических функций $\rho(\phi)$. После этого нужно аппроксимировать эти кривые конечным числом точек, например, с помощью кусочно-линейного восполнения или кубических сплайнов. После этого нужно построить индуцированное отображение этого конечномерного пространства в себя, производя пересчет точек. Затем следует воспользоваться методом, описанным в предыдущем пункте. Этот метод был проверен на следующей задаче:

$$\rho_{n+1} = 1 + (1 + \lambda)(\rho_n - 1), \qquad \phi_{n+1} = \phi_n + \alpha,$$

где α и λ — положительные числа. Видно, что отображение имеет неустойчивую инвариантную окружность $\rho = 1$. При аппроксимации использовалось кусочно-линейное восполнение, содержащее 16 точек. Приближенную многоугольником окружность удавалось стабилизировать при довольно существенных начальных возмущениях.

Стоит отметить возникающие при таком подходе сложности. Во первых, приходится считать все главные миноры матрицы, размерность которой равна числу точек аппроксимации плюс единица, что приводит к большому объему вычислений. Во вторых, имеет место проблема попадания в достаточно малую окрестность неустойчивой кривой, что гораздо сложнее, чем попадание в окрестность точки.

§ 5. Заключение

В настоящей работе был предложен подход, при котором несколько задач управления хаосом сводятся к стабилизации неустойчивой неподвижной точки отображения. С помощью него были впервые стабилизированы неустойчивые циклы в системах Лоренца и Чо. Также был предложен подход к стабилизации неустойчивых инвариантных кривых отображений.

ЛИТЕРАТУРА

1. Хакен Г. Синергетика. Иерархии неустойчивостей в самоорганизующихся системах и устройствах. М.: Мир, 1985
2. Дернов А.В. // Диф. уравнения. 2001. **37** № 11. С. 1554–1556.
3. Lorenz E.N. // J. Atmos. Sci. 1963. V. 20. P. 130.
4. Ott E., Grebogy C, Yorke J.A. // Phys. Rev. Lett. 1990. V 4. P. 1196–1199.
5. Магницкий Н.А., Сидоров С.В. // Дифф. уравнения. 1998. **35**. № 11. С. 1501–1509.
6. Shil'nikov L.P. // Int. J. Bifurcation and Chaos. 1994 V 4. N 3. P. 489–519.

ВЫДЕЛЕНИЕ ИСТОЧНИКА ИЗ СМЕСИ ЗВУКОВЫХ СИГНАЛОВ С ИСПОЛЬЗОВАНИЕМ БАЗЫ ДАННЫХ ЗАПИСЕЙ ЭТОГО ИСТОЧНИКА

Зинченко Е. Ю.

§ 1. Введение

Методы разделения сигналов имеют большую область применения. Одной из часто встречающихся задач является разделение речи людей в разговоре при наличии реверберации, шума. Эта задача может возникать при интервьюировании, при проведении телеконференций. Другая возможная задача состоит в выделении полезного сигнала, удалении шума, снятии реверберационных эффектов. Эти проблемы появляются при передачи голоса в мобильной связи, в ситуациях с наличием большого количества помех (человек говорит по телефону в движущемся транспорте). Эти методы могут быть использованы в медицине, например, при создании слуховых аппаратов.

Подходы к решению задачи разделения можно разбить на два класса. К первому классу относятся методы разделения речи без априорной информации, использующие только статистические свойства сигналов (BSS методы). Второй подход связан с использованием априорной информации. Например, если существует база данных речи спикеров, то задачу можно переформулировать как задачу идентификации записей базы данных в заданной смеси.

Все методы разделения сигналов типа BSS могут быть разделены на три этапа в общем виде [1]: *параметризация разделяющей системы, выбор критерия разделения и оптимизации критерия разделения.*

Работа выполнена при поддержке гранта РФФИ 02-07-90130, а также в рамках работы в студенческой лаборатории Intel-MSU, ВМК, МГУ.

Обсудим вопросы параметризации разделяющей системы. Модель звуковой смеси сигналов с учетом наличия отражения сигналов, шума в определенном временном отрезке $[t_{beg}, t_{end}]$ может быть записана в следующем виде:

$$x_j(t) = \sum_{i=1}^{N_i} \sum_{l=1}^{L} a_{ji}(l) s_i(t-l) + n(t), \tag{1}$$

где $n(t)$ — шум, $j = 1, 2, \ldots, N_j$, N_j — число сенсоров, N_i — число спикеров.

Данное преобразование сигнала $\vec{s}(t)$ есть линейный FIR фильтр с передаточной функцией $\{a_{ji}\}$. Задача разделения при отсутствии априорной информации формулируется как определение исходного сигнала $\vec{s}(t)$ по заданным сигналам $\vec{x}(t)$.

Сделаем следующие предположения.

1. Компоненты вектора $\vec{s}(t)$ представляют собой статистически независимые случайные процессы.

2. Матрица $\{a_{ji}\}$ не вырождена.

Тогда для случая смесей сигналов при отсутствии задержек

$$x_j(t) = \sum_{i=1}^{N_i} a_{ji} s_i(t) + n(t) \tag{2}$$

справедлива теорема о разрешимости задачи разделения при сделанных выше предположениях о свойствах исходных сигналов с точностью до скалярного множителя и порядка каналов [2, 3].

В работе [1] обсуждаются трудности, которые возникают при применении BSS методов для обработки реальных акустических сигналов.

Некоторые методы предполагают стационарность сигналов. Хотя на большом промежутке это неверное предположение, можно говорить о квазистационарности сигналов на малых промежутках времени. При этом процедура нахождения значения фильтров усложняется, так как усложняется функция критерия разделения.

При применении BSS методов предполагается, что число сенсоров не меньше числа источников. В работе [1] приводится пример, когда человек говорит в сильно реверберирующем помещении, например, в машине, где количество источников велико.

Стационарность положения людей также сильно влияет на свойства смесей. В [1] указывается, что поворот головы на 10–20 градусов оказывает большое влияние на вид передаточной функции между человеком и микрофонами.

Предложено большое количество BSS методов для разделения конволютивных моделей. Однако остается открытым вопрос об унификации критериев сравнения этих методов и создании единой базы данных для тестирования. Обсуждению этой проблемы посвящена работа [4].

Классические методы BSS представлены в работах [5, 6]. В работах [7, 8] предложены методы разделения источников с применением временной и спектральной фильтраций соответственно, где критерием разделимости служила сумма корреляций выходных сигналов разделяющей системы.

В данной работе предложен метод выделения речи человека из смеси с использованием базы данных записей речи людей. Задача разделения ставится как задача идентификации записей из базы данных в заданной смеси. Метод основан на применении линейного фильтра к заданной смеси звуковых сигналов. Длина применяемого фильтра определяется для голоса каждого человека в процессе обучения. Мы предполагаем, что ошибка фильтрации для смесей, содержащих сигнал идентифицируемого человека, будет меньше, чем для сигналов звуковых смесей, где голос данного человека не звучит. Этот метод свободен от ограничений, накладываемых BSS методами (статистическая независимость исходных сигналов, число сенсоров не меньше числа источников и т.д.).

§ 2. Постановка задачи

В комнате находится несколько человек. Установлен микрофон, записывающий речь людей, находящихся в комнате и говорящих одновременно. Сигнал, содержащий смесь речевых сигналов, будем обозначать как $x(t), t = t_{beg}, \ldots, t_{end}$.

Модель звуковой смеси сигналов с учетом наличия отражения сигналов, шума в определенном временном отрезке $t = t_{beg}, \ldots, t_{end}$ может быть записана в следующем виде:

$$x(t) = \sum_{i=1}^{N} \sum_{l=1}^{L} a_i(l) s_i(t-l) + n(t). \tag{3}$$

Задача заключается в построении алгоритма выделения заданного сигнала $s(t)$ из смеси звуковых сигналов $x(t)$. Задачу выделения сигнала $s(t)$ рассматриваем как построение оптимального FIR фильтра заданной длины для фильтрации поданного на вход сигнала смеси

Под FIR фильтром понимается следующее преобразование

сигнала:

$$\tilde{s}(t) = \sum_{l=0}^{L} w_l x(t-l). \tag{4}$$

Здесь L — длина фильтра.

Коэффициенты оптимального FIR фильтра должны минимизировать ошибку фильтрации:

$$\epsilon(L \mid s(t), x(t)) = \sqrt{\sum_{t \in [t_{beg}, t_{end}]} [s(t) - \sum_{l=0}^{L} w_l x(t-l)]^2} \longrightarrow \min. \tag{5}$$

Важную роль играет длина фильтра L. Этот параметр определяет сложность фильтрующей системы (4). Чем больше длина фильтра, тем меньше ошибка фильтрации. В предельном случае, когда длина окна равна количеству временных отсчетов сигнала, ошибка фильтрации $\epsilon(L)$ будет пренебрежимо мала. Но в этом случае фильтр станет *формирующим* в том смысле, что независимо от того, участвовал ли сигнал $s(t)$ при создании смеси $x(t)$ или нет, данный фильтр выделит из смеси $x(t)$ сигнал $s(t)$. Наша задача состоит в построении *разделяющего* фильтра, точность фильтрации которого зависела бы от наличия сигнала $s(t)$ в смеси $x(t)$. Таким образом, длину фильтра для выделения заданного источника нужно подобрать исходя из следующих двух ограничений:

1) ошибка фильтрации должна быть мала;

2) фильтр не должен стать формирующим.

Все вышесказанное требует строгой математической постановки.

Задан источник $s(t)$. Существует база данных обучения, в которой хранятся сигналы звуковых смесей $x_p(t)$, $p = 1, \ldots, N_p$, и индикаторы H_p, равные 1, если $s(t)$ звучал при создании смеси $x_p(t)$, 0 — иначе.

Пусть $I(\xi)$ — индикатор события ξ. Введем параметр ϵ_{th} — порог ошибки фильтрации. Если ошибка фильтрации заданного сигнала смеси $x(t)$ ниже этого порога, то сигнал $s(t)$ присутствует в смеси. Определим два функционала, описывающих два типа ошибки идентификации.

Функционал

$$P_{e1}(L,\epsilon_{th}) = \sum_{p=1}^{N_p} \frac{I((\epsilon(L \mid s(t), x_p(t)) < \epsilon_{th})\&(H_p = 0))}{N_p} \qquad (6)$$

характеризует количество записей звуковых смесей, не содержащих сигнал $s(t)$, в которых он неверно идентифицирован.

Функционал

$$P_{e2}(L,\epsilon_{th}) = \sum_{p=1}^{N_p} \frac{I((\epsilon(L \mid s(t), x_p(t)) > \epsilon_{th})\&(H_p = 1))}{N_p} \qquad (7)$$

характеризует количество записей звуковых смесей, содержащих сигнал $s(t)$, в которых он неверно не идентифицирован.

Тогда задачу обучения формулируем как задачу минимизации следующего функционала по параметрам L, ϵ_{th}:

$$P_e(L,\epsilon_{th}) = P_{e1}(L,\epsilon_{th})\left(\sum_{p=1}^{N_p} \frac{I(H_p = 0)}{N_p}\right)^{(-1)} + \\ + P_{e2}(L,\epsilon_{th})\left(\sum_{p=1}^{N_p} \frac{I(H_p = 0)}{N_p}\right)^{(-1)}. \qquad (8)$$

§ 3. Результаты экспериментов

Существует следующая тестовая база данных. Два человека произносят одновременно слова: “один”, “два”, . . . , “десять”. В базе данных хранятся 100 записей всех возможных пересечений. Данные записаны при частоте 44100 Hz, а потом частота сэмплирования была понижена до 7000 Hz.

На первом этапе сигналы разбиваются на N_w временных подокон. Длина окна T_w=0,18. Каждое временное окно представляет собой следующий промежуток: $T_k = [(k-1)T_w, kT_w]$, $k = 1, 2, \ldots, N_w$.

На каждом временном окне сигнал умножается на сглаживающую функцию окна (обычно используют окно Хэмминга). Таким образом, получаем следующий набор локализованных функций:

$$s_k(t) = s(t)g(t - (k - 0,5)T_w), \quad t \in T_k, \qquad (9)$$

$$x_k(t) = x(t)g(t - (k - 0,5)T_w), \quad t \in T_k. \qquad (10)$$

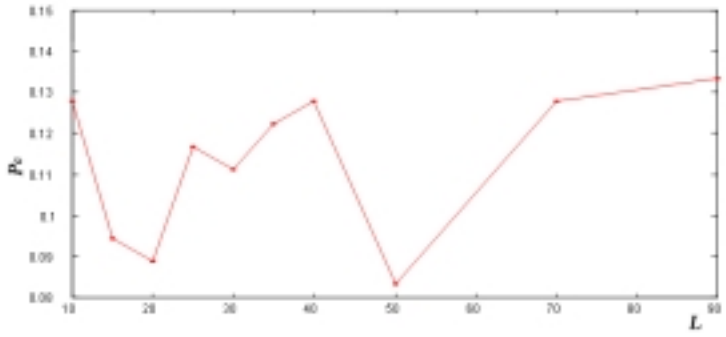

Рис. 1. Зависимость ошибки идентификации от длины фильтра L

К полученному набору сигналов $x_k(t)$, $s_k(t)$ применяется процедура удаления среднего значения энергии и нормализация. Убираем паузы в сигнале, отбрасывая из дальнейшего анализа те k , где энергия меньше некоторого порогового значения E_{th}. Множество индексов окон, содержащих информативный сигнал, обозначим K. На втором этапе для каждой локализованной функции $x_k(t)$ находится оптимальный FIR фильтр заданной длиной L в смысле наименьших квадратов.

Тогда ошибка фильтрации (SNR) оценивается, как

$$\epsilon(L \mid s(t), x(t)) = \sum_{k \in K} \frac{\epsilon(L \mid s_k(t), x_k(t))}{|K|}. \tag{11}$$

На рис. 1 изображена зависимость функционала ошибки идентификации от длины фильтра L. Минимальная ошибка 0.0833 достигается при длине фильтра 50.

На рис. 2–4 изображено несколько распределений точек $\epsilon(L \mid s(t), x(t))$ для разных значений $L = 10, 50, 90$. Точки соответствуют записям, не содержащим слово “четыре” первого человека. Крестики соответствуют записям, содержащим слово “четыре” первого человека. На рис. 3 видно, что наилучшее разделение происходит при длине фильтра $L = 50$.

§ 4. Заключение

Предложен метод выделения источника, использующий априорную информацию(база данных речи разделяемых людей). Данный метод свободен от ограничений, накладываемых BSS методами (достаточно сигнала с одного сенсора). Метод

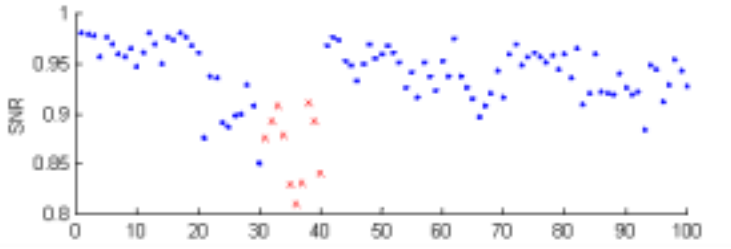

Рис. 2. Изменение распределения ошибки фильтрации при $L = 10$

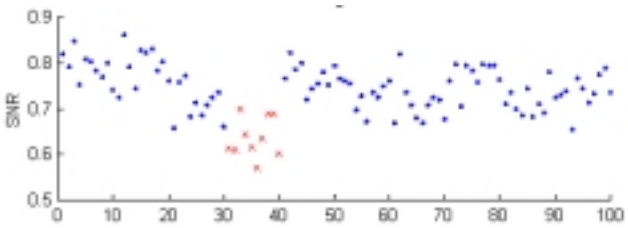

Рис. 3. Изменение распределения ошибки фильтрации при $L = 50$

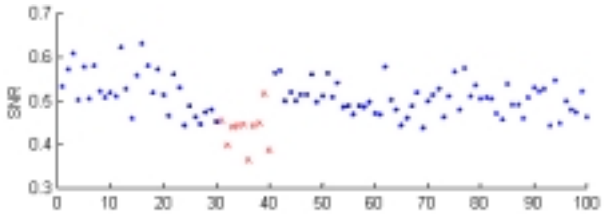

Рис. 4. Изменение распределения ошибки фильтрации при $L = 90$

показал хорошие результаты при тестировании. При выборе соответствующей длины фильтра соответствующий конкретному человеку функционал (11) можно рассматривать как меру близости между сигналами.

ЛИТЕРАТУРА

1. Torkkola K. Blind separation for audio signals — are we there yet?. Proc. Workshop on Independent Component Analysis and Blind Signal Separation, Jan 11–15, 1999, Aussois, France.
2. Cordosa J.-F. Blind signal separation: statistical principles// IEEE V 9. N 10. P. 2009–2025. 1998.
3. Comon P. Independent component analysis — a new concept?// Signal Processing. 36(3). P. 287–314. 1994.
4. Schobben D., Torkkola K., Smaragdis P. Evaluation of blind separation methods// Proceedings Int. Workshop Independent Component Analysis and Blind Signal Separation. Aussois, France. January 11–15. 1999. P. 261–266.
5. Bell A.J., Sejnowski T.J. An information-maximiation approach to blind separation and blind deconvolution// Neural Computation. 7(6) P. 1129–1159. 1995.
6. Cordosa J.-F., Laheld B. Equivariant adaptive source separation// IEEE Transactions on Signal Processing. 44(12) P. 3017–3030, December 1996.
7. Krongold B.S., Jones D.L. Blind Source Separation of Nonstationary Convolutively Mixed Signals// Proceedings of the 10th IEEE Workshop on Statistical Signal and Array Processing. Pocono Manor, PA. 2000. P. 53–57.
8. Anemuller J., Kollmeier B., Amplitude modulation decorrelation for convolutive blind source separation// P. Pajunen and J. Karhunen (eds.). Proceedings of the second international workshop on independent component analysis and blind signal separation. June 19–22. 2000. Helsinki, Finland. P. 215–220.

БАЗИСНОСТЬ В ПРОСТРАНСТВЕ $L_p(0,1)$ ОДНОЙ СИСТЕМЫ СОБСТВЕННЫХ ФУНКЦИЙ, ОТВЕЧАЮЩИХ ЗАДАЧЕ СО СПЕКТРАЛЬНЫМ ПАРАМЕТРОМ В ГРАНИЧНОМ УСЛОВИИ

Марченков Д. Б.

В работах [1,2] изучался вопрос о базисности в пространствах $L_2(0,1)$ и $L_p(0,1)$ систем собственных функций, отвечающих задачам, содержащим спектральный параметр в локальном граничном условии. В данной работе исследуется базисность системы собственных функций, отвечающей задаче со спектральным параметром в нелокальном граничном условии.

Рассматривается следующая спектральная задача :

$$\begin{aligned} &u''(x) + \lambda u(x) = 0, \\ &u(0) = 0, \quad u'(0) - d\lambda u(1) = 0, \quad d > 0. \end{aligned} \tag{1}$$

Базисные свойства собственых функций задачи (1) зависят от значения параметра d. Возможны два случая.

1. Пусть в задаче (1) $d \neq \frac{1}{x_i \sin x_i}$ $\forall i = 0, 1, 2 \ldots$, где x_i — корни уравнения $\sin x + x \cos x = 0$ на $(0, +\infty)$, пронумерованные в порядке возрастания.

В этом случае задача (1) имеет только собственные функции:

$$u_n(x) = \sqrt{2} \sin \sqrt{\lambda_n} x,$$

где λ_n — собственные значения, удовлетворяющие уравнению

$$d\sqrt{\lambda} \sin \sqrt{\lambda} = 1.$$

Последнее уравнение имеет бесконечное число корней, но в зависимости от значения d все они будут действительными или конечное число корней будет комплексными:

а) при $d > \frac{1}{x_0 \sin x_0}$ все собственные значения будут положительными числами. Присваиваем нулевой индекс любой собственной функции, а все остальные нумеруем в порядке возрастания собственных чисел;

б) при $d < \frac{1}{x_0 \sin x_0}$ уравнение $d\sqrt{\lambda}\sin\sqrt{\lambda} = 1$ имеет конечное число комплексно сопряженных комплексных корней,а остальные корни положительны. Нулевой индекс мы присваиваем любой собственной функции, а все остальные нумеруем в порядке возрастания $\mathrm{Re}\sqrt{\lambda_n}$.

Имеет место следующее утверждение.

Теорема 1. *Пусть* $d \neq \frac{1}{x_i \sin x_i}$ $\forall i = 0, 1, 2 \ldots$, *тогда система* $\{u_n(x)\}_{n=1}^{\infty}$, *т.е. система собственных функций задачи (1) без любой удаленной функции, образует базис в пространстве* $L_p(0,1)$, $p > 1$.

Доказательство этой теоремы опирается на ряд вспомогательных утверждений, которые сформулированы и обоснованы ниже.

Лемма 1. *Биортогонально сопряженная система* $\{\phi_k(x)\}_{k=1}^{\infty}$ *к системе* $\{u_k(x)\}_{k=1}^{\infty}$ *определяется по формуле:*

$$\phi_k(x) = \sqrt{2}\frac{(\overline{\sqrt{\lambda_0}}\sin\overline{\sqrt{\lambda_0}}(1-x) - \overline{\sqrt{\lambda_k}}\sin\overline{\sqrt{\lambda_k}}(1-x))}{\overline{\sqrt{\lambda_k}}\cos\overline{\sqrt{\lambda_k}} + \sin\overline{\sqrt{\lambda_k}}}.$$

Это утверждение является следствием следующих двух соотношений:

$$\int_0^1 \sin\sqrt{\lambda_n}\,x \cdot \sin\sqrt{\lambda_k}\,(1-x)\,dx = \frac{1}{d\sqrt{\lambda_n}\,\sqrt{\lambda_k}}, \quad n \neq k,$$

$$\int_0^1 \sin\sqrt{\lambda_n}\,x \cdot \sin\sqrt{\lambda_n}\,(1-x)\,dx = \frac{1}{2d\lambda_n} - \frac{\cos\sqrt{\lambda_n}}{2}.$$

Лемма 2. *Имеет место следующая асимптотическая оценка:*

$$\sqrt{\lambda_n} = \pi n + (-1)^n \frac{1}{\pi d n} + O(\frac{1}{n^3}).$$

Эта оценка легко выводится из уравнения для собственных чисел и поведения в окрестности нуля функции $\sin x$.

Покажем теперь, что система $\{u_n(x)\}$, $n = 1, 2, \dots$, удовлетворяет условию следующей теоремы Н.К. Бари [4, с. 373–382].

Всякая ω-линейно независимая последовательность $\left\{g_j\right\}_1^\infty$, квадратично близкая к базису Рисса $\left\{\psi_j\right\}_1^\infty$ сама является базисом Рисса.

В нашем случае последовательность $\left\{u_n(x)\right\}$ в силу леммы 2 является квадратично близкой к ортонормированному базису $\left\{\sqrt{2}\sin\pi nx\right\}_{n=1}^\infty$. Из этого, а также из леммы 1, следует ω-линейная независимость последовательности $\left\{u_n(x)\right\}$. Таким образом имеет место

Лемма 3. *Система $\left\{u_n(x)\right\}$, $n = 1, 2, \dots$, — базис Рисса в пространстве $L_2(0, 1)$.*

Лемма 4. *Пусть $f \in C[0, 1]$ и разложима в равномерно сходящийся на $[0, 1]$ ряд Фурье по системе $\left\{\sqrt{2}\sin\pi nx\right\}_{n=1}^\infty$, тогда эта функция разложима в ряд Фурье по системе $\left\{u_n(x)\right\}_{n=1}^\infty$, равномерно сходящийся на любом сегменте $[0, b]$, где $0 < b < 1$. Если $(f, \sin\sqrt{\lambda_0}\,(1 - x)) = 0$, то ряд Фурье функции f по системе $\left\{u_n(x)\right\}_{n=1}^\infty$ сходится равномерно на $[0, 1]$.*

Доказательство леммы 4 проводится аналогично доказательству теоремы 2 работы [3], с использованием лемм 1–3.

Приступим к доказательству основного утверждения.

Доказательство теоремы 1. В силу леммы 1 существует система $\left\{\phi_n\right\}_{n=1}^\infty$ — биортогонально сопряженная к системе $\left\{u_n\right\}_{n=1}^\infty$. А это означает, что система $\left\{u_n\right\}_{n=1}^\infty$ минимальна в $L_p(0, 1)$. Полнота системы $\left\{u_n\right\}_{n=1}^\infty$ следует из Леммы 4. Поэтому в силу критерия базисности [5, с. 19] для доказательства базисности $\left\{u_n\right\}_{n=1}^\infty$ в $L_p(0, 1)$ необходимо и достаточно дока-

зать следующее утверждение:

$$\exists M > 0: \ \forall f(x) \in L_p(0,1)$$
$$\Big\|\sum_{n=1}^{N}(f,\phi_n)u_n\Big\|_{L_p(0,1)} \leqslant M\|f\|_{L_p(0,1)} \quad \forall N = 1,2,\dots . \quad (3)$$

Введем обозначения:

$$d_n = \frac{\sqrt{2}\sqrt{\lambda_0}}{\sqrt{\lambda_n}\cos\sqrt{\lambda_n} + \sin\sqrt{\lambda_n}}, \quad c_n = \frac{\sqrt{\lambda_n}}{\sqrt{\lambda_n}\cos\sqrt{\lambda_n} + \sin\sqrt{\lambda_n}},$$
$$v_n(x) = \sqrt{2}\sin\pi n x, \qquad C_\lambda = \max_i \max_{x\in[0,1]} |\sin\sqrt{\lambda_i}x|.$$

Пусть $f \in L_p(0,1)$, $\frac{1}{p} + \frac{1}{q} = 1$. Тогда

$$\Big\|\sum_{n=1}^{N}(f,\phi_n)u_n\Big\|_{L_p(0,1)} \leqslant$$
$$\leqslant \Big\|\sum_{n=1}^{N} c_n(f, \sqrt{2}\sin\overline{\sqrt{\lambda_n}}(1-x))u_n(x)\Big\|_{L_p(0,1)} +$$
$$+ C_\lambda \cdot \Big\|f\Big\|_{L_p(0,1)} \cdot \Big\|\sum_{n=1}^{N} d_n u_n(x)\Big\|_{L_p(0,1)},$$
$$\Big\|\sum_{n=1}^{N} d_n u_n(x)\Big\|_{L_p(0,1)} \leqslant$$
$$\leqslant \Big\|\sum_{n=1}^{N} d_n v_n(x)\Big\|_{L_p(0,1)} + \Big\|\sum_{n=1}^{N} d_n(u_n(x) - v_n(x))\Big\|_{L_p(0,1)}.$$

Последнее слагаемое ограничено $\forall N$ в силу леммы 2. Для оценки первого рассмотрим два случая:

а) $p \geqslant 2$, тогда по теореме Рисса [6, с. 154]:

$$\Big\|\sum_{n=1}^{N} d_n v_n(x)\Big\|_{L_p(0,1)} \leqslant C_1 \cdot \Big(\sum_{n=1}^{\infty} |d_n|^q\Big)^{1/q};$$

б) $1 < p < 2$, тогда

$$\Big\|\sum_{n=1}^{N} d_n v_n(x)\Big\|_{L_p(0,1)} \leqslant C_2 \cdot \Big\|\sum_{n=1}^{N} d_n v_n(x)\Big\|_{L_2(0,1)} \leqslant C_2 \cdot \Big(\sum_{n=1}^{\infty} |d_n|^2\Big)^{1/2}.$$

Итак, $\Big\|\sum\limits_{n=1}^{N} d_n u_n(x)\Big\|_{L_p(0,1)} \leqslant C_3$, где C_1, C_2, C_3 не зависят от N,

$$\Big\|\sum_{n=1}^{N} c_n(f, \sqrt{2}\sin\sqrt{\overline{\lambda_n}}(1-x))u_n\Big\|_{L_p(0,1)} \leqslant$$

$$\leqslant \Big\|\sum_{n=1}^{N} \frac{c_n\sqrt{2}}{d\sqrt{\lambda_n}}(f, \cos\sqrt{\overline{\lambda_n}}\,x)u_n\Big\|_{L_p(0,1)} +$$

$$+ \Big\|\sum_{n=1}^{N} c_n \cos\sqrt{\overline{\lambda_n}}\,(f, \overline{u_n})u_n\Big\|_{L_p(0,1)},$$

$$\Big\|\sum_{n=1}^{N} \frac{c_n\sqrt{2}}{d\sqrt{\lambda_n}}(f, \cos\sqrt{\overline{\lambda_n}}\,x)u_n\Big\|_{L_p(0,1)} \leqslant$$

$$\leqslant \Big\|\sum_{n=1}^{N} \frac{c_n\sqrt{2}}{d\sqrt{\lambda_n}}(f, \cos\sqrt{\overline{\lambda_n}}\,x)v_n\Big\|_{L_p(0,1)} +$$

$$+ \Big\|\sum_{n=1}^{N} \frac{c_n\sqrt{2}}{d\sqrt{\lambda_n}}(f, \cos\sqrt{\overline{\lambda_n}}\,x)[u_n - v_n]\Big\|_{L_p(0,1)}.$$

Так как $\sum\limits_{n=1}^{\infty}\Big|\frac{\sqrt{2}c_n}{d\sqrt{\lambda_n}}\cdot(f, \cos\sqrt{\overline{\lambda_n}}\,x)\Big|^q < \infty$ при $q \in (1, 2]$, то используя теорему Рисса [6, с. 154], можно получить следующую оценку при $p \geqslant 2$:

$$\Big\|\sum_{n=1}^{N} \frac{\sqrt{2}c_n}{d\sqrt{\lambda_n}}(f, \cos\sqrt{\overline{\lambda_n}}\,x)v_n\Big\|_{L_p(0,1)} \leqslant C_4\Big\|f\Big\|_{L_p(0,1)},$$

где C_4 не зависит от N. При $1 < p < 2$ можно воспользоваться тем, что $L_2(0,1) \subset L_p(0,1)$. Имеем

$$\Big\|\sum_{n=1}^{N} c_n \cos\sqrt{\overline{\lambda_n}}\,(f, \overline{u_n})u_n\Big\|_{L_p(0,1)} \leqslant$$

$$\leqslant \Big\|\sum_{n=1}^{N}(f, u_n)u_n\Big\|_{L_p(0,1)} + C_5\Big\|f\Big\|_{L_p(0,1)}\cdot\sum_{n=1}^{N}\Big|c_n\cos\sqrt{\overline{\lambda_n}} - 1\Big|.$$

Второе слагаемое ограничено в силу леммы 2, оценим первое:

$$\Big\| \sum_{n=1}^{N} (f, u_n)u_n \Big\|_{L_p(0,1)} \leqslant$$

$$\leqslant \Big\| \sum_{n=1}^{N} (f, v_n)v_n \Big\|_{L_p(0,1)} + \Big\| \sum_{n=1}^{N} (f, v_n)[u_n - v_n] \Big\|_{L_p(0,1)} +$$

$$+ \Big\| \sum_{n=1}^{N} (f, u_n - v_n)v_n \Big\|_{L_p(0,1)} + \Big\| \sum_{n=1}^{N} (f, u_n - v_n)[u_n - v_n] \Big\|_{L_p(0,1)}.$$

Оценим каждое слагаемое.

1.Неравенство

$$\Big\| \sum_{n=1}^{N} (f, v_n)v_n \Big\|_{L_p(0,1)} \leqslant M_1 \cdot \Big\| f \Big\|_{L_p(0,1)}$$

верно, так как $\Big\{ v_n \Big\}_{n=1}^{\infty}$ — базис в $L_p(0,1)$ [5, с. 19].

2.

$$\Big\| \sum_{n=1}^{N} (f, u_n - v_n)[u_n - v_n] \Big\|_{L_p(0,1)} \leqslant$$

$$\leqslant \sum_{n=1}^{N} \Big\| f \Big\|_{L_p(0,1)} \times \Big\| u_n - v_n \Big\|_{L_p(0,1)} \times \Big\| u_n - v_n \Big\|_{L_q(0,1)} \leqslant$$

$$\leqslant \; C_6 \Big\| f \Big\|_{L_p(0,1)} \cdot \sum_{n=1}^{N} \frac{1}{n^2}.$$

3. Оценим $\Big\| \sum_{n=1}^{N} (f, u_n - v_n)v_n \Big\|_{L_p(0,1)}$:

а) $p \geqslant 2$, тогда по теореме Рисса [6, с. 154]

$$\Big\| \sum_{n=1}^{N} (f, u_n - v_n)v_n \Big\|_{L_p(0,1)} \leqslant C_7 \cdot \Big(\sum_{n=1}^{N} \Big| (f, u_n - v_n) \Big|^q \Big)^{1/q} \leqslant$$

$$\leqslant C_7 \cdot \Big\| f \Big\|_{L_p(0,1)} \cdot \Big(\sum_{n=1}^{\infty} \| u_n - v_n \|^q_{L_q(0,1)} \Big)^{1/q};$$

б) $1 < p < 2$. В этом случае можно воспользоваться тем, что $L_2(0,1) \subset L_p(0,1)$.

4. Рассмотрим оценку

$$\Big\| \sum_{n=1}^{N} (f, v_n)[u_n - v_n] \Big\|_{L_p(0,1)} \leqslant \sum_{n=1}^{N} \Big|(f, v_n)\Big| \cdot \Big\|u_n - v_n\Big\|_{L_p(0,1)} :$$

а) $1 < p < 2$. Тогда по теореме Рисса [6, с. 154]

$$\Big(\sum_{n=1}^{N} \Big|(f, v_n)\Big|^q\Big)^{1/q} \leqslant C_8 \cdot \Big\|f\Big\|_{L_p(0,1)},$$

следовательно,

$$\sum_{n=1}^{N} \Big|(f, v_n)\Big| \cdot \Big\|u_n - v_n\Big\|_{L_p(0,1)} \leqslant$$

$$\leqslant \Big(\sum_{n=1}^{\infty} \Big|(f, v_n)\Big|^q\Big)^{1/q} \times \Big(\sum_{n=1}^{\infty} \Big\|u_n - v_n\Big\|^p_{L_p(0,1)}\Big)^{1/p} \leqslant$$

$$\leqslant C_8 \cdot \Big(\sum_{n=1}^{\infty} \Big\|u_n - v_n\Big\|^p_{L_p(0,1)}\Big)^{1/p} \times \Big\|f\Big\|_{L_p(0,1)};$$

б) $p \geqslant 2$. Тогда

$$\exists M_p > 0 : \ \forall g \in [0,1] \quad \Big\|g\Big\|_{L_2(0,1)} \leqslant M_p \Big\|g\Big\|_{L_p(0,1)},$$

следовательно,

$$\Big(\sum_{n=1}^{\infty} \Big|(f, v_n)\Big|^2\Big)^{1/2} \leqslant C_9 \cdot \Big\|f\Big\|_{L_p(0,1)},$$

тогда

$$\sum_{n=1}^{N} \Big|(f, v_n)\Big| \cdot \Big\|u_n - v_n\Big\|_{L_p(0,1)} \leqslant$$

$$\leqslant C_9 \Big\|f\Big\|_{L_p(0,1)} \cdot \Big(\sum_{n=1}^{\infty} \Big\|u_n - v_n\Big\|^2_{L_p(0,1)}\Big)^{1/2},$$

где ряд в скобках сходится в силу леммы 2.

Итак (3) доказано. Следовательно, система $\Big\{u_n(x)\Big\}_{n=1}^{\infty}$ образует базис в $L_p(0,1)$. Теорема доказана.

2. Теперь рассмотрим случай, когда $d = \frac{1}{x_0 \sin x_0}$, где x_0 — один из положительных корней уравнения $\sin x + x \cos x = 0$. В этом случае задача (1) имеет помимо собственных функций

$$u_n(x) = \sqrt{2} \sin \sqrt{\lambda_n} x$$

(λ_n — собственные значения из уравнения $d\sqrt{\lambda}\sin\sqrt{\lambda}=1$, в котором $\mathrm{Re}\sqrt{\lambda}>0$) одну присоединенную функцию, отвечающую собственному значению $\lambda=x_0^2$. Занумеруем собственные функции в порядке возрастания $\mathrm{Re}\sqrt{\lambda_n}$, $n=1,2,3,\ldots$. В отличие от первого случая здесь вся система собственных функций образует базис в $L_p(0,1)$.

Теорема 2. *Система $\{u_n(x)\}_{n=1}^{\infty}$ собственных функций задачи* (1) *образует базис в $L_p(0,1)$, $p>1$.*

Доказательство этого утверждения проводится аналогично доказательству теоремы 1 с той лишь разницей, что биортогонально сопряженная система $\{\phi_k(x)\}_{k=1}^{\infty}$ к системе $\{u_k(x)\}_{k=1}^{\infty}$ определяется по-другому (см. следующую лемму).

Лемма 5. *Биортогонально сопряженная система $\{\phi_i(x)\}_{i=1}^{\infty}$ к системе $\{u_k(x)\}_{k=1}^{\infty}$ определяется так:*

$$\phi_i(x)=\begin{cases}\sqrt{2}\cdot\overline{C_i}\{\sqrt{\lambda_s}\sin\sqrt{\lambda_s}(1-x)-\overline{\sqrt{\lambda_i}}\sin\overline{\sqrt{\lambda_i}}(1-x)\},\\ \qquad\qquad\qquad\qquad\qquad\qquad\text{если } i\neq s,\\ \sqrt{2}\cdot C_s\{\sin\sqrt{\lambda_s}(1-x)+\sqrt{\lambda_s}(1-x)\cos\sqrt{\lambda_s}(1-x)\},\\ \qquad\qquad\qquad\qquad\qquad\qquad\text{если } i=s,\end{cases}$$

где s — номер собственного значения, которое совпадает с x_0^2, т.е. $\lambda_s=x_0^2$, и

$$C_s=\frac{\sqrt{\lambda_s}+1}{1+\frac{\lambda_s}{2}},\quad C_i=\frac{1}{\sqrt{\lambda_i}\cos\sqrt{\lambda_i}+\sin\sqrt{\lambda_i}}.$$

Автор благодарит Е.И. Моисеева за постановку задачи и обсуждение работы.

ЛИТЕРАТУРА

1. Капустин Н.Ю.,Моисеев Е.И.//Диф. уравнения. 1997. **33**. N 1. С. 115–119.
2. Капустин Н.Ю.,Моисеев Е.И.//Диф. уравнения. 2000. **36**. N 10. С. 1357–1360.
3. Капустин Н.Ю.,Моисеев Е.И.//Диф. уравнения. 2000. **36**. N 8. С. 1069–1074.
4. Гохберг И.Ц., Крейн М.Г. Введение в теорию линейных несамосопряженных операторов. М., 1965.
5. Кашин Б.С.,Саакян А.А. Ортогональные ряды. М.: Наука, 1984.
6. Зигмунд А. Тригонометрические ряды. Т. 2. М., 1965.

ПОСТРОЕНИЕ МАТЕМАТИКО-СТАТИСТИЧЕСКОЙ МОДЕЛИ ТЕЧЕНИЯ ЭТНОПОЛИТИЧЕСКОГО КОНФЛИКТА

Петрова М. А.

Одной из наиболее актуальных проблем современного общества являются этнополитические конфликты. Использование математического аппарата для анализа и прогнозирования течения конфликта позволяет более адекватно оценивать и решать возникающие проблемы. В данной работе описывается процесс построения математико-статистической модели этнополитического конфликта. Целью работы является прогнозирование течения конфликта, поиск точек смены характера процесса, а также исследование факторов, влияющих на социальную напряженность.

Феномен этничности на Северном Кавказе играет особо важную роль. Этничность, будучи универсальной формой человеческого бытия, выступала и будет выступать как относительно самостоятельный конфликтогенный фактор. Полиэтничность действительно повышает уровень конфликтогенности общества [1]. В качестве одной из главных причин конфликтности на Северном Кавказе многими исследователями полагается рост национального самосознания. Усиление роли этнической идентичности может быть основой межнациональной конфликтности. В качестве главной причины конфликтного напряжения в большинстве случаев является установление приоритета собственных национальных ценностей перед всеми другими [2].

Для исследования мотивации протестных действий жителей Северного Кавказа был использован регрессионный анализ. Была построена линейная регрессионная модель, сделаны выводы по мотивационным установкам, осуществлен пробный прогноз и оценена его точность.

В векторном виде модель регрессии записывается следующим образом: $y = Xu + \varepsilon$, $X \in R^{nxk}$, где X — матрица данных с элементами $X_{ij} = \psi_j(x_i)$, $i = 1, \ldots, n$; $j = 1, \ldots, k$.

В нашем случае регрессорами служат какие-либо факторы, наблюдаемые величины — каким-либо образом измеренная *социальная напряженность*. Необходимый объем выборки достигается рассмотрением нескольких регионов.

Требуемые линейные оценки θ могут быть получены по методу наименьших квадратов как решение задачи минимизации с целевой функцией $Q = Q(\theta) = \|y - X\|^2$, здесь норма $\|\cdot\|$ — евклидова. Приравнивая к нулю производную, получаем необходимое условие экстремума в виде системы нормальных уравнений $X^T X\theta = X^T y$. Но у такой системы очень часто бывает большое число обусловленности,что сильно портит картину и уменьшает доверие к полученным данным.

Избежать ухудшения обусловленности при формировании нормальных уравнений можно, не формируя их вовсе [3], используя существование сингулярного разложения (SVD — Singular Value Decomposition) матрицы. Известно (теорема о SVD): пусть $X \in R^{nxk}$, тогда существуют ортогональные матрицы U, V и матрица $\Sigma \in R^{nxk}$: $\Sigma = \mathrm{diag}\{\sigma_1, \ldots, \sigma_l\}$, где $l = \min\{n, k\}$, $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_l \geqslant 0$ (числа $\sigma_1, \ldots, \sigma_l$ — сингулярные значения), такие, что выполнено $X = U\Sigma V^T$. Использование SVD для решения задачи НК позволяет иметь гораздо меньшую погрешность при решении, так как при этом не формируется система вида $X^T X\theta = X^T y$, и не используются матрицы типа $X^T X$, обычно имеющие большое число обусловленности. Именно этот метод использовался при расчетах.

В качестве регрессоров использовались следующие факторы:

ψ_1 — уровень безработицы (в %),

ψ_2 — изменение этого уровня (в%),

ψ_3 — средняя продолжительность жизни (в годах),

ψ_4 — среднедушевой доход (в десятках рублей),

ψ_5 — количествово беженцев в регионе за текущий год (в сотнях человек),

ψ_6 — изменение удельного веса населения с доходами ниже прожиточного минимума (в%),

ψ_7 — уровень политической активности населения (измеренной как активность электората на выборах (в%)).

Социальная напряженность в регионе измеряется количест-

вом совершенных преступлений [4]. Выборка бралась по 9 регионам Северного Кавказа (кроме Чеченской республики) за 1999 год. Источники данных — сборник Госкомстата [5] и официальный сайт Центральной избирательной комиссии РФ [6]. Число обусловленности полученной матрицы $\mathrm{cond}(X) = \sigma_1/\sigma_l$ = 80,043 Соответственно был получен следующий вектор регрессионных коэффициентов:

$$\theta = \begin{bmatrix} -45,15 \\ 70,93 \\ 61,659 \\ 3,829 \\ -10,371 \\ 26,602 \\ -34,379 \end{bmatrix}, \text{ или после нормализации } \begin{bmatrix} -0,398 \\ 0,625 \\ 0,543 \\ 0,034 \\ -0,091 \\ 0,234 \\ -0,303 \end{bmatrix}.$$

Можно сделать вывод, что наибольший вклад в усилении социальной напряженности вносят ψ_2, ψ_3 и ψ_6, т.е. увеличение уровня безработицы, увеличение удельного веса населения с доходами ниже прожиточного минимума и средняя продолжительность жизни. При этом подтверждается гипотеза многих авторов (Попова А.В., S.Olzak, D.Myers), что на рост социальной напряженности оказывает влияние не собственно плохая экономическая ситуация, а ее изменение к худшему. По имеющимся данным был проведен пробный ретропрогноз на 1998 год, и появилась возможность оценить точность прогноза. На рис. 1 изображен уровень социальной напряженности в разных регионах Северного Кавказа (по оси x — порядковый номер региона) по данным Госкомстата (сплошная линия), а также прогнозируемый с помощью полученной формулы регрессии уровень напряженности (пунктирная линия). Средняя относительная погрешность прогноза составляет 0,101, максимальная по Ингушетии — 0,547.

Полученные результаты позволяют использовать регрессионный анализ как одну из методик прогнозирования изменения социальной напряженности на Северном Кавказе.

При моделировании процесса течения этнополитического конфликта важную роль играет обработка реальных данных о протестных действиях. Предполагается, что статистическое распределение периодов времени между протестными действиями (вспышками социальной активности) является двухпараметрическим распределением Вейбулла—Гнеденко [7, с. 17–39]:

$$F(x) = 1 - \exp\{-\lambda x^m\}.$$

При этом можно по статистической выборке определить его основные параметры m и λ, которые в статистике называются

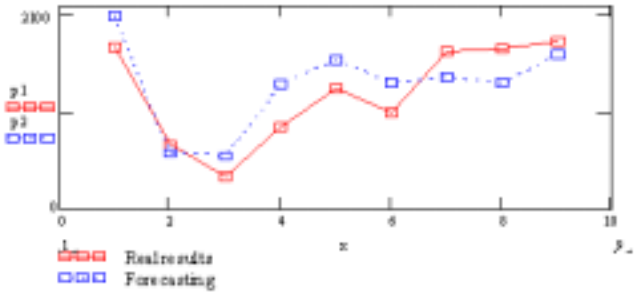

Рис. 1. Результаты ретропрогноза социальной напряженности в регионах Северного Кавказа за 1998г.

соответственно параметрами формы и масштаба.

Модель Вейбулла—Гнеденко для исследования и прогнозирования течения этнополитического конфликта использовали многие ученые (Petersen I.D., Кретов В.С., Закожурников С.Ю.). Модель предполагает, что интервалы между протестными действиями имеют распределение Вейбулла—Гнеденко, и для прогнозирования течения конфликта достаточно определить параметр формы этого распределения:

$$F(x) = 1 - \exp\{-\lambda x^m\}.$$

Если $m = 1$, то интенсивность рассматриваемого процесса не зависит от времени, т.е. система находится в устойчивом состоянии, активность не изменяется.

Если $m < 1$, то процесс носит затухающий характер, интенсивность исследуемого процесса уменьшается со временем. Система находится в состоянии контролируемых изменений: она изменяется монотонно, срабатывают механизмы “сильной” отрицательной обратной связи.

Если $m > 1$, то интенсивность процесса возрастает со временем. Это система с положительной обратной связью [8].

Первым использовал этот подход Петерсен И.Д., при этом он предполагал, что в международных и межэтнических отношениях существуют некоторые динамические законы, которые меняются во времени, но которые можно и нужно изучать и исследовать [7, с. 11–15].

Покажем, что распределение Вейбулла—Гнеденко — это распределение минимальной статистики. Пусть у нас есть выборка вида $(z_1, z_2, z_3, \ldots, z_n)$ (назовем ее *родительской выборкой*). Предположим, что z — это независимые одинаково

распределенные случайные величины, и пусть они распределены с плотностью вероятности $\varphi(z)$. Определим *минимальную статистику* как

$$Z_{\min(n)} = \min(z_1, z_2, \ldots, z_n).$$

Выборку, состоящую из $Z_{\min(n)}$, называют *дочерней выборкой.*

Обозначим ее функцию распределения как $F_{\min(n)}(x)$:

$$F_{\min(n)}(x) = P\left(Z_{\min(n)} < x\right) = 1 - P\left(Z_{\min(n)} > x\right) = \\ = 1 - P(z_i > x,\ 1 \leqslant i \leqslant n) = 1 - (1 - \Phi(x))^n,$$

где

$$\Phi(x) = \int_0^x \varphi(z)$$

(это равенство следует из определения функции распределения и независимости z).

Если в правой окрестности нуля функция $\varphi(z)$ удовлетворяет следующему условию: $\lim\limits_{z \to +0} z^{-\Delta}\varphi(z) = c$, $c > 0$, то можно показать, что при $n \to \infty$ предельное распределение минимальной статистики будет распределением Вейбулла—Гнеденко [9, с. 115–118]:

$$\lim_{n\to\infty} F_{\min(n)}(x) = \lim_{n\to\infty} 1 - \left(1 - \frac{\mu x^\gamma}{n}\right)^n = 1 - \exp(-\mu x^\gamma),$$

т.е. это двухпараметрическое распределение Вейбулла—Гнеденко. Описанный механизм отражает *модель цепочки*: в цепочке n звеньев, и событие происходит, когда рвется самое слабое звено.

Была построена модель этнополитического конфликта, обосновывающая использование распределения Вейбулла—Гнеденко для описания протестных действий в регионах Северного Кавказа. Для каждого человека определяются два параметра — *оппозиционный настрой* и *готовность к действию. Оппозиционный настрой* характеризует способность человека в принципе совершать протестные действия ради какой-либо цели. Человек, не имеющий *оппозиционного настроя* (в данный момент), не будет совершать каких-либо протестных действий, во всяком случае до тех пор, пока такой настрой не появится. Будем полагать *оппозиционный настрой* бинарной величиной (он либо есть, либо — нет). “Для того чтобы

протест состоялся, необходим *определенный уровень социального недовольства*, придание силы и *массовых* действий в качестве приемлемого средства социальных перемен. Росту депривации[1)] и активизации протестных действий способствуют радикальные идеологии, лозунги и символические акции, недоверие к политическому режиму, упадок веры в традиционные способы выражения требований". Этот определенный уровень социального недовольства и есть оппозиционный настрой. Он возникает при общем ухудшении ситуации в регионе (это было установлено исследованием мотивационных установок с помощью регрессионного анализа).

Для людей, у которых такой настрой есть, вводим понятие группы единомышленников. Люди обычно совершают протестные действия в группе. Так, исследователь Ерижева А.Х. из Кабардино-Балкарии пишет: "Основными субъектами политического конфликта в республике являются группы, *выражающие мнение своего этноса* или *присваивающие себе это право* [10]". Пусть некоторая группа людей действует как единое целое. Параметр *готовность к действию* может принимать различные положительные целые значения и означает примерно следующее: *готовность к действию можно определять как количество дней, которое осталось до протестных действий при неизменных внешних условиях, готовность 1 — готовность действовать прямо сейчас, готовность 2 — готовность действовать завтра и т.п.* Когда у одного (или нескольких) из членов группы этот параметр равен единице, он заражает своим настроем своих соратников, группа готова действовать и совершает некоторое протестное действие, происходит некоторое *событие.* После этого *готовность к действию* у всех членов группы резко падает, у каждого — на свой уровень. События мы можем наблюдать и регистрировать.

По построению модели ясно, что весь процесс (совокупность протестных действий, совершенных данной группой) описывается распределением Вейбулла—Гнеденко, так как готовность к действию — случайная величина: когда минимум равен единице, происходит событие, родительская выборка содержит время, через которое готовность будет равной единице, минимальная статистика выбирает минимальное такое время. Временной интервал между событиями (протестными

[1)] Депривация — состояние недовольства, вызываемое расхождением между реальным и/или оцениваемым и ожидаемым состоянием, к которому стремится субъект. Депривация — одна из наиболее распространенных концепций, объясняющих причины и механизмы протестного поведения.

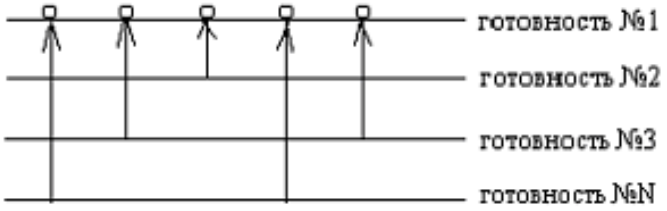

Рис. 2. Момент события. Для всех готовность становится равной 1

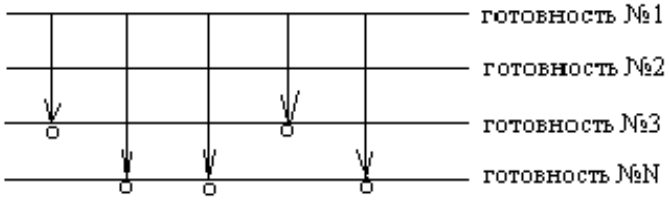

Рис. 3. После события. Готовность людей к действию резко падает, у каждого на свой уровень

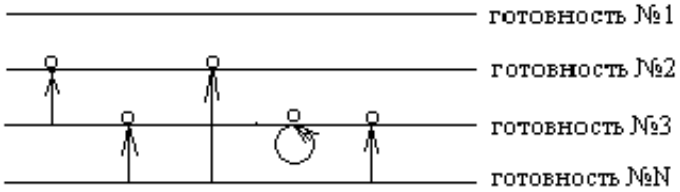

Рис. 4. Промежуточный момент. Готовность людей к действию растет или остается неизменной

действиями, совершенными *одной* группой) будет описываться распределением Вейбулла—Гнеденко. Вопрос возникает, когда мы начинаем рассматривать *все* группы, действующие в пределах одной местности. Можно показать [10], что если группы независимы друг от друга, то результирующее распределение тоже будет распределением Вейбулла—Гнеденко. Если же действия различных групп являются взаимозависимыми (что имеет место в реальной жизни), то условия, при кото-

рых результирующее распределение является распределением Вейбулла—Гнеденко, до конца не выявлены, и приходится это проверять эмпирически.

Итак, рассмотрим данные, которыми мы будем оперировать. В качестве t_n берется промежуток времени между событиями с номером n и $n-1$. По имеющимся данным можно построить $y(t)$ — эмпирическую функцию распределения выборки $\{t_n\}$. Рассматриваемая случайная величина — ξ (промежуток времени между событиями, характеризующимися социальной напряженностью), $(t_1, \ldots, t_n)$ — выборка из реализаций случайной величины с неизвестной функцией распределения.

Для начала, действуя в рамках предположения о виде распределения случайной величины ξ, можно оценить параметры этого распределения. Итак, пусть функция распределения ξ — функция Вейбулла—Гнеденко

$$F_W(t) = 1 - \exp(-\lambda t^m).$$

Плотность распределения имеет вид

$$f(t; \lambda, m) = m\lambda t^{m-1} \exp(-\lambda t^m).$$

При этом функция правдоподобия для выборки $(t_1, \ldots, t_n)$ имеет вид

$$L(\lambda, m) = \lambda^n m^n \prod_{i=1}^{n} t_i^{m-1} \exp\left(-\lambda \sum_{i=1}^{n} t_i^m\right).$$

Оценка максимального правдоподобия величин λ, m — это оценка, обращающая в максимум функцию правдоподобия. Как обычно, вместо функции $L(\lambda, m)$ берется ее логарифм

$$\ln(L(\lambda, m)) = n\ln(\lambda) + n\ln(m) + (m-1)\sum_{i=1}^{n} \ln(t_i) - \lambda \sum_{i=1}^{n} t_i^m.$$

В точке максимума производные функции $\ln(L(\lambda, m))$ обращаются в 0. Следовательно, выполняются следующие соотношения [7, с. 30]:

$$\frac{\partial \ln(L(\lambda, m))}{\partial \lambda} = \frac{n}{\lambda} \sum_{i=1}^{n} t_i^m = 0,$$

$$\frac{\partial \ln(L(\lambda, m))}{\partial m} = \frac{n}{m} + \sum_{i=1}^{n} \ln(t_i) - \lambda \sum_{i=1}^{n} \ln(t_i) t_i^m = 0.$$

Отсюда находим, что оценки максимального правдоподобия $\hat{\lambda}$, $\hat{m}$ должны удовлетворять следующим уравнениям:

$$\hat{\lambda} = \frac{n}{\sum\limits_{i=1}^{n} t_i^{\hat{m}}}, \tag{1}$$

$$f(\hat{m}) = \frac{n}{\hat{m}} + \sum_{i=1}^{n} \ln(t_i) - \frac{n}{\sum\limits_{i=1}^{n} t_i^{\hat{m}}} \sum_{i=1}^{n} \ln(t_i) t_i^{\hat{m}}. \tag{2}$$

Последнее уравнение не может быть решено аналитически, но его можно решить с помощью численных методов. В работе [12, с. 192] оно было решено методом простой итерации $x_{n+1} = x_n + \tau f(x_n)$. В качестве начального приближения бралось приближение по методу наименьших квадратов, полученное из графика lnln (см. рис. 5). Соответственно были получены оценки максимального правдоподобия $\hat{\lambda}$, $\hat{m}$.

Построенная модель подразумевает, что данные о социальной напряженности в регионе, рассмотренные как реализации некоторой случайной величины ξ, подчиняются распределению Вейбулла—Гнеденко. Эта гипотеза нуждается в проверке.

Итак, рассматривается гипотеза $_0 : F_W(t) = 1 - \exp(-\lambda t^m)$ аппроксимирует $y(t)$, т.е. функцией распределения случайной величины ξ является $F_W(t)$. Для проверки этой гипотезы используется критерий согласия Колмогорова [13, с. 107]. Статистикой критерия является величина

$$D_n = D_n(x) = \sup_{-\infty<x<+\infty} |F_W(t) - y(t)|,$$

представляющая собой максимальное отклонение эмпирической функции распределения от гипотетической. Подразумевается, что $F_W(t) = 0$ при $t < 0$. Если гипотеза истинна, то $F_W(t)$ при каждом t является оптимальной оценкой для $y(t)$ [12, с. 44]. Распределение статистики D_n при гипотезе $_0$ не зависит от вида функции распределения (в данном случае $F_W(t)$). Если (при $n = 20$) наблюдавшееся значение $d = D_n(t)$ статистики удовлетворяет неравенству $\sqrt{n}d \geqslant k_\alpha$, где k_α определяется по таблице в соответствии с заданным уровнем значимости α, то гипотезу $_0$ отвергают, в противном случае делают вывод, что статистические данные не противоречат гипотезе [13, с. 108]. Так, $k_\alpha = 1,3581$ при $\alpha = 0,05$. При вычислениях в качестве λ, m брались оценки максимального правдоподобия, полученные из условий (1), (2) .

Целью данного исследования является поиск точек смены характера распределения. Так как процесс получения точек выборки $\{t_n\}$ был довольно растянут по времени (типичные выборки были получены в течение 400–500 дней, вообще говоря, в разных условиях), резонно предположить, что (в силу внешних причин или внутренних свойств моделируемого объекта) параметры распределения к концу выборки могли измениться. Если в некоторых точках эти параметры изменяются, то можно предположить, что процесс течения этнополитического конфликта меняет свой характер. Главным критерием, характеризующим результат подгонки, полагается качество аппроксимации эмпирической функции распределения искомой теоретической функцией распределения (критерий Колмогорова).

Сначала определяются области, подозрительные на содержание точек смены параметров. Проводится исследование графиков интенсивности, при этом подозрительными являются точки резкой смены интенсивности процесса. Также для определения подозрительных точек привлекается внешняя информация, известная из других источников. После определения множества подозрительных точек на каждом подозрительном интервале рассматривается следующий график [10, с. 117]. Из уравнения

$$F_W(t) = 1 - \exp(-\lambda t^m)$$

двойным логарифмированием можно получить выражение

$$\ln(\ln\{1/[1 - F_W(t)]\}) = m \ln t + \ln \lambda.$$

Поэтому для исследования можно рассмотреть графики

$$\ln(\ln\{1/[1 - y(t)]\}),$$

где по оси абсцисс отложен $\ln t$. В случае соответствия эмпирического распределения на интервале распределению Вейбулла—Гнеденко этот график должен быть прямой, причем коэффициент наклона этой прямой соответствует параметру формы m, а абсцисса в точке 0 (intercept) соответствует параметру масштаба λ. Найденные с помощью метода наименьших квадратов приближенные значения параметров m и λ используются в качестве начального приближения при решении уравнения (2) методом простой итерации при вычислении оценок максимального правдоподобия.

Следующий этап: с помощью критерия Колмогорова проводится исследование соответствия теоретической функции распределения эмпирической на каждом интервале. Также по критерию Смирнова проверяется гипотеза однородности для всех

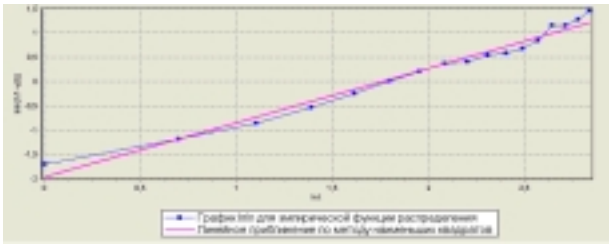

Рис. 5. График lnln эмпирической функции распределения для реальных данных (Дагестан, 2000–2001г.) и линейное приближение по методу наименьших квадратов

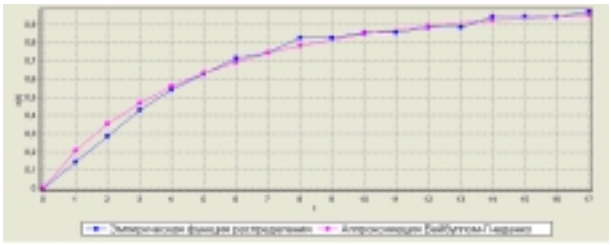

Рис. 6. График эмпирической функции распределения (Дагестан, 2000–2001г.) и теоретической функции Вейбулла—Гнеденко с параметрами, оцененными с помощью функции максимального правдоподобия

соседних частей найденного разбиения выборки. В случае, если эта гипотеза подтверждается, найденная точка не соответствует смене параметров, и количество элементов разбиения сокращается.

Таким образом, с помощью вышеописанной процедуры можно отслеживать точки смены параметров. Это позволяет подбирать теоретическую функцию распределения, наилучшим образом соответствующую реальным данным. Проведенное исследование позволяет, во-первых, прогнозировать течение этнополитического конфликта при неизменных внешних условиях, во-вторых, анализировать точки смены характера процесса и по дополнительным данным определять, что именно повлекло за собой эскалацию, или соответственно затухание процесса.

Описанная совокупность методик (регрессионный анализ,

анализ параметров распределения Вейбулла—Гнеденко) позволяет исследовать и прогнозировать течение этнополитического конфликта. Построенная модель используется для реальных расчетов.

Автор выражает благодарность к.ф.-м.н. Шведовскому В.А. за научное руководство и внимание к данной проблеме.

ЛИТЕРАТУРА

1. Савва Е.В. Этнополитический конфликт: проблемы построения теоретической модели. http://www.stavsu.ru/CONFR/KONFL_CONF/SEC1/sav.html.
2. Казанцев В.Г. (полномочный Представитель Президента России на Северном Кавказе) Межнациональные конфликты на Северном Кавказе в контексте геополитических реалий. Автореферат канд. дисс.
3. Уфимцев М.В. Методы многомерного статистического анализа. М. Изд-во МГУ, 1997.
4. Шведовский В.А. Динамическая модель этнополитического конфликта: построение, возможности и результаты применения// Математическое моделирование социальных процессов. Вып.2 М., 2000.
5. Регионы России. Сборник Госкомстата. М., 1999.
6. Центральная избирательная комиссия РФ. Официальный сайт. http://www.fci.ru.
7. Petersen I.D. The Dynamic Laws of International Political Systems 1823-1973. Institute of political studies, University of Copenhagen, Copenhagen Political Studies, 1980.
8. Бабынин И.В., Власов И.Е., Кретов В.С., Фролов И.В., Информационная модель политического конфликта // Сборник трудов 5 национальной конференции “Искусственный интеллект - 96”. АИИ. Казань, 1996. С. 379–383.
9. Indow T. Analyses of Events Counted on Time-Dimension: a Soft Model of Extreme Statistics// Behaviometrika. Vol. 20. N 2. P. 109–124.
10. Ерижева А.Х. Региональный конфликт в полиэтничном обществе. http://www.stavsu.ru/CONFR/KONFL_CONF/SEC4/er.htm.
11. Indow T. Weibull Form in Memory, Reaction Time, and Social Behavior: Asymptotic Distribution of Minima from Heterogeneous Population// Institute for mathematical behavioral sciences. Technical report series. MBS 95-04, 1995.
12. Самарский А.А., Гулин А.В. Численные методы. М.: Наука, 1989.
13. Ивченко Г.И., Медведев Ю.И. Математическая статистика. М.: Высшая школа, 1984.

УДК 517.956

СУЩЕСТВОВАНИЕ И ЕДИНСТВЕННОСТЬ ОБОБЩЕННОГО РЕШЕНИЯ СМЕШАННОЙ ЗАДАЧИ ДЛЯ ВОЛНОВОГО УРАВНЕНИЯ С НЕЛИНЕЙНЫМ НЕЛОКАЛЬНЫМ ГРАНИЧНЫМ УСЛОВИЕМ

Чабакаури Г. Д.

В настоящей работе рассматривается смешанная задача с нелинейным нелокальным граничным условием для одномерного волнового уравнения, доказывается существование и единственность обобщенного решения этой задачи из класса, допускающего существование ограниченной энергии.

Обозначим символом Q_T прямоугольник $Q_T = [0 \leqslant x \leqslant l] \times [0 \leqslant t \leqslant T]$.

Определение 1. Будем говорить, что функция двух переменных $u(x,t)$ *принадлежит классу* $\widehat{W}_2^1(Q_T)$, если функция $u(x,t)$ непрерывна в замкнутом прямоугольнике $\bar{Q}_T$ и имеет в этом прямоугольнике обе обобщенные частные производные $u_x(x,t)$ и $u_t(x,t)$, каждая из которых принадлежит классу $L_2(Q_T)$ и, кроме того, принадлежит классу $L_2[0 \leqslant x \leqslant l]$ при любом фиксированном t из сегмента $[0,T]$ и классу $L_2[0 \leqslant t \leqslant T]$ при любом фиксированном x из сегмента $[0,l]$.

Определение 2. Будем говорить, что функция одной переменной $\underline{\nu}(t)$ принадлежит классу $\underline{W}_2^1[0,T]$, если эта функция определена для всех $t \leqslant T$, принадлежит классу $W_2^1[0,T]$ и удовлетворяет условиям $\underline{\nu}(0) = 0$, $\underline{\nu}(t) \equiv 0$ для всех $t < 0$.

Настоящая работа выполнена при финансовой поддержке Российского Фонда фундаментальных исследований (проект 00–15–96104 поддержки ведущих научных школ).

Рассмотрим в прямоугольнике Q_T следующую задачу для волнового уравнения:

$$u_{tt}(x,t) - u_{xx}(x,t) = F(x,t) \quad \text{в } Q_T, \tag{1}$$

$$u(x,0) = \varphi(x), \quad u_t(x,0) = \psi(x) \quad \text{при } 0 \leqslant x \leqslant l, \tag{2}$$

$$u(0,t) = \mu(t), \;\; u_x(l,t) = \int\limits_0^l f(u(\xi,t), u_t(\xi,t), u_\xi(\xi,t), \xi, t)\, d\xi \quad \text{при } 0 \leqslant t \leqslant T, \tag{3}$$

где $F(x,t) \in L_2(Q_T)$, $\varphi(x) \in W_2^1[0,l]$, $\psi(x) \in L_2[0,l]$ и $\mu(t) \in W_2^1[0,T]$.

Определение 3. *Решением из класса* $\widehat{W}_2^1(Q_T)$ *задачи* (1)–(3) назовем функцию $u(x,t)$ из этого класса, которая удовлетворяет тождеству

$$\begin{aligned}
&\int\limits_0^l \int\limits_0^T u(x,t)\,[\Phi_{tt}(x,t) - \Phi_{xx}(x,t)]\,dxdt + \\
&\qquad + \int\limits_0^l [\varphi(x)\Phi_t(x,0) - \psi(x)\Phi(x,0)]dx - \\
&\qquad - \int\limits_0^T \mu(t)\Phi_x(0,t)dt - \int\limits_0^T \nu(t)\Phi(l,t)dt - \\
&- \int\limits_0^T \int\limits_0^l f(u(\xi,t), u_t(\xi,t), u_\xi(\xi,t), \xi, t)\Phi(l,t)d\xi dt = \\
&\qquad = \int\limits_0^l \int\limits_0^T F(x,t)\Phi(x,t)dxdt
\end{aligned} \tag{4}$$

для произвольной функции $\Phi(x,t) \in C^2(\overline{Q}_T)$, подчиненной условиям $\Phi(0,t) \equiv 0$, $\Phi_x(l,t) \equiv 0$ при $0 \leqslant t \leqslant T$ и условиям $\Phi(x,T) \equiv 0$, $\Phi_t(x,T) \equiv 0$ при $0 \leqslant x \leqslant l$, и которая, кроме того, удовлетворяет первому условию (3) и первому начальному условию (2) в классическом смысле, а второму начальному условию (2) и второму условию (3) — в смысле равенства элементов $L_2[0,l]$.

В работе В.А. Ильина и Е.И. Моисеева [1] была рассмотрена задача для волнового уравнения, у которой второе из условий (3) имеет вид $u(l,t) = \sum\limits_{k=1}^{n} \alpha_k(t) u(\xi_k, t) + \nu(t)$, где $0 \leqslant \xi_1 < \xi_2 < \ldots < \xi_n < l$ и $\alpha_k(t)$ — произвольные функции, определенные на $0 \leqslant t \leqslant T$.

В настоящей работе рассматривается вопрос о существовании и единственности решения задачи (1)–(3).

Теорема 1. *Пусть выполнены следующие условия:*
1) условие согласования $\mu(0) = \varphi(0)$*;*
2) условие Липшица для функции $f(z_1, z_2, z_3, x, t)$*, т.е.*

$$|f(z_1, z_2, z_3, x, t) - f(y_1, y_2, y_3, x, t)| \leqslant \\ \leqslant C\left(|z_1 - y_1| + |z_2 - y_2| + |z_3 - y_3|\right)$$

для любых $z_i, y_i \in R$, $i = 1, 2, 3$.

Тогда, для любого $T > 0$ *существует единственное решение задачи* (1)–(3) *из класса* $\widehat{W}_2^1(Q_T)$.

Доказательство. Без ограничения общности можно считать, что $\varphi(x) \equiv 0$, $\psi(x) = 0$ и $F(x,t) = 0$. Действительно, продолжим функции $\varphi(x)$ и $\psi(x)$ сначала на сегмент $[-l, 0]$ нечетно относительно точки $x = 0$, а затем на сегмент $[l, 2l]$ так, чтобы $\varphi(x) \in W_2^1[-l, 2l]$ и $\psi(x) \in L_2[-l, 2l]$, а функцию $F(x,t)$ продолжим тождественным нулем с прямоугольника Q_T на все пространство R^2. С таким образом продолженными функциями $\varphi(x)$, $\psi(x)$ и $F(x,t)$ рассмотрим функцию

$$V(x,t) = \Bigg[(\varphi(x-t) + \varphi(x+t))/2 + \\ + \int\limits_{x-t}^{x+t} \psi(\xi)\,d\xi + \int\limits_0^t \int\limits_{x-(t-\tau)}^{x+(t-\tau)} F(\xi,\tau)\,d\xi\,d\tau\Bigg],$$

где $u(x,t)$ есть решение исходной задачи. Предположим, что решение задачи (1)–(3) из класса $\widehat{W}_2^1(Q_T)$ существует. Тогда очевидно, что функция $v(x,t) = u(x,t) - V(x,t)$ является обобщенным решением из класса $\widehat{W}_2^1(Q_T)$ задачи вида (1)–(3) с $\varphi(x) \equiv 0$, $\psi(x) = 0$ и $F(x,t) = 0$. Таким образом, всюду далее

будем рассматривать задачу:

$$u_{tt}(x,t) - u_{xx}(x,t) = 0 \quad \text{в } Q_T, \tag{5}$$

$$u(x,0) = 0, \quad u_t(x,0) = 0 \quad \text{при } 0 \leqslant x \leqslant l, \tag{6}$$

$$u(0,t) = \mu(t), \ u_x(l,t) = \int_0^l f(u(\xi,t), u_t(\xi,t), u_\xi(\xi,t), \xi, t)\, d\xi \quad \text{при } 0 \leqslant t \leqslant T. \tag{7}$$

Для начала рассмотрим частный случай, когда $T = l$.

В силу определения класса $\widehat{W}_2^1(Q_T)$ существует след функции $u(x,t)$ на прямой $x = l$, т.е. $u(l,t) = \nu(t) \in W_2^1[0,T]$. Очевидно, что функция $u(x,t)$ является обобщенным решением из класса $\widehat{W}_2^1(Q_T)$ смешанной задачи (5), (6) и

$$v(0,t) = \mu(t), \qquad v(l,t) = \nu(t).$$

Определение обобщенного решения из класса $\widehat{W}_2^1(Q_T)$ этой задачи и ее детальное исследование представлены в работе В.А. Ильина [2]. В частности, в этой работе показано, что в случае, когда $T = l$б справедлива формула:

$$u(x,t) = \underline{\mu}(t-x) + \underline{\nu}(t+x-l) \quad \text{для всех } 0 \leqslant x \leqslant l, \tag{8}$$

где $\underline{\nu}(t) \in \underline{W}_2^1[0,T]$.

Функция $u(x,t)$, определяемая формулой (8), удовлетворяет (5), (6) и первому из условий (7). Покажем, что функцию $\nu(t)$ можно выбрать таким образом, чтобы выполнялось и второе из условий (7). Подставляя (8) во второе из условий (7), получим следующее интегро-дифференциальное уравнение для определения функции $\underline{\nu}(t)$:

$$\underline{\nu}'(t) = \int_0^l f\Big(\underline{\mu}(t-x) + \underline{\nu}(t+x-l), \underline{\mu}'(t-x) + \underline{\nu}'(t+x-l), - \underline{\mu}'(t-x) + \underline{\nu}'(t+x-l), x, t\Big) dx, \tag{9}$$

$$\underline{\nu}(0) = 0. \tag{10}$$

Заметим, что последнее уравнение можно свести к интегральному уравнению относительно $\underline{\nu}'(t)$, если в подынтегральном выражении заменить $\underline{\nu}(t+x-l)$ на $\int_0^{t+x-l} \underline{\nu}'(\xi) d\xi$.

Рассмотрим теперь нелинейный оператор $A : L_2[0,T] \to L_2[0,T]$, определяемый следующей формулой:

$$A(g) = \int_0^l f\Bigg(\underline{\mu}(t-x) + \int_0^{t+x-l} \underline{g}(\xi)d\xi, \underline{\mu}'(t-x) + \underline{g}(t+x-l), \\ - \underline{\mu}'(t-x) + \underline{g}(t+x-l), x, t\Bigg)dx, \quad (11)$$

где $\underline{g}(t) = g(t)$ при $t \in [0,T]$ и $\underline{g}(t) = 0$ при $t < 0$. Очевидно, что решение задачи (9)–(10) эквивалентно решению задачи $\underline{\nu}'(t) = A(\underline{\nu}')$ при условии $\underline{\nu}(0) = 0$. Докажем существование единственного решения этой задачи.

Докажем, что некоторая степень оператора A является сжимающим отображением.

В силу условия Липшица

$$|A(g_1) - A(g_2)| \leqslant C \int_0^l \Bigg[\int_0^{t+x-l} |\underline{g}_1(\xi) - \underline{g}_2(\xi)|d\xi + \\ + |\underline{g}_1(t+x-l) - \underline{g}_2(t+x-l)|\Bigg]dx.$$

В правой части полученного неравенства произведем замену переменной $y = t + x - l$. Учитывая, что подынтегральное выражение обращается в ноль при $y < 0$ и что $t \leqslant l$ получим, что

$$|A(g_1) - A(g_2)| \leqslant \\ \leqslant C \int_0^t \Bigg[\int_0^y |g_1(\xi) - g_2(\xi)|d\xi + |g_1(y) - g_2(y)|\Bigg]dy = \\ = C \int_0^t (t-y+1)|g_1(y) - g_2(y)|dy \leqslant C_2 \int_0^t |g_1(y) - g_2(y)|dy. \quad (12)$$

Применяя к (12) неравенство Коши—Буняковского легко получить оценки

$$|A(g_1) - A(g_2)|^2 \leqslant C_3 \int_0^t |g_1(y) - g_2(y)|^2 dy, \quad (13)$$

$$|A(g_1) - A(g_2)|^2 \leqslant C_3 t ||g_1 - g_2||^2_{L_2(0,T)}. \tag{14}$$

Из (13) и (14) тривиально следует неравенство

$$|A^n(g_1) - A^n(g_2)|^2 \leqslant \frac{(C_3 t)^n}{n!} ||g_1 - g_2||^2_{L_2(0,T)}.$$

Интегрируя полученное неравенство от 0 до T, получаем, что

$$\|A^n(g_1) - A^n(g_2)\|_{L_2(0,T)} \leqslant T \frac{(C_3 T)^n}{(n+1)!} ||g_1 - g_2||^2_{L_2(0,T)}.$$

Следовательно, при достаточно больших n оператор A^n является сжимающим отображением. Значит, существует единственная функция $\underline{\nu}'(t)$ из $L_2(0,T)$, являющаяся решением уравнения (8). Функцию $\underline{\nu}(t)$ однозначно определяем из условия $\underline{\nu}(0) = 0$. Таким образом, для случая $T \leqslant l$ теорема доказана.

Рассмотрим случай $T = 2l$. Если решение задачи (5)–(7) существует, то в прямоугольнике $[0,l] \times [l,2l]$ оно совпадает с решением следующей задачи:

$$v_{tt}(x,t) - v_{xx}(x,t) = 0 \quad \text{в } [0,l] \times [l,2l], \tag{15}$$

$$v(x,l) = \widehat{\varphi}(x), \quad v_t(x,l) = \widehat{\psi}(x) \quad \text{при } 0 \leqslant x \leqslant l, \tag{16}$$

$$v(0,t) = \mu(t), \quad v_x(l,t) = \int\limits_0^l f(v(\xi,t), v_t(\xi,t), v_\xi(\xi,t), \xi, t) d\xi \quad \text{при } l \leqslant t \leqslant 2l, \tag{17}$$

где (16) $\widehat{\varphi}(x) = u(x,l) \in W_2^1[0,l]$ и $\widehat{\psi}(x) = u_t(x,l) \in L_2[0,l]$, а $u(x,t)$ есть решение (5)–(7) в случае $T = l$. Производя замену переменной $\tau = t - l$, получим задачу, аналогичную задаче (1)–(3) для случая $T = l$. Таким образом, вопрос о существовании и единственности решения задачи (15)–(17) сведен к рассмотренному выше случаю, когда $T = l$ и, следовательно, теорема доказана для $T = 2l$. Аналогично проводится доказательство для любого $T = 2lk$, где k — натуральное число.

Теорема полностью доказана. □

Рассмотрим теперь задачу (1), (2) и соотношения

$$u(0,t) = 0, \quad u_x(l,t) = (g(\cdot,t), u(\cdot,t))_{W_2^1[0,l]} + k(t), \tag{18}$$

где $k(t) \in L_2[0,T]$ и $g(x,t) \in L_2(Q_T)$.

Теорема 2. *Пусть выполнено условие согласования* $\mu(0) = \varphi(0)$. *Тогда для любого* $T > 0$ *существует единственное решение задачи* (1), (2) *и* (18) *из класса* $\widehat{W}_2^1(Q_T)$.

Доказательство. Как и при доказательстве теоремы 1 будем предполагать, что $\varphi(x) \equiv 0$, $\psi(x) = 0$ и $F(x,t) = 0$. В случае, когда $g(x,t) \in C^1[Q_T]$ доказательство тривиально следует из теоремы 1. Рассмотрим случай, когда $g(x,t) \in L^2[Q_T]$. Пусть $T = l$.

Также как при доказательстве теоремы 1, вопрос о существовании и единственности обобщенного решения из класса $\widehat{W}_2^1(Q_T)$ задачи (1), (2) и (18) сводится к решению интегро-дифференциального уравнения (10) с начальным условием (10) и функцией

$$f(z_1, z_2, z_3, x, t) = z_1 g(x,t) + z_2 g_t(x,t) + z_3 g_x(x,t) + k(t)/l. \quad (19)$$

Производя в уравнении (9) с функцией $f(z_1, z_2, z_3, x, t)$, определяемой соотношением (19), замену переменной $y = t + x - l$ получаем уравнение

$$\widehat{\underline{\nu}}'(t) = \int\limits_0^t \left[\widehat{\underline{\nu}}(\xi) g(\xi - t + l, t) + \widehat{\underline{\nu}}'(\xi) g_x(\xi - t + l, t)\right] d\xi - \widehat{k}(t), \quad (20)$$

где $\widehat{k}(t)$ — некоторая известная функция из $L_2[0,T]$, которая выражается через $k(t)$ и $\underline{\mu}(t)$.

Принимая во внимание тот факт, что $\widehat{\underline{\nu}}(0) = 0$, с помощью интегрирования по частям легко убедиться, что

$$\int\limits_0^t \widehat{\underline{\nu}}(\xi) g(\xi - t + l, t) d\xi = -\int\limits_0^t \widehat{\underline{\nu}}'(\xi) \left(\int\limits_l^{\xi - t + l} g(x,t) dx \right) d\xi.$$

Из последнего соотношения и из формулы (20) получаем, что

$$\widehat{\underline{\nu}}'(t) - \int\limits_0^t \widehat{\underline{\nu}}'(\xi) G(\xi, t) d\xi = \widehat{k}(t), \quad (21)$$

где $G(\xi, t) = g_x(\xi - t + l, t) - \int\limits_l^{\xi - t + l} g(x,t) dx.$

Из определения функции $g(x,t)$ следует, что $G(\xi,t) \in L_2([0,T]\times[0,T])$, если только $T \leqslant l$. Итак, для $\underline{\widehat{\nu}}'(t) \in L_2(0,T)$ мы получили уравнение Вольтерра второго рода с правой частью из $L_2[0,T]$ и ядром из $L_2([0,T]\times[0,T])$. Из разрешимости уравнения Вольтерра второго рода следует, что для случая $T = l$ теорема доказана. Доказательство для произвольного T проводится также как в теореме 1. □

Рассмотрим теперь задачу (1), (2) и

$$u(0,t) = \mu(t), \quad u_x(l,t) = \sum_{k=1}^{n} \alpha_k(t) u(\xi_k,t) + k(t), \qquad (22)$$

где $0 \leqslant \xi_1 < \xi_2 < \ldots < \xi_n \leqslant l$ и $\alpha_k(t)$ — произвольные функции из $L_2[0 \leqslant t \leqslant T]$.

Следствие. *Для любого $T > 0$ необходимым и достаточным условием существования и единственности обобщенного решения смешанной задачи* (1), (2), (22) *из класса $\widehat{W}_2^1(Q_T)$ является выполнение условия согласования $\mu(0) = \varphi(0)$.*

Доказательство. Из определения класса $\widehat{W}_2^1(Q_T)$ мы получаем, что для любого фиксированного $t_0 \in [0,T]$ функция $u(x,t_0)$ принадлежит классу $W_2^1[0,l]$ как функция переменной x. Но тогда, по теореме Рисса существует единственный набор функций $g_i(x) \in W_2^1[0,l]$, $i = 1,\ldots,n$, таких, что $u(\xi_i,t) = (g_i(\cdot), u(\cdot,t))_{W_2^1[0,l]}$. Тогда очевидно, что

$$u_x(l,t) = (g(\cdot,t), u(\cdot,t))_{W_2^1[0,l]} + k(t),$$

где $g(x,t) = \sum\limits_{k=1}^{n} \alpha_k(t) g_k(x)$. Справедливость доказываемого утверждения немедленно следует из теоремы 2. □

Автор благодарит В.А. Ильина, прочитавшего рукопись и высказавшего ряд полезных замечаний, учтенных при написании этой работы.

ЛИТЕРАТУРА

1. *Ильин В.А., Моисеев Е.И.*// Диф. уравнения. 2000. **36**. № 5. С. 728–733
2. *Ильин В.А.*// Диф. уравнения. 2000. **36**. № 12. С. 1670–1686.

УДК 681.3.06

ИСПОЛЬЗОВАНИЕ ПОСЛЕДОВАТЕЛЬНОГО ВЫПОЛНЕНИЯ ДЛЯ ОТЛАДКИ ПАРАЛЛЕЛЬНЫХ MPI ПРОГРАММ

Яковенко П. Н.

§ 1. Введение

Стадия отладки является одной из самых сложных и наименее проработанных в жизненном цикле параллельной программы. Это обусловлено как отсутствием стандартов в этой области, так и неразвитостью методологии параллельной отладки. Как результат, на сегодняшний день накоплено большое количество отладчиков, которые так и не получили широкого распространения. Бóльшая их часть создавалась в рамках учебно-научных исследовательских проектов и доступна только для одной платформы. Наиболее значимым продвижением стало опубликование предварительной версии стандарта на параллельный отладчик, разработанного форумом HPD [1] и частично реализованного в отладчиках TotalView [2], P2D2 [3] и др. Однако окончательная версия стандарта и его полная реализация все еще находятся в разработке, несмотря на значительное отставание от сроков, изначально декларированных форумом.

Отладка параллельной программы разбивается на две основные стадии: отладка ошибок и настройка производительности. Задача первой стадии состоит в том, чтобы параллельная программа на любом допустимом наборе входных данных выдавала корректный результат. На этой стадии необходимо локализовать и устранить все ошибки, связанные с логикой

программы, коммуникационными операциями, синхронизацией, блокировками, тупиками (deadlock) и т.п. Отладка логики работы программы в отдельных случаях может быть проведена с помощью последовательного отладчика, например, в случае SPMD программы, которая выполняется на одном узле. Однако некоторые ветви в коде такой программы могут получить управление только при условии, что программа запущена на нескольких узлах вычислительной системы. Поэтому для проведения полноценной отладки ошибок требуется использование отладчика, позволяющего одновременно анализировать выполнение всех процессов параллельной программы.

Настройка производительности заключается в получении высокоэффективной параллельной программы, максимально адаптированной для той платформы, на которой она будет функционировать. Основная работа на этой стадии посвящена повышению эффективности алгоритма, увеличению числа операций, выполняемых параллельно, и уменьшению времени простоев каждого процесса. Традиционно наибольшее внимание уделяется распараллеливанию циклов и преобразованию синхронных операций взаимодействия в асинхронные, поскольку трансформация именно этих частей кода параллельной программы позволяет получить наибольшее ускорение времени выполнения при анализе относительно небольших блоков программы. При настройке производительности необходимо учитывать особенности реализации алгоритма на данной архитектуре, иерархию памяти, пропускную способность коммуникационных каналов и задержки передачи сообщений между узлами, особенности генерируемого кода и размещения данных в памяти.

Важным на стадии анализа производительности является вопрос масштабируемости параллельной программы. Масштабируемость программы необходимо рассматривать применительно к конкретной масштабируемой вычислительной системе.

Производительность каждого узла вычислительной системы (а также производительность всей вычислительной системы в целом) измеряется при помощи специальных эталонных программ, называемых бенчмарками. Это связано с тем, что производительность системы может значительно различаться на разных классах задач. Поэтому для каждого класса задач используются свои бенчмарки.

В работе [4] предложены следующие определения масштабируемой вычислительной системы и масштабируемой программы.

Определение 1. Параллельная вычислительная система называется масштабируемой, если производительность системы пропорциональна сумме производительностей всех ее узлов.

Определение 2. Масштабируемость параллельной программы — это линейная зависимость скорости ее выполнения от производительности масштабируемой вычислительной системы.

Для масштабируемых вычислительных систем и параллельных программ коэффициент пропорциональности должен быть близок к единице. На практике в силу накладных расходов с увеличением числа процессоров рост производительности замедляется. Для каждой масштабируемой параллельной программы, выполняющейся на данной вычислительной системе, существует максимальное число узлов (оно определяется архитектурой системы) такое, что при большем числе узлов ускорения выполнения программы не происходит.

Далее в статье будут рассматриваться вопросы параллельной отладки, связанные со стадией отладки ошибок.

В настоящее время при реализации параллельного отладчика используется один из двух подходов:

- построение распределенного отладчика, позволяющего одновременно управлять выполнением всех процессов параллельной программы, не внося каких-либо изменений в код программы и библиотек поддержки выполнения. Для проведения отладки достаточно откомпилировать исходные тексты так, чтобы компилятор сохранил отладочную информацию. При отладке используются стандартные библиотеки;
- адаптация программы и/или библиотек поддержки выполнения таким образом, чтобы была возможность использовать для отладки традиционный последовательный отладчик. При таком подходе стадии отладки предшествует инструментирование исходных текстов программы, подключение модифицированных библиотек и другие методы, позволяющие отлаживать программу при помощи произвольного последовательного отладчика.

Распределенный отладчик реализуется в виде управляющей надстройки над набором последовательных отладчиков, по одному для каждого выполняющегося процесса. При этом последовательный отладчик расширяют функциями, позволяющими удаленно управлять им и получать информацию о состоянии отлаживаемого процесса и значениях его внутренних структур данных. Подавляющее большинство реализаций

основывается на последовательном отладчике *gdb*, исходные тексты которого доступны бесплатно для многих программно-аппаратных платформ.

Разработка оригинального распределенного отладчика сопряжена с большими денежными и временны́ми затратами, поэтому подобных реализаций крайне мало и бóльшая их часть выполнена непосредственно производителем выпускаемой аппаратной платформы. Наиболее известны TotalView, Prism [5] и IPD [6]. Преимуществами таких отладчиков являются высокая скорость работы и поддержка всех особенностей архитектуры вычислительной системы. К недостатком следует отнести высокую стоимость и отсутствие версий для других программно-аппаратных платформ. Исключением является TotalView, реализованный для Compaq Alpha, HP, SUN Sparc, Intel x86 и др. Однако из операционных систем поддерживаются только коммерческие и бесплатные версии UNIX.

Существенным недостатком распределенных отладчиков является тот факт, что пользователь (в силу физиологических ограничений) в каждый момент времени может заниматься мониторингом только одного процесса, в то время как остальные продолжают выполняться скрытым от него образом. Пользователь должен постоянно контролировать состояние процессов программы, чтобы ни один из них не продвинулся далее той точки, в которой он хочет исследовать значения структур данных и другие характеристики процессов. Управление отлаживаемой программой становится особенно сложным, когда программа выполняется на реальных объемах данных и число процессов исчисляется десятками, а иногда сотнями. Для облегчения управления процессами в отладчиках предусмотрены различные механизмы, такие как группировка процессов, синхронизационные контрольные точки и т.п. Однако отображение нумерации процессов MPI на группы процессов в среде отладчика и многие другие задачи возлагаются на пользователя.

Преодолеть указанный недостаток призваны отладочные средства, реализованные с использованием второго подхода. Основная его идея состоит в том, что параллельная программа и/или библиотеки поддержки выполнения адаптируются таким образом, чтобы была возможность применять для отладки параллельной программы традиционные последовательные отладчики.

Метод, основанный на изоляции процесса в MPI программе, предложен в работе [7] и основывается на симулировании окружения процесса. Принцип этого метода состоит в следующем. Все сообщения, принимаемые некоторым процессом,

записываются в трассировочный файл. После этого данный процесс, и только он, запускается под отладчиком, а система поддержки выполнения полностью имитирует для него внешнее окружение, создавая иллюзию того, что он выполняется параллельно со всеми остальными процессами программы. Например, программа состоит из четырех параллельных процессов и требуется отладить третий. На первом этапе программа выполняется на некотором фиксированном наборе входных данных и содержимое всех сообщений, получаемых третьим процессом, записывается в трассировочный файл. На втором этапе число выполняющихся процессов уменьшается до одного. Система поддержки выполнения гарантирует, что функция *MPI_Comm_rank* будет возвращать значение "2", а *MPI_Comm_size* — "4" и т.д. При обращении к функциям приема сообщения соответствующее сообщение будет извлекаться из трассировочного файла, отправляемые сообщения просто отбрасываются. В силу того, что после изоляции процесса параллельная программа трансформируется в последовательную, ее можно отлаживать при помощи стандартного последовательного отладчика.

К достоинствам описанного подхода следует отнести возможность отлаживать некоторый фиксированный процесс параллельной программы при помощи произвольного последовательного отладчика, а также устранение задержек в точках обращения к коммуникационным и синхронизирующим операциям MPI. Действительно, принимаемые сообщения извлекаются из трассировочного файла, который может быть расположен на локальном носителе, и задержка приема сообщений определяется только скоростью работы дисковой подсистемы компьютера. Передаваемые сообщения отбрасываются. Операции синхронизации возвращают управление немедленно, поскольку других процессов в программе нет и реально синхронизироваться изолированному процессу ни с кем не надо.

Авторы метода изоляции процесса признают, что предлагаемый ими подход к отладке параллельных программ не лишен недостатков. Во-первых, подготовительные работы, необходимые для изоляции процесса, сравнительно велики. Изменение входных данных или правка исходных текстов требуют полного повторения всех служебных прогонов программы. Во-вторых, динамика выполнения всех остальных процессов остается скрытой от пользователя. Он не может, находясь под отладчиком, изучать состояние других процессов программы.

Отладочные средства, адаптирующие параллельную программу с целью использования последовательных отладчиков, представляют собой расширение функциональности распреде-

ленных параллельных отладчиков и никоим образом не претендуют на их замену.

В этой работе предлагается механизм управления временем выполнения процессов в MPI программе, существенно облегчающий параллельную отладку. Метод **последовательного выполнения** объединяет в себе аспекты обоих подходов к параллельной отладке: построение распределенного отладчика и адаптация библиотеки MPI с целью использования последовательных отладчиков для отладки параллельной программы. С одной стороны предлагаемый механизм полностью сохраняет функциональность всех процессов параллельной программы во время отладки, т.е. процессы выполняются на тех же узлах, никакие изменения в исходные тексты не вносятся, в любой момент времени пользователь может просмотреть состояние каждого процесса, анализировать взаимодействия между ними, блокировки и т.д. С другой — в определенные точки кода каждого процесса вставляются задержки с целью реализации между процессами параллельной программы режима разделения времени. Величина задержки определяется непосредственно во время выполнения программы и подбирается так, чтобы в любой момент времени никакие два процесса параллельной программы не выполнялись одновременно.

При такой организации времени выполнения процессов параллельную программу можно рассматривать как последовательную, однако практически применять последовательный отладчик для отладки какого-либо процесса имеет смысл только тогда, когда длина интервала, в течение которого данный процесс выполняется монопольно, будет достаточно большой. В MPI программе наиболее оптимально в качестве такого интервала рассматривать блок кода, расположенный между двумя обращениями к коммуникационным функциям библиотеки MPI (например, между *MPI_Recv* и *MPI_Send*), а сами функции обмена сообщениями рассматривать как сервисные вызовы, обеспечивающие взаимодействие процесса с внешним миром (по аналогии со служебными вызовами операционной системы). Действительно, обмен информацией с другими процессами, как правило (в особенности в SPMD программах), происходит до начала обработки данных (получение исходных данных) и после ее завершения (возврат результата), а блок операторов, расположенный между двумя функциями обмена сообщениями, представляет собой последовательный код, реализующий некоторую законченную часть вычислений. Более того, одним из необходимых условий ускорения работы параллельной программы является минимизация числа взаимодействий между процессами. Поэтому для отладки блока кода,

расположенного между двумя обращениями к коммуникационной части библиотеки MPI, целесообразно использовать последовательный отладчик.

Использование стандартного профилировочного интерфейса MPI [8] позволяет реализовывать механизм последовательного выполнения, не внося изменений в код библиотеки MPI. При этом значительно упрощается перенос реализации предлагаемого метода на любую платформу, для которой существует реализация MPI.

§ 2. Последовательное выполнение

"Последовательное выполнение" является одним из способов управления временем выполнения процессов в параллельной программе. Система поддержки последовательного выполнения представляет собой набор управляющих модулей, распределенных по всем узлам параллельной вычислительной системы. Основная их задача — обеспечить такое поведение процессов, чтобы в любой момент времени в программе был активен только один процесс, а остальные находились в состоянии ожидания, т.е. реализовать режим разделения времени в распределенной параллельной программе.

Для объяснения принципов работы механизма последовательного выполнения обратимся к кооперативной (иногда называемой коллективной) многозадачности, представляющей собой один из способов разделения процессорного времени между выполняющимися процессами. Принцип кооперативной многозадачности состоит в том, что операционная система не занимается решением проблемы распределения процессорного времени. Распределяют его сами программы. Причем активная программа самостоятельно решает, отдавать ли процессор другой программе. Момент передачи управления зависит от хода выполнения задачи. Одним из основных недостатков кооперативной многозадачности является низкая надежность, так как зависание одного процесса приводит к зависанию других, выполняемых вместе с ним.

Механизм последовательного выполнения имеет много общего с кооперативной многозадачностью. На выделенном компьютере функционирует централизованный менеджер процессов. Каждый раз, когда процесс обращается к функциям библиотеки MPI, он отправляет сообщение менеджеру, полностью характеризующее тип выполняемой коммуникационной операции, и ожидает разрешения на продолжение выполнения. Детальное описание MPI операции необходимо для распознавания блокировок, которые могут возникать вследствие ошибок

во взаимодействии между процессами. При принятии сообщения от одного из процессов об окончании активности менеджер процессов проверяет наличие блокировок в программе и предоставляет разрешение на выполнение следующему процессу. При этом, если параллельная программа недетерминированная, он учитывает трассировочную информацию, полученную на стадии тестирования, для детерминированного воспроизведения программы. Таким образом, в каждый момент времени, а точнее на любом интервале между двумя обращениями к функциям MPI, активен только один процесс, а все остальные находятся в состоянии ожидания. Переключение между процессами происходит в точках MPI вызовов и после завершения выполнения активного процесса.

Не ограничивая общности, будем рассматривать программы только с коммуникационными операциями вида "точка-точка". Многие реализации MPI, в особенности доступные для различных программно-аппаратных платформ, реализуют групповые операции через набор коммуникационных операций вида "точка-точка". Например, операцию широковещательной рассылки (*MPI_Bcast*) можно рассматривать как вызов пользовательской функции, которая последовательно передает сообщения всем процессам параллельной программы.

Остановимся более подробно на логике работы менеджера процессов.

В каждый момент времени в активном состоянии находится только один процесс. В начальный момент времени активным является процесс с номером "0" (в соответствии с MPI нумерацией процессов), все остальные процессы выстраиваются в очередь в порядке их номеров. Очередь устроена по принципу FIFO (первым пришел — первым обслуживаешься).

Процессы в очереди могут находиться в одном из двух состояний:

- готовности;
- ожидания.

Состояние готовности характеризуется наличием у процесса всей необходимой информации для продолжения выполнения. Например, после инициализации программы (*MPI_Init*) все процессы помещаются в очередь в состоянии готовности.

Процесс находится в состоянии ожидания, если для выполнения коммуникационной операции ему требуется, чтобы второй процесс достиг определенной точки в программе. Например, принимающий процесс находится в состоянии ожидания, если передающий процесс еще не готов к отправке сообщения. Передающий процесс помещается в состояние ожидания,

если коммуникационная операция является синхронной и принимающий процесс не готов к приему. При возникновении в системе ожидаемого события (передающий процесс отправил сообщение) выполняется коммуникационная операция — сообщение копируется в область памяти принимающего процесса — и принимающий процесс переходит в состояние готовности.

Использование состояний готовности и ожидания необходимо для эффективного планирования времени выполнения процессов, а также распознавания тупиков и других блокировок в параллельной программе [9].

Алгоритм работы системы выглядит следующим образом:

ПОМЕТИТЬ все процессы как "готовые"
ПОМЕСТИТЬ все процессы в очередь в порядке номеров
ЦИКЛ ПОКА есть "готовые" процессы в очереди ИЛИ пользователь не закончил сеанс отладки
    АКТИВИРОВАТЬ первый в очереди процесс в состоянии готовности
    ЖДАТЬ ПОКА процесс не закончит работу ИЛИ начнет коммуникационную операцию
    ЕСЛИ процесс начал коммуникационную операцию
        ПРИОСТАНОВИТЬ процесс
        ЕСЛИ можно выполнить операцию
            ВЫПОЛНИТЬ пересылку данных
            ПОМЕТИТЬ взаимодействовавшие процессы как "готовые"
        ИНАЧЕ
            ПОМЕТИТЬ процесс как "ожидание"
        КОНЕЦ ЕСЛИ
        ПОМЕСТИТЬ процесс в конец очереди
    КОНЕЦ ЕСЛИ
КОНЕЦ ЦИКЛ
ЕСЛИ в очереди остались "ожидающие" процессы И пользователь не прерывал сеанс отладки
    ВЫДАТЬ предупреждающее сообщение
КОНЕЦ ЕСЛИ

При активировании процесса головной элемент очереди изымается из нее.

После завершения работы цикла в очереди могут остаться процессы в состоянии ожидания, что свидетельствует о наличии ошибки в программе. Такая ситуация возникнет, например, если процесс ожидает получения сообщения, которое не было ему отправлено или было ошибочно отправлено другому процессу. В этом случае менеджер процессов должен выдать предупреждающее сообщение пользователю.

Для обоснования корректности алгоритма заметим, что последовательное выполнение программы на произвольной вычислительной системе представляет собой выполнение той же программы под управлением операционной системы, реализующей кооперативную многозадачность. Действительно, в такой вычислительной системе функции извещения операционной системы об окончании активности процесса расположены на уровне библиотеки MPI. Реализовывать такую функциональность в теле программы нецелесообразно с точки зрения переносимости. Существенное отличие состоит в том, что менеджер процессов в системе поддержки последовательного выполнения не передает управление процессу, который находится в состоянии ожидания. Это позволяет снизить нагрузку на коммуникационные каналы, ведущие к централизованному менеджеру процессов, за счет устранения излишнего обмена сообщениями.

Следует заметить, что механизму последовательного выполнения присущи недостатки кооперативной многозадачности. Некорректная программа в некоторых случаях может привести к блокировке выполнения всех ее процессов. Если один из процессов зациклился или ждет появления некоторого внешнего события, то он не выйдет в точку обращения к библиотеке MPI и не сообщит системе поддержки выполнения об окончании своей активности. Такая ситуация требует вмешательства пользователя для разрешения конфликта.

В отличие от описанных выше подходов к параллельной отладке механизм последовательного выполнения обладает следующими преимуществами:

1. отладка выполняется при помощи стандартного последовательного отладчика;
2. исходные тексты программы не изменяются, поскольку задержки вносятся на уровне библиотеки MPI;
3. программа отлаживается на реальных объемах данных и произвольном числе узлов. В случае SPMD программы это позволяет анализировать те ветви, которые не получат управления при выполнении программы на одном узле вычислительной системы;
4. сеанс отладки не требует предварительных прогонов программы для сбора служебной отладочной информации. При отладке недетерминированной программы трассировочная информация, необходимая для детерминированного воспроизведения, передается со стадии тестирования вместе с тестовым набором;
5. динамика поведения всех процессов доступна пользовате-

лю для анализа во время сеанса отладки, — он может просматривать значения переменных и другие характеристики любого из отлаживаемых процессов;
6. пользователь может во время сеанса отладки анализировать трассы и буферы сообщений, блокировки и другие ошибки, связанные с некорректным взаимодействием процессов.

Механизм последовательного выполнения не решает всех задач, возникающих при отладке параллельной программы. Отдельного исследования требует настройка производительности, анализ задержек коммуникационных операций и синхронизации, масштабируемость программы. Доводка параллельной программы может потребовать использования средств анализа производительности, функционирующих на базе стандартной системы поддержки выполнения MPI программ. Кроме того, последовательное выполнение неминуемо приведет к замедлению работы программы, что может быть критичным для систем реального времени.

§ 3. Особенности отладки недетерминированных параллельных программ

Одной из основных проблем параллельной отладки является недетерминизм, вольно или невольно вносимый программистами в код. Недетерминизм играет важную роль в области параллельных и распределенных вычислений. С одной стороны программа, с детерминированным поведением обладает исключительными для многих областей применения свойствами, и исследователи считают, что надежное программное обеспечение должно быть детерминированным [10]. С другой, — использование недетерминированных коммуникационных операций позволяет повысить степень параллелизма в программе за счет снятия жестких ограничений на порядок обмена данными между процессами. Как следствие, недетерминизм позволяет достигнуть более высокой производительности и полнее использовать потенциальные возможности параллельной вычислительной системы.

В некоторых случаях недетерминизм является необходимым свойством программы, например, при моделировании объектов или явлений реального мира. Если компьютерная система симулирует некоторое физическое явление с недетерминированными характеристиками, то моделирующая программа должна обладать аналогичными свойствами.

Потенциальным источником ошибок в недетерминированных программах является ситуация, когда разработчик не

оценил все возможные варианты поведения программы при выборе той или иной альтернативы в точках недетерминизма. Программа может корректно функционировать длительный период времени, прежде чем проявится ошибка. Чаще всего это случается при изменении внешних условий, например, после смены программно-аппаратного окружения, в котором функционирует параллельная программа. Ошибки этого класса крайне сложны в локализации.

Определение 3. Программа является недетерминированной, если ее успешные выполнения на одних и тех же допустимых входных данных могут приводить к различным корректным результатам.

Под результатами следует понимать как окончательный ответ, так и временные результаты вычислений, а также порядок взаимодействия между процессами. Правильный ответ для каждого фиксированного набора входных данных может быть получен с использованием различных временных результатов и различного порядка обмена сообщениями между выполняющимися процессами. Простым примером может быть параллельная программа, суммирующая элементы массива, распределенного между процессами. Каждый процесс суммирует свою часть и передает результат головному процессу. Тот, в свою очередь, суммирует все принятые числа и выдает окончательный ответ. Вне зависимости от порядка приема сообщений результат будет одним и тем же.

Источники недетерминизма достаточно разнообразны:

- неравномерная загрузка узлов вычислительной системы;
- влияние менеджера процессов операционной системы на каждом узле;
- механизм виртуальной памяти;
- задержки и конфликты в сети.

В MPI программах влияние этих источников проявляется в последовательности приема сообщений для тех операций *MPI_Recv* (и аналогичных), где передающий процесс определяется по маске (например, *MPI_ANY_SOURCE*), а также результатах выполнения асинхронных операций. Кроме того, недетерминизм выполнения программы (как параллельной, так и последовательной) может вызываться обращением к различным системным функциям, таким как генератор псевдослучайных чисел.

Недетерминизм программы требует от отладочных средств функциональных возможностей по решению следующих задач:

1. воссоздание поведения параллельной программы, записанного на стадии тестирования, для повторного детерми-

нированного выполнения. В противном случае ошибка, обнаруженная на тестовом наборе, может не проявиться при выполнении программы под отладчиком;
2. анализ всех теоретически возможных вариантов выполнения недетерминированной параллельной программы. В силу наличия точек недетерминизма параллельная программа допускает изменение порядка передаваемых сообщений между процессами, при этом выполнение программы должно приводить к корректным результатам;
3. минимизация влияния отладочных средств на поведение отлаживаемой программы.

3.1. Воссоздание поведения параллельной программы. Процесс тестирования и отладки программы носит итеративный характер. На первой итерации программа выполняется на тестовом наборе. Если результат ее выполнения не является корректным, то программа содержит ошибку (ошибки), требующую локализации, идентификации и последующего исправления. Для этого программа выполняется под отладчиком, позволяющим анализировать ее поведение и просматривать значения переменных и другие характеристики процессов в заданных контрольных точках.

Наилучшим решением задачи локализации ошибки было бы обратное выполнение программы от точки получения некорректного результата до точки возникновения ошибки. Однако в большинстве случаев обратить выполнение программы не представляется возможным, например, при работе с файлами. Поэтому традиционной методикой является поиск ошибки при помощи метода последовательного приближения. На каждой итерации данного процесса пользователь при помощи отладчика приостанавливает выполнение программы в точке вероятного возникновения ошибки и анализирует состояние процесса. Если собранной информации достаточно для исправления ошибки, то процесс отладки останавливается, исходные тексты редактируются, и проводится повторное тестирование. В противном случае контрольная точка сдвигается вперед или назад относительно текущей итерации и программа перезапускается (при сдвиге вперед перезапуск не требуется).

Особенность поведения недетерминированной параллельной программы состоит в том, что ее успешные выполнения на допустимых фиксированных наборах входных данных могут приводить к различным корректным результатам (временным, окончательным). Порядок обмена сообщениями между процессами также может варьироваться. В такой ситуации выполнение программы на тестовом наборе входных данных

может оказаться невоспроизводимым во время отладки. Во-первых, даже если в ходе тестирования была обнаружена ошибка, она может не проявиться под отладчиком, и ее локализация на стадии отладки может оказаться крайне трудоемкой. Во-вторых, ошибка, обнаруженная на тестовом наборе, может быть подменена совершенно другой (другими) ошибкой во время итеративного выполнения программы под отладчиком. Поэтому крайне важной является возможность отладчика воспроизводить в точности то поведение параллельной программы, которое было зафиксировано в ходе тестирования.

Традиционное решение этой задачи состоит в использовании двухшагового механизма "*записи и воспроизведения*" ("*record and replay*") [11]. Суть этого метода состоит в трассировке относительного порядка возникновения событий в параллельной программе, связанных со взаимодействием процессов. На первом шаге (стадия записи), как правило, в ходе выполнения программы на тестовом наборе, для каждого процесса в трассировочный файл сохраняется вся необходимая информация, описывающая, в каком порядке данный процесс принимал сообщения от других процессов программы. При воспроизведении отладчик использует сохраненную информацию и контролирует во всех точках появление точно того же события, что и на стадии записи. Например, при выполнении на тестовом наборе было зафиксировано, что в точке обращения к функции *MPI_Recv* с параметром *MPI_ANY_SOURCE* реально было принято сообщение от процесса с номером "3"; отладчик сохраняет этот номер в трассировочный файл. На последующих итерациях воспроизведения (возможно не на одной) отладчик фиксирует обращение к этой функции в данной точке, извлекает реальный номер отправителя из трассировочного файла и подменяет параметр *MPI_ANY_SOURCE* на конкретный номер процесса отправителя.

Использование стандартного профилировочного интерфейса MPI для реализации механизма "записи и воспроизведения" не позволяет обрабатывать недетерминизм, вызываемый обращением к различным системным функциям, таким как генератор псевдослучайных чисел. Полная реализация данного механизма требует инструментирования (внесения изменений) исходных текстов программы и фиксирования возвращаемых недетерминированными системными функциями значений, а это существенно снижает переносимость и удобство использования отладочной системы. В большинстве случаев обработка коммуникационных операций является достаточной.

3.2. Получение всех вариантов выполнения недетерминированной параллельной программы. Поведение недетерминированной параллельной программы характерно тем, что ее успешные выполнения на любом допустимом фиксированном наборе входных данных могут приводить к различным корректным результатам. Варианты поведения параллельной программы, обозреваемые в ходе тестирования, являются наиболее вероятными при данном внешнем окружении, но далеко не всеми теоретически возможными. В силу того, что на поведение недетерминированной программы влияют параметры внешнего окружения, которые зачастую носят случайный характер, необходимо иметь возможность анализировать корректность результатов, которые могут быть получены при других, альтернативных выполнениях параллельной программы.

Подавляющее большинство подходов к решению этой задачи основывается на устранении недетерминизма в программе с последующим полным перебором заданных некоторым образом вариантов ее выполнения. Все недетерминированные операции в программе заменяются их детерминированными аналогами. Каждая замена допускает несколько (зачастую значительное число) вариантов, в результате чего, объединяя все возможные подстановки, образуется множество теоретически возможных вариантов выполнения программы. Каждый вариант выполнения может быть описан в виде детерминированной трассы событий. После такой подготовки параллельная программа итеративно воспроизводится в соответствии с полученными трассами и результаты ее выполнения проверяются на корректность.

Множество всех трасс может быть специфицировано вручную [12], автоматизированным образом при помощи специальных правил, описывающих коммуникационную структуру программы [13], или автоматически, используя полный перебор всех возможных трасс для данного тестового набора [14].

Ручной подход имеет смысл применять только в том случае, если множество всех трасс конечно и невелико. Второй подход требует от разработчика дополнительно специфицировать на специальном языке коммуникационную структуру программы, в то время как она уже описана на некотором языке программирования. Это накладывает дополнительные обязанности на разработчика. Кроме того, спецификация сама по себе требует тестирования; также необходимо доказать, что она полностью описывает все возможные варианты выполнения программы для любого допустимого набора входных данных. В противном случае использование такой специфика-

ции не гарантирует решения исходной задачи.

Метод автоматического определения освобождает разработчика от необходимости каким-либо образом описывать трассы событий или специфицировать коммуникационную структуру программы. Основная идея этого подхода состоит в следующем. Сначала программа выполняется на тестовом наборе и все события, необходимые для ее воспроизведения, записываются в трассировочный файл. На втором этапе файл анализируется, в нем выделяются все точки, в которых поведение программы не детерминировано, и автоматически генерируется набор трассировочных файлов для последующего воспроизведения программы. Каждый сгенерированный файл получается из исходного путем перестановки двух или более событий, соответствующих точкам недетерминизма параллельной программы. На последнем этапе программа воспроизводится в соответствии с каждым полученным трассировочным файлом, и результаты выполнения проверяются на корректность.

На практике число всех возможных вариантов выполнения может быть настолько большим, что полное тестирование окажется практически нереализуемым, например, если в программе есть обращение к генератору псевдослучайных чисел. Важно отметить, что каждое воспроизведение программы в соответствии с некоторым видоизмененным файлом может породить дополнительные точки недетерминизма, которые также должны быть учтены разработчиком наравне с исходным оригинальным поведением программы. Поэтому основная задача отладочной среды состоит не только в автоматической генерации всех возможных вариантов выполнения недетерминированной параллельной программы, но и в выделении среди них наиболее важных для тестирования. В противном случае этап тестирования программы может затянуться на неопределенно большой интервал времени.

3.3. Минимизация влияния отладчика. Отладка программы, как правило, проводится с привлечением дополнительных средств, позволяющих разработчику анализировать состояния процессов в заданных контрольных точках. Состояния программы могут быть просмотрены во время выполнения при помощи отладчика или с использованием операторов вывода значений переменных на экран монитора, в файл, на принтер и т.д. Вывод значений может осуществляться непосредственно вставкой разработчиком соответствующих операторов в код программы или различными средствами инструментирования исходных текстов. В любом случае, отладочные средства вносят задержки в выполнение параллельной про-

граммы и ее поведение становится отличным от оригинального. Это особенно критично для недетерминированных программ, так как влияние отладчика может проявиться в изменении порядка передаваемых между процессами сообщений, в результате чего могут пропасть обнаруженные в ходе тестирования ошибки.

Очевидно, что полностью устранить влияние отладчика на поведение отлаживаемой программы невозможно, потому что само присутствие отладчика в памяти компьютера искажает оригинальное поведение программы, поэтому можно говорить только о минимизации влияния отладчика на отлаживаемую программу. Степень влияния отладочных средств определяется суммарным временем работы отладочных функций и размером памяти, занимаемым ими. Эта проблема активно исследовалась во многих научных работах. Предлагались различные методики, связанные с корректировкой трассы событий, полученной после завершения работы программы [15], введением дополнительных часов, останавливающий свой ход во время работы отладочных функций [16], и другие. Однако для недетерминированных программ обычно используется двухэтапный метод отладки [17]. На первом этапе программа выполняется с минимально необходимым числом отладочных функций, задача которых состоит в трассировке только самой необходимой информации для детерминированного воспроизведения программы на следующем этапе. Несмотря на внесение некоторых изменений во временны́е характеристики отлаживаемой программы, это выполнение считается эталонным. На втором этапе производится полноценная отладка с использованием всех необходимых средств анализа поведения программы. В силу того, что воспроизводимое на втором этапе поведение программы является детерминированным, отладчик не влияет на порядок передаваемых сообщений, а затрагивает лишь временны́е характеристики программы. Для большинства программ это не является существенным.

§ 4. Заключение

Метод последовательного выполнения является одним из способов планирования времени выполнения процессов и реализует в распределенной параллельной программе механизм разделения времени. Передача управления от одного процесса другому производится только при выполнении активным процессом коммуникационной операции MPI. При этом достигается возможность использования произвольного последовательного отладчика и всей семантики отладки последовательных

программ для параллельной MPI программы, на которую не наложены жесткие временны́е рамки (например, системы реального времени). На интервале между двумя обращениями к функциями библиотеки MPI пользователь может проводить пошаговую трассировку активного процесса.

Механизм последовательного выполнения реализуется на базе стандартного профилировочного интерфейса MPI и не требует вносить изменения в исходные тексты отлаживаемой программы и библиотеки MPI.

Библиотека поддержки выполнения, реализованная предложенным образом, не зависит от конкретного отладчика и может использоваться любым средством анализа производительности параллельных MPI программ.

ЛИТЕРАТУРА

1. High Performance Debugging Forum. HPD Version 1 Standard: Command Interface for Parallel Debuggers/ Под ред. Pancake, C. и Francioni// J. Technical Report CSTR-97. Dept. of Computer Science, Oregon State University, 1997.
2. Dolphin Interconnect Solutions, Inc. TotalView Multiprocess Debugger User's Guide. Версия 3.7.7. 1997.
3. Hood R. The p2d2 Project: Building a Portable Distributed Debugger// Proceedings of the SIGMETRICS Symposium on Parallel and Distributed Tools. Philadelphia, 1996.
4. Аветисян А.И., Арапов И.В., Гайсарян С.С., Падарян В.А. Параллельное программирование с распределением по данным в системе ParJava// Вычислительные методы и программирование. Т. 2. 2001.
5. Think Machines Corporation. Prism 2.0 Release Notes. 1994.
6. Intel Corporation. iPSC/2 and iPSC/860 Interactive Parallel Debugger Manual. 1991.
7. Kranzlmuller D. Using Sequential Debuggers for Parallel Programs// Proceedings of PDCS 2001, 14th Intl. Conf. on Parallel and Distributed Computing Systems. Richardson, Texas, USA, 2001.
8. Message Passing Interface Forum. MPI: A Message Passing Interface Standard, chpt. 8. Profiling Interface. International Journal of Supercomputing Applications. 8, № 3/4. 1994.
9. Gligor V.D., Shattuck S.H. On Deadlock Detection in Distributed Systems// IEEE Transactions on Software Engineering. **6**, № 5. 1980.
10. Stein R.M. Does Determinism Dictate Dignity?// IEEE Computer. **31**, № 3. 1998.

11. LeBlanc T.J., Mellor-Crummey J.M. Debugging Parallel Programs with Instant Replay// IEEE Transaction on Computing. C-36. № 4. 1987.
12. Tai K.C., Carver R.H. Testing Distributed Programs// Parallel and Distributed Computing Handbook. Zomaya A.Y. (eds.), McGraw-Hill, New York, 1996.
13. Oberhuber M. Elimination of Nondeterminacy for Testing and Debugging Parallel Programs// Proceedings of AADEBUG '95, 2nd International Workshop on Automated and Algorithmic Debugging. Saint Malo, France, 1995.
14. Kranzlmuller D., Volkert J. NOPE: A Nondeterministic Program Evaluator// Zinterhof P., Vajtersic M., Uhl A. (eds.)// Parallel Computation. Proceedings of ACPC '99, 4th Intl. ACPC Conference, Lecture Notes in Computer Science. **1557**. Springer, Salzburg, 1999.
15. Malony A.D., Reed D.A., Wijshoff H. Performance Measurement Intrusion and Perturbation Analysis// IEEE Transactions on Parallel And Distributed Systems. **3**. № 4. 1992.
16. Cai W., Turner S.J. An Approach to the Run-Time Monitoring of Parallel Programs// The Computer Journal, British Computer Society. **37**. № 4. 1994.
17. Zheng Q., Chen G., Huang L. Optimal Record and Replay for Debugging of Nondeterministic MPI/PVM Programs// Proceedings of the Fourth International Conference/Exhibition on High Performance Computing in Asia-Pacific Region, 2000.