ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 004.725.7

В.П. Корячко, Ю.Л. Ижванов, А.П. Шибанов

МЕТОД РАСЧЕТА ХАРАКТЕРИСТИК ВЫСОКОСКОРОСТНЫХ ОПОРНЫХ КАНАЛОВ РОССИЙСКОЙ УНИВЕРСИТЕТСКОЙ СЕТИ RUNNet

Приводятся характеристики Федеральной университетской компьютерной сети RUNNet, формулируются проблемы повышения качества ее функционирования. Предлагается метод расчета вероятностно-временных характеристик опорного высокоскоростного канала сети при условиях использования схемы защиты канала и контроля вариации задержек пакетов.

Ключевые слова: *Федеральная университетская компьютерная сеть, агрегированный канал с защитой, производящая функция моментов, генетические алгоритмы.*

Введение. Федеральная университетская компьютерная сеть RUNNet (Russian UNiversity Network) поддерживает единую образовательную информационную среду России и обеспечивает ее интеграцию в мировое информационное сообщество. В настоящее время сеть RUNNet является крупнейшей российской научно-образовательной IP-сетью, предоставляющей услуги более чем 500 университетам и другим крупным образовательным и научно-исследовательским учреждениям. В число пользователей RUNNet входят такие крупные научно-образовательные сети, как RBnet, FREEnet, RUHEP/Radio-MSU, RELARN-IP, сети Московского и Санкт-Петербургского государственных университетов. Общее количество пользователей сети RUNNet по независимым экспертным оценкам составляет около 2 000 000 человек, что делает ее одной из крупнейших компьютерных сетей России [1]. В состав опорной инфраструктуры RUNNet входят федеральные узлы сети в Москве, Санкт-Петербурге, Самаре, Новосибирске, Хабаровске, Екатеринбурге, Нижнем Новгороде, Ростове-на-Дону, Владивостоке и региональные узлы в университетах. Для обеспечения внутрироссийской магистральной связности используются цифровые каналы связи ЗАО «Компания ТрансТелеКом», ОАО «Ростелеком», ЗАО «Синтерра» и других канальных операторов.

Узлы RUNNet, находящиеся на площадках научно-образовательной сети стран Северной Европы (NORDUnet, Стокгольм), Института субатомной физики Нидерландов (NIKHEF, Амстердам) и Суперкомпьютерного центра Финляндии (CSC, Хельсинки), обеспечивают доступ RUNNet и всех научно-образовательных сетей России в международные межнациональные научно-образовательные компьютерные сети.

Сеть RUNNet функционирует на основе технологии плотного волнового мультиплексирования (Dense Wavelength Division Multiplexing – DWDM). Запущенная DWDM-система соединяет российские научно-образовательные сети с сетями Европы и США высокоскоростной магистралью с возможностью использования до 72 длин волн, при этом общая скорость пропуска трафика может составить 7200 Гбит/с.

С помощью DWDM-инфраструктуры научно-образовательной сети стран Северной Европы (Дания, Швеция, Норвегия, Финляндия, Исландия) NORDUnet каналы сети RUNNet продляются до Стокгольма (узел доступ сети NORDUnet) и Амстердама (Институт субатомной физики NIKHEF). Созданная инфраструктура будет использоваться для доступа российских научно-образовательных сетей к научно-образовательным сетям Европы и США,

обеспечения связности с сетями консорциума GLIF (Global Lambda Integrated Facility), поддержки проекта GLORIAD (Global Ring for Advanced Application). Инсталлированная инфраструктура позволит российским ученым более эффективно участвовать в ряде международных проектов, в частности проекте БАК (Большой Адронный Коллайдер), EGI, PRACE и других, в которых пропускная способность существующих сетей недостаточна.

В процессе выполнения работ постоянно осуществляется качественное развитие и существенное улучшение возможностей учреждений науки и высшей школы по доступу к международным научно-образовательным информационным ресурсам за счет модернизации опорной инфраструктуры сети RUNNet на базе внедрения волоконно-оптических технологий нового поколения.

Это реализуется посредством:

- выработки системных проектных решений по оптимизации обмена информацией между отечественными и международными органи-зациями науки и образования и развитию волоконно-оптической канальной инфраструктуры национальной компьютерной сети RUNNet;
- разработки проектов опорных узлов и программного обеспечения системы мониторинга и управления сети RUNNet;
- реализации системных решений, обеспечивающих адекватные современным потребностям научно-образовательного сообщества требования по совершенствованию инфраструктуры сетевой поддержки научных исследовании, хранения и передачи новых знаний;
- разработки и реализации проектов волоконно-оптической инфраструктуры международного сегмента сети RUNNet, программных комплексов опорных узлов, программного обеспечения системы мониторинга и управления.

Цель работы – разработка метода расчета характеристик высокоскоростных опорных каналов RUNNet с комплексным учетом характеристик надежности (коэффициента готовности подканалов), задержки передачи и ее вариации, а также стоимостных характеристик оборудования в режиме передачи синхронного трафика: аудио-, высококачественного видео-, телеконференций, телефонных переговоров по Интернет, информации Web-камер и т.п.

Теоретические исследования. Информация в звене между двумя мультиплексорами может передаваться по защищаемому и защищающему подканалам. Резервирование осуществляется по схеме 1+1. В момент поступления очередного кадра анализируется состояние подканалов. Если оба подканала работоспособны, то пакет делится на две части и каждая из них передается параллельно по двум подканалам. Если только один из подканалов готов к работе, то пакет передается по этому подканалу. Если оба подканала неработоспособны, то фиксируется отказ агрегированного канала.

Для передачи через IP-сеть данные разбиваются на пакеты, которые часто прибывают в пункт назначения с разбросом по времени доставки (с джиттером), который снижает качество передачи. Различают три формы джиттера: из-за ограниченной полосы пропускания, из-за колебания задержки передачи между конечными абонентами в разных направлениях, как результат теплового шума.

Для расчета величины джиттера используем нормированное распределение Эрланга. Функция и плотность нормированного распределения Эрланга определяются выражениями:

$$F_k(x)=1-e^{-k\alpha x}\sum_{i=0}^{k-1}\frac{(k\alpha x)^i}{i!},$$

$$f_k(x)=\frac{k\alpha(k\alpha x)^{k-1}}{(k-1)!}e^{-k\alpha x}.$$

Время передачи пакета номинальной длины по подканалу охарактеризуем производящей функцией моментов $M(s)=e^{sT}(k\alpha/k\alpha-s)^k$, где T – постоянная составляющая времени передачи (без учета джиттера). Распределение времени передачи пакета в одном звене (что равносильно оценке джиттера) находится экспериментально. Для обеспечения максимальной степени соответствия этой функции и используемого для ее аппроксимации нормированного распределения Эрланга выполняется подбор значения параметра распределения α. Степень соответствия этих двух функций оценивается по критерию χ^2. Плотность распределения вероятностей времени передачи пакета по подканалу находится через вычеты с вычислением интеграла по контуру Бромвича [2–4]

$$\varphi(t)=\frac{1}{2\pi i}\int_{-i\infty}^{i\infty}e^{zt}\Phi_E(z)dz=$$

$$=\frac{1}{2\pi i}\int_{-i\infty}^{i\infty}e^{z(t-T)}\left(\frac{k\alpha}{k\alpha+z}\right)^k dz.$$

Для практических расчетов достаточно принять $k=2$, тогда

$$\varphi(t)=4\alpha^2(t-T)e^{-2\alpha(t-T)},\ t\ge T,$$

а

$$F(t)=1-e^{-2\alpha(t-T)}[1+2\alpha(t-T)],\ t\ge T.$$

Если длина пакета уменьшается в r раз, то переменная s заменяется на s/r. Тогда

$$M_{(r)}(s)=\exp(sT/r)\left[rk\alpha/(rk\alpha-s)\right]^k.$$

Плотность распределения времени передачи по подканалу пакета, длина которого уменьшена в r раз,

$$\varphi_{(r)}(t)=4r^2\alpha^2\left[t-T/r\right]\exp\left\{-2r\alpha\left[t-T/r\right]\right\},$$

а функция распределения

$$F_{(r)}(t)=1-\exp\left[-2\alpha(rt-T)\right]\left[2\alpha(rt-T)+1\right],$$

$$t-\frac{T}{r}\ge 0.$$

Рассмотрим параллельную работу двух подканалов ($r=2$) при равномерном распределении байт пакета по подканалам. Тогда время передачи пакета определяется как максимальная из двух случайных величин, а его функция распределения вероятностей

$$F_{(2)}(t)=\left\{1-e^{-2\alpha(2t-T)}\left[2\alpha(2t-T)+1\right]\right\}^2,\ t-\frac{T}{2}\ge 0.$$

Соответствующая плотность распределения вероятностей

$$\varphi_{(2)}(t)=16\alpha^2(2t-T)\,e^{-2\alpha(2t-T)}\times$$
$$\times\left\{1-e^{-2\alpha(2t-T)}\left[1+2\alpha(2t-T)\right]\right\},\ t-\frac{T}{2}\ge 0.$$

По данному выражению с использованием теории GERT-сетей (*Graphical Evaluation and review technique*) [5–7] найдена производящая функция моментов

$$M_{(2)}(s)=$$
$$=32\alpha^2e^{s\frac{T}{2}}\left[\frac{1}{(4\alpha-s)^2}-\frac{1}{(8\alpha-s)^2}-\frac{8\alpha}{(8\alpha-s)^3}\right].$$

Математическое ожидание $\mu_{1(2)}$ и дисперсия $\sigma^2_{(2)}$ времени передачи пакета одновременно по двум подканалам вычисляются через первый и второй моменты функции $M_{(2)}(s)$ относительно начала координат:

$$\mu_{1(2)}=\frac{T}{2}+\frac{11}{16\alpha};\ \sigma^2_{(2)}=35/256\alpha^2.$$

Если в агрегированном канале с защитой 1+1 подканалы имеют одинаковую вероятность отказа $1-q$, то вероятность работоспособности обоих подканалов есть q^2, с вероятностью $(1-q)^2$ фиксируется отказ агрегированного канала, с вероятностью $1-(1-q)^2$ работает хотя бы один подканал.

Эквивалентная W - функция времени передачи пакета по агрегированному каналу с защитой 1+1

$$W_{(1+1)}(s)=\frac{2(1-q)}{2-q}\cdot e^{sT}\left(\frac{2\alpha}{2\alpha-s}\right)^2+$$
$$+\frac{32\alpha^2q}{2-q}e^{s\frac{T}{2}}\left[\frac{1}{(4\alpha-s)^2}-\frac{1}{(8\alpha-s)^2}-\frac{8\alpha}{(8\alpha-s)^3}\right].$$

По данному выражению рассчитывается плотность распределения вероятностей агрегированного канала с защитой 1+1:

$$\varphi_{(1+1)}(t)=\frac{2(1-q)}{2-q}f_I(t)+\frac{q}{2-q}f_{II}(t),$$

где

$$f_I(t)=\begin{cases}0, & t<T,\\ \varphi_I(t), & t\ge T,\end{cases}\quad f_{II}(t)=\begin{cases}0, & t<T/2,\\ \varphi_{II}(t), & t\ge T/2;\end{cases}$$

$$\varphi_I(t)=4\alpha^2(t-T)e^{-2\alpha(t-T)};$$

$$\varphi_{II}(t)=16\alpha^2\left(t-\frac{T}{2}\right)e^{-2\alpha\left(t-\frac{T}{2}\right)}\times$$
$$\times\left\{1-e^{-2\alpha\left(t-\frac{T}{2}\right)}\left[1+2\alpha\left(t-\frac{T}{2}\right)\right]\right\}.$$

По эквивалентной W - функции времени передачи найдены среднее время $t_{\text{ср}}$ и дисперсия времени передачи пакета через агрегированный канал связи с защитой 1+1:

$$t_{\text{ср}\,(1+1)}=\mu_{1(1+1)}=\frac{1}{2(2-q)}\left[(4-3q)T+\frac{32-21q}{8\alpha}\right],$$

$$\sigma^2_{(1+1)}=\frac{1}{8(2-q)^2}\times$$
$$\times\left[q(1-q)\left(4T^2+5\frac{T}{\alpha}\right)+\frac{512-648q+171q^2}{32}\cdot\frac{1}{\alpha^2}\right].\quad(1)$$

Эквивалентная производящая функция моментов времени передачи пакета по тракту, состоящему из нескольких агрегированных каналов с защитой 1+1, равна произведению производящих функций моментов времени передачи пакета по отдельным каналам. Среднее время передачи через канал, состоящий из нескольких звеньев, равно сумме средних задержек в отдельных звеньях. Аддитивными величинами являются и дисперсии времени передачи по отдельным звеньям.

При проведении оптимизации сети должны приниматься во внимание и производимые затраты. Пусть переменные $c_{\text{тек}}$ и $c_{\text{зад}}$ обоз-

начают соответственно уже вложенные на данный момент затраты для обеспечения коэффициента готовности $q_{\text{тек}}$ и затраты, которые необходимо произвести для доведения его до величины $q_{\text{зад}}$. Эти затраты будем считать в относительных единицах, т.е. $0 \le c \le 1$. Величина затрат как функция коэффициента готовности q чаще всего описывается линейной либо степенной зависимостями.

Для отражения технического состояния отдельных подканалов достаточно рассматривать диапазон изменения значений аргумента $0{,}9 \le q \le 1$. В разных звеньях эти затраты будут по абсолютной величине разными. С учетом вышесказанного увеличение стоимости в относительных единицах может быть выражено через коэффициент готовности подканала по формуле

$$c = \left[(q - 0{,}9)/0{,}1\right]^k, \quad 0{,}9 \le q \le 1, \quad 0 \le c \le 1.$$

На рисунке 1 изображены функции $c = f(q)$ в зависимости от значения k.

В качестве оптимизируемых параметров выберем оценку джиттера и относительную стоимость затрат, а на математическое ожидание времени передачи пакета наложим ограничение

$$\mu_{1\,\text{защ}} = \frac{1}{2(2-q)}\left[(4-3q)\,T + \frac{32-21q}{8\alpha}\right] \le \mu_{1\,\text{пр}},$$

где $\mu_{1\,\text{пр}}$ – предельно допустимая величина среднего времени передачи пакета.

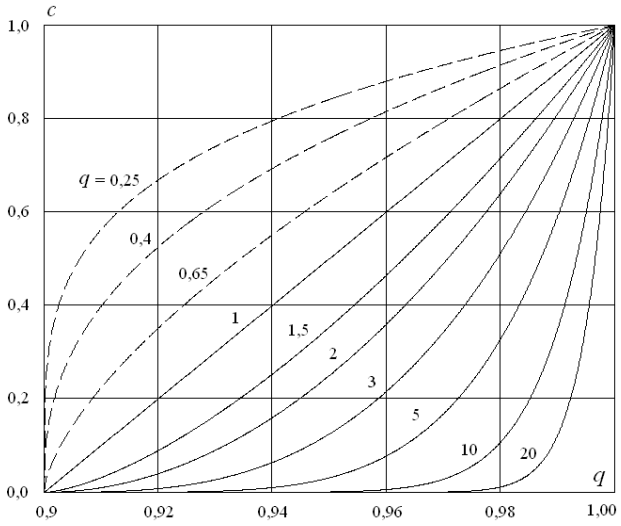


Рисунок 1 – Семейство функций $c = f(q)$ при различных значениях k

Построим кривую возможных компромиссов для критериев оценки джиттера D и стоимости

$$c = \left[(q - 0{,}9)/0{,}1\right]^k, \; 0{,}9 \le q \le 1, \; 0 \le c \le 1$$

при условии изменения параметра q, т.е. $\{D, c\} = f(q)$. Рассмотрим вид кривой при типовых значениях параметров T и α. Для этого найдем зависимость $c = f(D)$, исключив переменную q. Используя формулу (1) для расчета дисперсии времени передачи пакета через произвольное звено, находим выражение для оценки джиттера D через интервал “трех сигм”, и решая систему уравнений

$$\left.\begin{aligned} D &= 3\sqrt{\left[q(1-q)\left(4T^2 + 5\frac{T}{\alpha}\right) + \frac{512 - 648q + 171q^2}{32\alpha^2}\right] \Big/ 8(2-q)^2}, \\ c &= \left[(q-0{,}9)/0{,}1\right]^k, \quad 0{,}9 \le q \le 1, \quad 0 \le c \le 1, \end{aligned}\right\}$$

получаем выражение для вычисления величины c как функции переменной D:

$$c = \left(5 \cdot \frac{\beta + 32D^2 + \sqrt{\chi - \delta D^2}}{8D^2 + \varepsilon} - 9\right)^k,$$

где

$$\varepsilon = 36T^2 + \frac{45T}{\alpha} - \frac{1539}{32\alpha^2},$$

$$\beta = 36T^2 + 45\frac{T}{\alpha} - \frac{729}{4\alpha^2},$$

$$\chi = \frac{9639T^2}{\alpha^2} + \frac{19035T}{2\alpha^3} + \frac{88209}{16\alpha^4} + 1296T^4 + \frac{3240T^3}{\alpha},$$

$$\delta = \frac{900}{\alpha^2} + 2304T^2 + \frac{2880T}{\alpha}.$$

На рисунке 2 представлен общий вид графиков функций $c = f(D)$ при заданных значениях параметров канала с защитой и разных значениях коэффициента k. Функции $c = f(D)$ являются строго убывающими, а это означает, что при любом подобранном значении k можно на кривой компромисса $c = f(D)$ выбрать точку, которая и будет определять оптимальное решение.

На рисунке 3 изображены три зависимости $c = f(D)$, соответствующие значениям: k = 0,25 [вогнутая зависимость $c = f(q)$], k = 1 [линейная зависимость $c = f(q)$] и k = 7 [выпуклая зависимость $c = f(q)$].

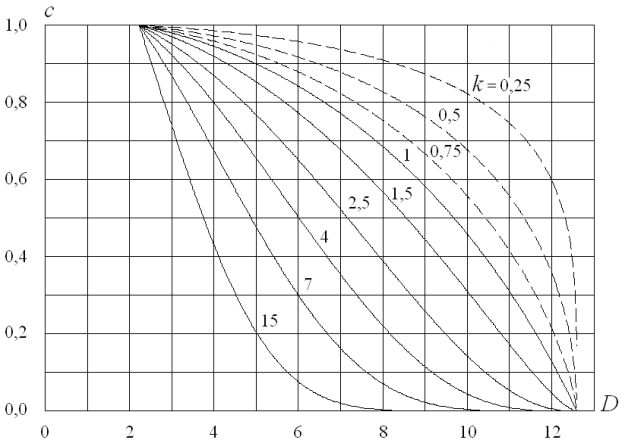


Рисунок 2 – Функция $c = f(D)$ для канала с параметрами (T = 20 мс; α = 0,5 мс$^{-1}$)

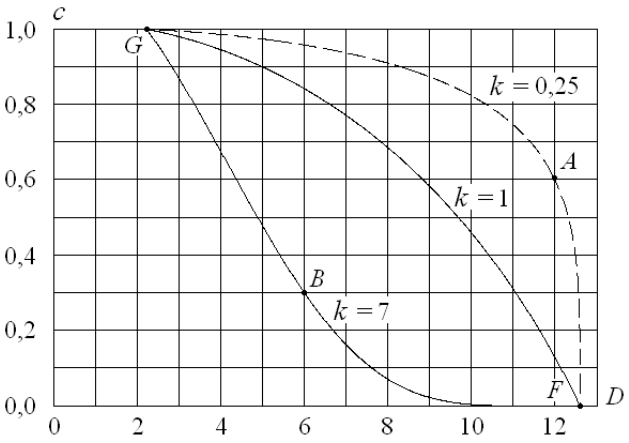


Рисунок 3 – Выбор точки компромисса между критериями c и D

Если структура затрат такова, что ее можно описать выпуклой зависимостью $c = f(q)$, то на участке $[F, A]$ вложение значительных затрат пока еще не дает заметного уменьшения джиттера. Проведение мероприятий по дополнительной отладке канала (уже не требующих значительных новых вложений) существенно уменьшает джиттер в канале с защитой (участок кривой $[A, G]$).

При структуре затрат, характеризующейся выпуклой зависимостью $c = f(q)$, на участке $[F, B]$ последовательное уменьшение джиттера требует нелинейного увеличения затрат. Однако на участке $[B, G]$ зависимость $c = f(D)$ близка к линейной, несмотря на то, что соответствующая кривая $c = f(q)$ является выпуклой. Это объясняется тем, что относительная стоимость рассматривается в зависимости от изменения коэффициента готовности одного подканала, а величина джиттера относится к агрегированному каналу с защитой. В общем случае линейный участок может и отсутствовать.

При структуре затрат, характеризующейся линейной зависимостью ($c = f(q)$, $k = 1$) функция $c = f(D)$, в общем случае не является линейной и занимает промежуточное положение между кривыми, характеризующимися значениями $k = 0{,}25$ и 7.

Увеличение относительной стоимости c для повышения коэффициента готовности подканала q обязательно приводит к уменьшению джиттера в агрегированном канале с защитой. Это позволяет сделать вывод о том, что наилучшее решение оптимизационной задачи по всему тракту «из конца в конец» может быть получено путем суперпозиции локальных решений, найденных в пределах отдельных звеньев.

Заключение. Полученные выражения для расчета среднего времени передачи пакета и дисперсии времени передачи между терминальными мультиплексорами RUNNet можно использовать для формирования функции полезности генетического алгоритма при оптимизации структуры и параметров сетей типа «цепь» (состоит из последовательно соединенных звеньев) и «кольцо». При этом учитывается также и стоимость оборудования. Такая оптимизация выполняется на первой стадии моделирования. Ее целью является подбор множества подканалов с такими коэффициентами готовности, параметрами быстродействия и стоимости, чтобы целевая функция достигала экстремального значения.

Более полное моделирование с учетом возможного влияния очередей для различных вариантов структуры сети может быть осуществлено на втором этапе с использованием средств компьютерной имитации [8, 9]. Для получения выборочных значений времени передачи пакета через звенья применяется метод обратной функции. Для его использования нужно знать соответствующие функции распределения вероятностей через звенья с защитой и без защиты, которые и получены в данной работе.

Измерения, проведенные на высокоскоростном фрагменте сети RUNNet с использованием технологии Gigabit Ethernet на магистральной части сети и на «последних милях» к пользователю при пропускной способности TCP-сессии в 10 Мбит/с, показали, что применение предлагаемого метода позволяет: сократить потери пакетов с величины порядка 1 процента до значения не более 0,5 процента; уменьшить одностороннюю задержку при передаче пакетов со 150 мс до 100 мс, а колебания задержки с 30 мс до 20 мс.

Работа поддержана Российским фондом фундаментальных исследований, грант 11-07-00121-а.

Библиографический список

1. *Ижванов Ю.Л., Гугель Ю.В.* Сравнительный анализ характеристик российских и международных научно-образовательных сетей // Информатизация образования и науки, 2009. № 1. – С. 28–33.

2. *Пчелин Б.К.* Специальные разделы высшей математики. – М.: Высшая школа, 1973. – 464 с.

3. *Привалов И.М.* Введение в теорию функций комплексного переменного. – М.: Наука, 1984. – 432 с.

4. *Маркушевич А.И., Маркушевич Л.А.* Введение в теорию аналитических функций. – М.: Просвещение, 1977. – 320 с.

5. *Pritsker A.A.*. GERT" Graphical evaluation and review technique. Memorandum RM-4973-NASA, 1966.

6. Шибанов А.П. Нахождение плотности распределения времени исполнения GERT-сети на основе эквивалентных упрощающих преобразований // Автоматика и телемеханика, 2003. № 2. – С. 117–126.

7. *Корячко В.П., Кравчук Н.В., Шибанов А.П., Шибанов В.А.* Основы теории GERT-сетей. – М.: Горячая линия – Телеком, 2010. – 207 с.

8. *Корячко В.П., Шибанов А.П., Шибанов В.А., Чернышев А.С.* Имитационная система моделирования телекоммуникационных сетей // Телекоммуникации, 2001. № 10. – С. 28–32.

9. *Шибанов А.П.* Разработка программного обеспечения для моделирования локальных сетей Ethernet // Программирование, 2002. № 6. – С. 62–71.

УДК 681.31

А.Н. Пылькин, Р.В. Тишкин, С.В. Труханов

ЗАДАЧИ DATA MINING И ИХ РЕШЕНИЕ В СОВРЕМЕННЫХ РЕЛЯЦИОННЫХ СУБД

Рассматриваются технологии и основные задачи интеллектуальной обработки данных «Data Mining». Приведена постановка задач обработки информации и поддержки интеллектуальной обработки информации с использованием реляционных структур СУБД.

Ключевые слова: *алгоритм, данные, интеллектуальная обработка информации, Data Mining.*

Введение. Сущность интеллектуального анализа данных можно охарактеризовать как технологию, которая предназначена для поиска в больших объемах данных неочевидных, объективных и полезных на практике закономерностей [1].

С точки зрения применения в реляционных СУБД под термином «интеллектуальный анализ данных» (Data Mining - DM, добыча данных) необходимо понимать процесс анализа данных с целью нахождения скрытых шаблонов с помощью автоматических методик.

Цель работы: рассмотреть основные этапы процесса DM, провести классификацию существующих алгоритмов, рассмотреть алгоритмы DM, используемые в СУБД Microsoft SQL Server и СУБД Oracle, и выявить требующие решения задачи в сфере интеллектуальной обработки данных применительно к реляционным СУБД.

Примеры задач, решаемых методами интеллектуального анализа:

- выявление преднамеренного искажения грузовых таможенных деклараций [2];
- выявление аномалий в работе бортовых систем космических аппаратов [3];
- прогноз развития раковых заболеваний на основе анализа состава крови [4];
- классификация двойных затменно-переменных звезд [5];
- распознавание и идентификация объектов на гиперспектральных снимках земной поверхности.

Предварительная обработка данных. Данные необходимо обработать таким образом, чтобы их было удобно располагать в специальной реляционной структуре хранения, а также существовала возможность непосредственной обработки алгоритмами DM.

Можно выделить следующие этапы подготовки данных.

Извлечение – на этом этапе решается задача извлечения данных из различных источников и организуется их грамотное хранение в специальной реляционной структуре.

Очистка – производится оценка качества данных и их преобразование к требуемой форме (проводится восстановление пропущенных значений, сокращаются незначащие и избыточные признаки).

Предобработка – выражается в дополнительном преобразовании данных, необходимом для решения конкретной задачи интеллектуальной обработки.

Группы задач DM. Все задачи интеллектуальной обработки информации можно условно разделить на несколько основных групп:

ассоциация – это поиск закономерностей между связанными объектами, заключающийся в определении элементов, которые часто появляются вместе, и выявлении по ним правил взаимосвязей;

кластеризация – это разбиение по набору атрибутов группировок вариантов данных (кластеров). Варианты внутри одной группы имеют схожие значения атрибутов;

классификация – это упорядочивание по некоторому принципу множества объектов, которые имеют сходные классификационные признаки, выбранные для определения сходства или различия между этими объектами;

регрессия – это поиск зависимости между исследуемыми переменными, которая может быть выражена в числовом значении;

прогнозирование – это попытка предсказания будущих событий. В качестве входных данных используется последовательность чисел, которая является рядом значений в течение некоторого времени. Затем определяются будущие значения для этих рядов;

анализ последовательностей – это поиск шаблонов в рядах дискретных состояний событий. Например, молекула ДНК может иметь состояния: A, C, B и D;

анализ отклонений – это поиск в данных значений, сильно отличающихся от нормированных.

Реляционная структура хранения данных. Для хранения большого объема данных, используемого для интеллектуальной обработки, применяют хранилища данных [6].

В реляционных СУБД для организации хранилищ данных наиболее простой является логическая структура объектов хранения (таблиц) типа «звезда». Структура «звезда» - это схема в базе данных, имеющая форму звезды. На рисунке 1 приведен пример схемы «звезда» для набора данных по измерительной информации бортовых систем космического аппарата (КА).

Центром схемы является таблица фактов «СОСТОЯНИЕ КА», отражающая состояние бортовых систем КА на фиксированный момент времени. Таблица фактов связана через внешние ключи с таблицами измерений фактов «ДАТЧИКИ ДАВЛЕНИЯ БОРТА» и «ДАТЧИКИ ТЕМПЕРАТУРЫ БОРТА», содержащими информацию по системам измерительных датчиков давления и температуры с борта КА, и таблицей «ДАТА/ВРЕМЯ ИЗМЕРЕНИЯ», хранящей соответствующий набор значений времени проведения измерений. Схема «звезда» позволяет эффективно создавать аналитические SQL-запросы и обрабатывать хранимую информацию.

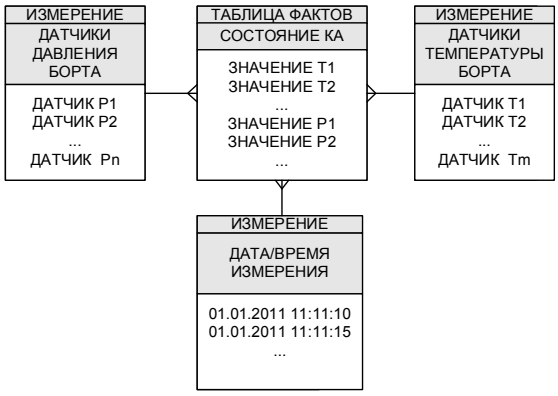


Рисунок 1 – Пример схемы «звезда» для набора данных по измерительной информации бортовых систем КА

Основные алгоритмы DM. Существующие алгоритмы можно объединить по признаку решаемых базовых задач DM в несколько групп.

Алгоритмы задач ассоциации:

алгоритм Apriori – в основе лежит решение задачи по поиску частого предметного набора, определения частоты появления этого набора в исследуемых данных и создания на основе полученных результатов правил ассоциации;

алгоритм многоуровневых правил – в основе лежит поиск правил ассоциации для наборов данных, имеющих дополнительную иерархию;

алгоритм последовательных шаблонов – в основе лежит решение задачи поиска шаблона, описывающего зависимости данных аналогично алгоритму Apriori, но с учетом временных последовательностей появления предметов анализа.

Алгоритмы задач кластеризации. Все алгоритмы кластеризации по методам обработки информации можно разделить на две большие группы: иерархические и итеративные.

При кластеризации иерархическими методами производится последовательное объединение меньших кластеров в большие или разделение больших кластеров на меньшие.

При кластеризации итеративными методами производится разделение исходной совокупности данных. Итеративный процесс деления на новые кластеры выполняется до тех пор, пока не будет выполнено правило остановки.

В иерархические методы кластеризации входят:

агломеративные алгоритмы (Agglomerative

Nesting, AGNES) - в начале работы алгоритма все объекты являются отдельными кластерами. На первом шаге наиболее похожие объекты объединяются в кластер. На последующих шагах объединение продолжается до тех пор, пока все объекты не будут составлять один кластер;

дивизимные алгоритмы (Divisive analysis, DIANA) – в начале работы алгоритма все объекты принадлежат одному кластеру, который на последующих шагах делится на меньшие кластеры, в результате образуется последовательность расщепляющих групп.

Пример работы алгоритмов иерархических методов кластеризации приведен на рисунке 2.

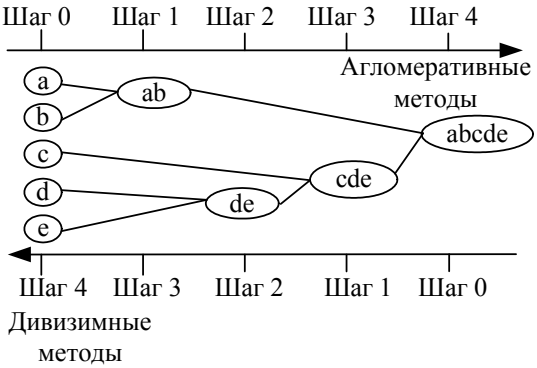


Рисунок 2 – Пример диаграммы работы алгоритмов иерархических методов кластеризации

При проведении кластерного анализа критерием для определения схожести и различия кластеров (мерой сходства) может являться расстояние между точками на диаграмме рассеивания. В случае анализа информации по двум измерениям (например, X и Y) меру расстояния между кластерами вычисляют как евклидово расстояние между двумя точками i и j на плоскости, когда известны координаты X и Y:

$$D_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \ . \qquad (1)$$

Евклидово расстояние является наиболее популярной мерой сходства. Существуют также такие меры сходства, как «Манхэттенское расстояние», расстояние Чебышева, метод ближнего соседа, метод наиболее удаленных соседей, метод Варда [1].

Среди итеративных методов классификации можно выделить следующие:

- *алгоритм k-средних (алгоритм быстрого кластерного анализа)* – для набора данных задается число k кластеров, затем все объекты набора данных разбиваются на отдельные кластеры, формируемые на максимальном удалении друг от друга;

- *нечеткая кластеризация* – методы кластеризации исследуемой информации на основе нечеткой логики (позволяют одному и тому же объекту принадлежать одновременно нескольким кластерам, но с различной степенью).

Алгоритмы задач классификации, прогнозирования и регрессии. По методам обработки информации можно выделить следующие основные группы алгоритмов:

- *деревья решений;*
- *байесовская классификация;*
- *нейронные сети;*
- *метод опорных векторов;*
- *статистические методы (линейная регрессия).*

Методы деревьев решений наиболее широко применяются для решения задач классификации и прогнозирования. Дерево решений – это способ представления классифицирующих правил типа «ЕСЛИ... ТО...» в иерархической структуре. Для принятия решения, к какому классу отнести некоторый объект или ситуацию, требуется ответить на вопросы, стоящие в узлах этого дерева, начиная с его корня.

Байесовская классификация включает в себя ряд алгоритмов, самым известным из которых является простой байесовский классификатор (Naive Bayes). Он основан на принципе нахождения максимума условной вероятности принадлежности исследуемого объекта к каждому классу. Выбирается тот класс, для которого условная вероятность максимальна.

Привлекательным представляется использование теории *искусственных нейронных сетей* (ИНС) ввиду их особых свойств [7] при решении задач DM. Анализ зарубежных публикаций показал, что этот вопрос изучается достаточно давно [8, 9, 10] и всесторонне [11].

В целом подход к применению искусственных нейронных сетей при решении задач DM заключается в использовании ИНС для кластеризации (как правило, на основе карт Кохонена и других ИНС на основе самоорганизации и конкуренции), а также для выработки решающих правил. Кроме того, практикуется использование генетических алгоритмов совместно с ИНС для решения задач DM [12].

Использование нечетких множеств и нечетких множеств второго типа [13, 14, 15] для решения задач DM изучено в значительно меньшей степени. Исследования в этом направлении представляют практический интерес, так как теория нечетких множеств второго типа показывает их весьма привлекательные свойства при решении интеллектуальных задач.

Метод опорных векторов (Support Vector Machine - SVM) – определяет классы данных с помощью границ областей.

В основе метода лежит понятие плоскостей решений. Плоскость решения разделяет объекты с разной классовой принадлежностью [1]. Цель метода – найти плоскость, разделяющую множества объектов. На рисунке 3 приведен пример двух множеств с объектами разного типа (А и В), через которые проведена разделяющая плоскость, и определены опорные векторы.

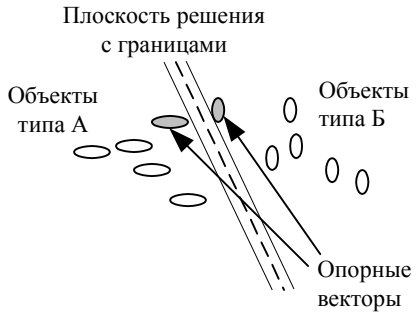


Рисунок 3 – Пример применения метода опорных векторов

Опорные векторы – это объекты множества, которые располагаются на границах плоскости решения.

Статистические методы – основаны на математической статистике. Среди них можно выделить алгоритмы линейной и логистической регрессии. Алгоритмы DM на основе линейной регрессии [16] решают задачу нахождения коэффициентов уравнения линейной регрессии, имеющего вид:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n \qquad (2)$$

где y – выходная (зависимая) переменная модели;

$x_1, x_2 \ldots x_n$ - входные (независимые переменные;

b_i - коэффициенты линейной регрессии.

Задача линейной регрессии заключается в подборе коэффициентов b_i уравнения (2) таким образом, чтобы на заданный входной вектор $X = (x_1, x_2 \ldots x_n)$ регрессионная модель формировала желаемое выходное значение y.

Часто алгоритмы DM на основе линейной регрессии используют для решения задач прогнозирования, когда наблюдения из прошлого являются входными переменными x_i модели, а y определяется как прогнозируемое значение.

Алгоритмы DM на основе логистической регрессии используют модификацию регрессии, которая решает задачу оценки вероятности того, что зависимая переменная примет значения в интервале от 0 до 1. Как правило, алгоритмы логистической регрессии применяются, когда выходной результат может быть бинарным (например, принимать значения «Да» или «Нет»).

Алгоритмы DM в современных СУБД. Фирмой Microsoft разработаны следующие алгоритмы DM для СУБД SQL Server[17]:

- Microsoft Naive Bayes (Байесовский классификатор);
- Microsoft Decision Trees (дерево решений);
- Microsoft Time Series (анализ временных рядов);
- Microsoft Clustering (кластеризация);
- Microsoft Sequence Clustering (регрессия и кластеризация);
- Microsoft Association Rules (ассоциация);
- Microsoft Neural Network and Microsoft logistic regression (нейронные сети и логистическая регрессия).

Фирмой Oracle разработаны следующие алгоритмы интеллектуальной обработки информации для СУБД Oracle [18]:

- Apriory (ассоциация);
- Decision Tree (дерево решений);
- Generalized Linear Model (линейная регрессия);
- k-Means (быстрый кластерный анализ k-средних);
- Minimum Description Length (алгоритм поиска существенных атрибутов в прогнозировании на основе регрессии);
- Naive Bayes (Байесовский классификатор);
- Non-Negative Matrix Factorization (итеративная классификация);
- O-Cluster (кластеризация);
- Support Vector Machines (метод опорных векторов).

Часть алгоритмов DM, реализованных в рассмотренных СУБД, имеют узкую коммерческую специализацию. Существующие программные продукты DM как в составе СУБД, так и сторонние, имеют высокую стоимость и содержат закрытый исходный код. В связи с широким применением свободно распространяемых СУБД с открытым исходным кодом (PostgreSQL, Firebird), не имеющих собственных эффективных средств DM, выявляется необходимость создания для них эффективных и доступных средств интеллектуальной обработки.

Пример решения задачи выявления аномалий в работе бортовых систем космических аппаратов с применением рассмотренных алгоритмов DM. Разработанная в NASA индуктивная система мониторинга (IMS - Inductive Monitoring System) решала задачу интеллектуального анализа потока телеметрической информации с борта космического корабля STS-107 Columbia [3]. Как известно, «Шаттл»

потерпел катастрофу из-за отрыва куска изоляционной обшивки, пробившей термоизоляцию на левом крыле. Отрыв произошел во время старта корабля. Однако о проблемах с термоизоляцией стало известно лишь через 17 дней, во время приземления «Шаттла». IMS выдала сигнал о возникновении неисправности в течении двух минут с момента ее возникновения.

Описание задачи. При эксплуатации «Шаттла» STS-107 Columbia обеспечивался процесс мониторинга работоспособности бортовых систем корабля в масштабе реального времени при использовании компьютерных моделей на основе входной телеметрической информации. Одной из задач являлся автоматический анализ состояния термоизоляции корпуса «Шаттла» для выявления случаев аномальных ситуаций.

Решение задачи. База данных IMS строилась на основе анализа данных предыдущих пяти полетов космического корабля. В IMS был использован алгоритм DM k-means (алгоритм быстрого кластерного анализа k-средних). IMS накапливала данные о нормальной работе системы температурных датчиков «Шаттла» и строила модель её поведения, представленную набором кластеров. Каждый кластер являлся вектором входных данных, содержащим информацию о значениях параметров группы температурных датчиков. Алгоритм DM выявлял аномалии в значениях расстояний между векторами данных. На момент повреждения обшивки корабля в базе данных IMS содержалось 490 кластеров данных по левому крылу «Шаттла» и 237 – по правому. На рисунке 4 представлен результат интеллектуального анализа телеметрии с борта «Шаттла» системой IMS на момент времени повреждения обшивки.

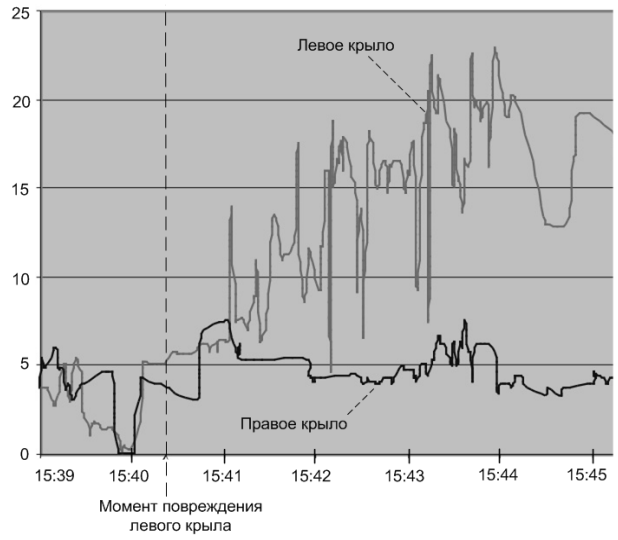


Рисунок 4 – Результат интеллектуального анализа телеметрии с борта «Шаттла» STS-107 Columbia

В горизонтальной оси указано время в минутах, включающее момент повреждения обшивки. В вертикальной оси представлено в % отклонение от номинальных значений в поведении модели системы температурных датчиков, определенное IMS как отклонение векторов данных (кластеров) от нормы (кластеров, полученных в период обучения при нормальной работе бортовых систем «Шаттла»). Вертикальной линией указан момент времени (15 часов 40 минут 22 секунды), когда произошло повреждение обшивки левого крыла. Как видно из рисунка, интеллектуальный анализ информации о состоянии системы термоизоляции «Шаттла» выявил аномальное поведение системы левого крыла уже через две минуты после возникновения нештатной ситуации.

Выводы. На основе рассмотренных примеров решения задач интеллектуальной обработки информации можно сделать вывод о высокой эффективности применения описанных алгоритмов DM. Сложные системы анализа данных, системы поддержки принятия решений должны строиться на их основе.

Библиографический список

1. *Чубукова И.А.* Data Mining. Основы информационных технологий. Специальные курсы. Издательство «Бином». Лаборатория знаний, 2006. - 384 с.

2. *Майоров А.А.* Применение технологий DM в решении задач таможенной службы, компания РДТЕХ //http://www.rdtex.ru/iso/root/main_page.html.

3. *D.L. Iverson*, Inductive System Health Monitoring, Proceedings of The 2004 International Conference on Artificial Intelligence (IC-AI'04), CSREA Press, Las Vegas, NV, 2004 //http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.5526&rep=rep1&type=pdf.

4. *Кузнецов В.А., Ившина А.В., Кузнецова А.В., Сенько О.В.* Анализ фенотипа лимфоцитов крови в прогнозировании метастазирования у больных остеосаркомой. Подход, основанный на распознавании нечетких систем // Иммунология. 1995. № 5. С.52-58.

5. *Калиниченко Л.А., Лебедев А.Е., Малков О.Ю.* Применение методов и средств извлечения знаний из данных в астрономии на примере классификации затменных звезд. М.: ИПИ РАН, 2007.

6. *Конноли Т., Бегг К., Страчан А.* Базы данных: проектирование, реализация и сопровождение. Теория и практика, 2-е изд.: Пер. с англ. — М.: Издательский дом «Вильямс», 2001. - 1120 с.

7. *А. Хайкин С.* Нейронные сети: полный курс. 2-е. – М.: Вильямс, 2006.- 1104 с.

8. *Craven M.W., Shawlik J.W.* Using Neural Networks for DM / Future Generation Computer Systems, 1997.

9. *C. Jason Ong, Syed Sibte Raza Abidi* DM Using Self-Organizing Kohonen maps: A. Technique for Effective Data Clustering & Visualisation. // International Conference on Artificial Intelligence (IC-AI'99), June 28-July 1 1999, Las Vegas.

10. *D. Krieger C.* Neural Networks in DM // www.cs.uml.edu/~ckrieger/user/Neural_netrworks.pdf.

11. *E. Basilio de Braganca Pereria, Calyampudi Radhakrishna Rao* DM using Neural Networks:A Guide for Statisticians – Editor: Textbook Revolution, 2009.

12. *F. Richard S. Segall, Qingyu Zhang* Applications of Neural Network and Genetic Algorithm DM Techniques in Bioinformatics Knowledge Discovery – A Preliminary Study // http://www.swdsi.org/swdsi06/ Proceedings06/Papers/KMS01.pdf.

13. *Заде Л.А.* Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.: Мир, 1976.-165 с.

14. *H. Mendel, J.M.,* Type-2 Fuzzy Sets and Systems and Overview // IEEE Computational Intelligence Magazine 2(1):20-29, Febriary 2007.

15. *Демидова Л.А., Пылькин А.Н.*. Методы и алгоритмы принятия решений в задачах многокритериального анализа. - М.: Горячая линия - Телеком, 2007.-232 с.

16. *Паклин Н., Орешков В.* Бизнес-аналитика: от данных к знаниям: учеб. Пособие. - СПб.: Питер, 2010.-704 с.

17. *Макленн Дж.,Танг Чж., Криват Б.*. Microsoft SQL Server 2008: DM – интеллектуальный анализ данных - СПб.: БХВ-Петербург, 2009.-720 с.

18. Oracle DM Concepts, 11g Release 1 (11.1) // http://download.oracle.com/docs/cd/B28359_01/ datamine.111/b28129.pdf.

УДК 681.3.048

В.В. Белов, В.И. Чистякова

ИНТЕРПРЕТАЦИОННАЯ ФОРМУЛА ДЛЯ ДВОИЧНОГО КОДА ВЕЩЕСТВЕННОГО ТИПА EXTENDED (LONG DOUBLE)

Излагаются результаты синтеза на основе экспериментальных исследований интерпретационной формулы для длинного двоичного кода десятибайтного вещественного типа (Extended, long double) данных. Получена интерпретационная формула для транслятора Turbo Pascal фирмы Borland. Приводимые сведения могут использоваться при реализации надёжных программных приложений, реализуемых с учётом специфики внутримашинного представления данных.

***Ключевые слова**: IEEE 754, представление вещественных чисел, машинный ноль, машинная эпсилон.*

Введение. Профессиональный программист и специалист-исследователь, интенсивно использующие программирование, должны иметь достаточно полные знания о внутримашинном представлении данных, для того чтобы, во-первых, избежать досадных ошибок и, во-вторых, способствовать максимизации эффективности создаваемого программного кода. Наблюдаемый в настоящее время ренессанс отечественного вычислительного программирования, обусловленный некоторым оживлением научных исследований, в первую очередь связанных с созданием нанотехнологий, предопределяет потребность в вычислениях с максимально возможной точностью. Это, в свою очередь, обусловливает применение в вычислительных программах вещественных типов максимальной длины – Extended и long double. В литературе и Интернет-источниках содержатся многочисленные сведения о структуре кодов и формулах их интерпретации для всех вещественных типов, кроме типа Extended. Показателен в этом плане список ссылок, приводимый практически во всех статьях Википедии, посвящённых стандарту IEEE 754.

Floating point precisions
IEEE 754:
16-bit: Half (binary16)
32-bit: Single (binary32), decimal32
64-bit: Double (binary64), decimal64
128-bit: Quadruple (binary128), decimal128
Other:
Minifloat Extended precision
Arbitrary-precision

Код десятибайтного вещественного типа попал в «мёртвую информационную зону».

Целью настоящей статьи является устранение указанной выше коллизии и получение интерпретационной формулы для типа Extended, что явится хотя и малой, но заметной составляющей знаний, способствующих созданию корректных программ, реализующих вычислитель-

ные алгоритмы в приложениях с повышенными требованиями к точности вычислений.

Прежде всего, отметим, что десятибайтный вещественный тип с идентификатором Extended реализован в языках программирования систем Turbo Pascal и Delphi. Этот же тип с названием long double реализован в языке программирования среды C++ Builder фирмы Borland International, Inc. Полезно знать, что тип с названием long double поддерживается и в языке MS Visual C++, но, к сожалению, только как синоним типа double, т.е. является обычным восьмибайтным вещественным типом, часто называемым типом удвоенной точности.

И ещё приносим извинения читателю с развитой математической культурой за использование термина «диапазон» (range), столь широко применяемого в англоязычной литературе по информатике, вместо математически строгих терминов «интервал» и «отрезок».

Имеющиеся описания десятибайтного вещественного типа крайне лаконичны. Например, в 1152-страничной книге [1] типу Extended посвящена одна строка в таблице и две строки в тексте на странице 867. Многочисленные учебники по Turbo Pascal, Delphi и Borland C++ информируют только о следующем: 1) диапазон модуля значений величин рассматриваемого типа составляет $3.6\cdot10^{-4932}..1.1\cdot10^{4932}$; 2) тип обеспечивает 19–20 десятичных значащих цифр; 3) величины типа Extended занимают 10 байтов (80 бит) памяти.

Заметим, что указанный выше диапазон значений десятибайтного вещественного типа свойственен среде Turbo Pascal, в Delphi же и Borland C++ диапазон значений составляет $3.6\cdot10^{-4951}..1.1\cdot10^{4932}$. Об этом свидетельствуют Интернет–источники [2], Help среды программирования и авторы, его читающие, например [3], [4].

Как уже отмечалось, в литературе крайне трудно найти описание формата двоичного кода десятибайтного вещественного типа при одновременном изобилии описаний типов Single и Double. Такое положение дел обусловлено содержанием базового документа [6]: он подробно определяет коды и интерпретационные формулы для типов Single и Double и содержит только самые общие сведения об Extended. Интересен факт: в [6] предлагался не один, а два варианта расширенных форматов двоичных кодов – Single Extended и Double Extended (см. таблицу 1).

Как видно из таблицы, параметры расширенных типов не определены, а обозначены. Конкретизацию параметров должны осуществлять компании, реализующие эти типы данных. Обстоятельства сложились так, что до сего дня для типа Single Extended реализаций не было предложено, а Double Extended был реализован фирмой Borland в проектах Turbo Pascal, Delphi и Borland C++ как типы Extended и long double.

Таблица 1 - Сводка параметров вещественных форматов данных IEEE 754 [6, стр. 7]

Параметр	Формат			
	Single	**Single Extended**	**Double**	**Double Extended**
Длина мантиссы (бит)	24	≥ 32	53	≥ 64
Максимальный порядок, E_{max}	+127	≥ +1023	+1023	≥ +16383
Минимальный порядок, E_{min}	–126	≤ –1022	–1022	≤ –16382
Смещение порядка	+127	не определено	+1023	не определено
Длина порядка (бит)	8	≥ 11	11	≥ 15
Длина формата (бит)	32	≥ 43	64	≥ 79

Базовый принцип построения кодов данных вещественных типов. Концептуально в основе кодов вещественных типов лежит научная форма представления (scientific notation) вещественных чисел, называемая также стандартной, показательной, инженерной и «с плавающей точкой»: $v = \pm\mu\cdot q^{p}$, где v – представляемое число; μ – мантисса числа; p – порядок числа; q – основание системы счисления.

Машинные двоичные коды вещественных чисел типов Single и Double [5, 6] имеют следующую специфику:

1) для представления отрицательных порядков в машинных кодах применяется смещение b (bias) порядка; в формулах, по которым вычисляется хранимое значение, порядок числа представляется в виде $p = e - b$, где e – характеристика (exponent) числа, которая физически хранится в коде и принимает только неотрицательные значения $0 \le e \le e_{max}$;

2) для унификации кодов – обеспечения одновариантности представления одного числа – осуществляется нормализация мантиссы; нормализованная мантисса удовлетворяет условию: $1 \le \mu < q$, для двоичной системы счисления это условие имеет вид: $1 \le \mu < 2$;

3) для уменьшения машинного нуля – непредставимой в коде окрестности нуля $-\delta < v < \delta$, определяемой ближайшим к нулю положительным числом δ, представимым в

машинном коде,– используются денормализованные (субнормальные) коды с *минимальным* порядком и мантиссой меньшей единицы;

4) старший разряд мантиссы физически не хранится, поскольку в двоичной системе счисления старший разряд нормализованной мантиссы всегда равен единице.

Согласно IEEE 754 двоичный код bCodeF(v) числа v типов Single и Double состоит из трёх полей: bCodeF(v) = $< s, e, f >$, где s – знаковый бит (sign bit) числа, e – характеристика (exponent) и f – дробная часть мантиссы (fraction). Все компоненты кода принято рассматривать как целые неотрицательные числа.

Код $< s, e, f >$ имеет два первичных параметра: L – количество бит, выделяемых под характеристику e; N – количество бит, выделяемых под дробную часть мантиссы f, и несколько вторичных (вычисляемых) параметров, важнейшие из которых: $e_{\max} = 2^L - 1$ – максимальное возможное значение характеристики (двоичный код $e_{\max}$ состоит из L единиц); $b = 2^{L-1} - 1$ – значение смещения порядка (двоичный код b состоит из $L-1$ единиц); $f_{\max} = 2^N - 1$ – максимальное возможное значение дробной части мантиссы (двоичный код $f_{\max}$ состоит из N единиц).

Интерпретация хранимого кода $< s, e, f >$ осуществляется по формуле [5, 6]:

$$v = \begin{cases} (-1)^s \cdot 1.f \cdot 2^{e-b}, \text{если } 1 \le e \le e_{\max} - 1 \; u \; f = *; \\ (-1)^s \cdot 0.f \cdot 2^{-b+1}, \text{если } e = 0 \; u \; f \ne 0; \\ (-1)^s \cdot 0, \text{если } e = 0 \; u \; f = 0; \\ (-1)^s \cdot Inf, \text{если } e = e_{\max} \; u \; f = 0; \\ NaN, \text{если } e = e_{\max} \; u \; f \ne 0, \end{cases} \quad (1)$$

где «*» – символизирует «любое возможное значение»; «$1.f$» – мантисса нормализованного кода числа; «$0.f$» – мантисса денормализованного кода числа; *Inf* – идентификатор неопределённости «бесконечность» (от английского *Infinity*); *NaN* – идентификатор неопределённости «не число» (от английского *Not a Number*), ему, в частности, соответствует результат деления 0/0.

Согласно интерпретационной формуле (1) можно выделить следующие пять групп кодов:

1) нормализованные коды – это внутримашинные представления «основных» значений, для которых справедливо утверждение: $1 \le e < e_{\max}$;

2) денормализованные коды малых чисел – уменьшителей пустого промежутка между нулём и наименьшим отличным от нуля числом, (машинным нулём δ), для которых справедливо утверждение: $(e = 0)$ & $(0 < f \le 2^N - 1)$;

3) код точного машинного нуля, для которого справедливо утверждение: $(e = 0)$ & $(f = 0)$; код обеспечивает два разных по форме, но одинаковых по математическому значению представления нуля: «0» и «–0»; эту возможность можно использовать в прикладных программах, в тех случаях, когда в формируемом результате вычислений целесообразно отразить, с какой стороны вычислительный процесс подошел к нулю;

4) код «Бесконечности» *Inf* со знаком, для которого справедливо утверждение:

$$(e = e_{\max}) \; \& \; (f = 0);$$

5) код «Не числа» *NaN*, для которого справедливо утверждение:

$$(e = e_{\max}) \; \& \; (0 < f \le 2^N - 1).$$

Заметим, что значения характеристики $e = 0$ и $e = e_{\max}$ используются для представления маргиналов: $e = 0$ – для представления нуля и денормализованных кодов малых чисел, а $e = e_{\max}$ – для идентификаторов неопределённостей «Бесконечность» и «Не число». Некоторые системы программирования не способны присваивать переменным значения неопределённостей и возбуждают исключение при попытке деления на ноль или вычисления «0/0». В таких случаях работа с неопределенностями может осуществляться только с помощью собственного кода с использованием операций приведения типа.

Интерпретационная формула (2) такова, что справедливо утверждение: в рамках типов Single и Double отсутствуют недопустимые коды, – не используемые для представления данных в вещественном типе; каждая комбинация элементов кода имеет вполне конкретную семантику.

Структура кода данных типа Extended. В отличие от типов Single и Double [5, 6] двоичный код bCodeE(v) вещественных данных в типе Extended (long double) имеет не три, а четыре поля [2, 7]:

$$\text{bCodeE}(v) = < s, e, i, f >,$$

где s, e, f – то же, что и в кодах типов Single и Double; i – бит целой части мантиссы. Во всех вещественных типах, кроме Extended, старшая цифра мантиссы физически не хранится, она подразумевается, т.е. является виртуальной. В нормализованных кодах эта цифра всегда равна единице, а в денормализованных равна нулю. В

Extended для старшей цифры мантиссы выделен специальный бит i. Все составляющие кода $<s,e,i,f>$ принято рассматривать как неотрицательные целые числа.

В реализации фирмы Borland характеристика e занимает $L=15$ бит, а под дробную часть f мантиссы выделяется $N=63$ бита.

Интерпретационная формула типа Extended в среде Turbo Pascal. Описание интерпретации хранимого кода $<s,e,i,f>$ в среде Turbo Pascal [7] имеет вид:

$$v=\begin{cases}(-1)^s\cdot 1.f\cdot 2^{e-b}, & \text{если } 0\le e<e_{\max};\\ (-1)^s\cdot Inf, & \text{если } e=e_{\max} \text{ и } f=0;\\ NaN, & \text{если } e=e_{\max} \text{ и } f<>0,\end{cases} \qquad (2)$$

где $e_{\max}=2^L-1=32767$ – максимальное значение характеристики (в двоичной системе счисления значение 32767 кодируется пятнадцатью единицами); $b=2^{L-1}-1=16383$ – смещение порядка (в двоичной системе счисления значение 16383 кодируется четырнадцатью единицами); $(e-b)$ – порядок числа; «$1.f$» – нормализованная мантисса числа с целой частью, равной единице, и дробной частью, равной f.

Заметим, что формула (2):

1) не определяет явно точный ноль; наименьшее (по модулю) хранимое значение согласно (2) равно $2^{-b}=2^{-16383}\approx 1{,}67\cdot 10^{-4932}$;

2) не предполагает хранение денормализованных (субнормальных) чисел, – мантисса которых меньше единицы;

3) не использует значение бита i.

Перечисленное означает, что формулу (2) можно квалифицировать как неполную.

Очень важно понимать, что термин «интерпретация кода вещественного числа» неоднозначен – *имеют место три интерпретации* этого кода:

1) во время работы процедур вывода и преобразования числа в строку (интерпретация конвертора);

2) при отображении значения переменной отладчиком (интерпретация отладчика);

3) при выполнении арифметических операций (интерпретация вычислителя).

Задача интерпретации кода компланарна задаче трансляции констант, присутствующих в исходном программном коде.

Исследуем код типа Extended, опираясь на интерпретационную формулу (1), определённую для типов Single и Double.

Точный ноль в Extended всё же существует. Его код легко получить, присвоив переменной x типа Extended ноль и осуществив операцию приведения типа Extended к типу «Массив байтов». Результат операции приведения – десять нулевых байтов. Таким образом, код точного нуля – это <0,0,0,0>.

Заметим, что имеет место следующий факт: Turbo Pascal не различает «0» и «–0» на программном уровне: операторы

x:=0; **x:=–0**;

порождают один и тот же код <0,0,0,0>. В то же время, если в участке памяти, занимаемом переменной **x**, искусственно синтезировать код <1,0,0,0>, то процедура **Write(x)** выведет на экран «–0», что полностью соответствует спецификации типов Single и Double.

Отладчик корректно отображает коды <0,0,0,0>. и <1,0,0,0> как значения «0» и «–0» соответственно.

Если умножить переменную с искусственно сформированным значением «–0» на любое другое неотрицательное число, то в результате будет получено значение «–0», корректно отображаемое отладчиком и процедурами вывода.

Итак, для всех трёх интерпретаций кода Extended (процедур конвертации кода в строку, отладчика и арифметических операций) *справедливо утверждение*:

$$((\text{bCodeE}(v)=<0,0,0,0>)\rightarrow(v=0))$$

&

$$((\text{bCodeE}(v)=<1,0,0,0>)\rightarrow(v=-0)).$$

Реакция отладчика среды Turbo Pascal на маргинальные значения такова: если $e=0$, то отладчик отображает ноль; если $e=e_{\max}=2^{15}-1=32767=0x111111111111111$, то отладчик отображает 0.0E32766, т.е. отображаемое отладчиком значение v таково, что *справедливо утверждение*:

$$((e=0)\rightarrow(v=0))\ \&\ ((e=e_{\max})\rightarrow(v=0.0\text{E}32766)).$$

Более чем странное значение 0.0E32766 отображается вместо идентификаторов неопределённостей.

Таким образом, отладчик не различает ноль и малые денормализованные значения. Данный факт можно квалифицировать как беду «малой руки», а вот то, что отладчик не знает идентификаторов *Nan* и *Inf* – это уже совсем плохо!

Денормализованные представления чисел в типе Extended используются вычислительными процедурами. Проведём несложные испытания. Сформируем код <0,0,0,1>, т.е. назначим следующие значения составляющим кода: $s=0$; $e=0$; $i=0$; $f=1$. Если среда допускает хранение денормализованных чисел, то этот код

соответствует значению $2^{-16445} \approx 3.6452 \cdot 10^{-4951}$ – наиболее близкому к нулю значению. Присвоим этот код переменной с именем x вещественного типа Extended и затем: 1) выведем x на экран дисплея; 2) просмотрим текущее значение переменной x в режиме отладки; 3) выполним оператор $y := x * 2 / x$, где y также переменная типа Extended; 4) умножим x на 2^{63}, а затем ещё на 2^{16382}, то есть всего на 2^{16445} (двухшаговое умножение необходимо по той причине, что максимальная степень двойки, представимая в Extended, равна 2^{16383}).

Наблюдаться будут следующие результаты: 1) на экране дисплея и в F-формате (обыкновенная десятичная дробь), и в E-формате (научный формат – мантисса с порядком) значение x будет отображено как ноль; 2) отладчик также покажет, что значение x равно нулю; 3) значение y окажется равным двум; 4) произведение $x \cdot 2^{16445}$ окажется равным единице.

Согласно первым двум результатам: $x = 0$. Третий результат свидетельствует, что $x \neq 0$, а четвёртый однозначно утверждает:

$$x = 2^{-16445} \approx 3.6452 \cdot 10^{-4951}.$$

Таким образом, процедуры конвертации кода Extended в строку толкуют денормализованные коды как ноль, в то же время вычислительные процедуры Turbo Pascal работают с денормализованными кодами вещественных чисел абсолютно корректно.

Трансляция вещественных констант в Turbo Pascal имеет следующую специфику: *константы*, требующие денормализованного представления, не поддерживаются на внешнем (пользовательском) уровне – они транслируются в точный ноль. Например, код

x:=1E–4933; Writeln(x);

выводит ноль, и память переменной **x** заполняется кодом <0,0,0,0>.

В то же время, поскольку вычислитель поддерживает денормализованные коды, то, присвоив переменной минимальное нормализованное значение 2^{-16382}, а затем поделив его на 2, получим первое денормализованное число 2^{-16383}. Коды и мантиссы указанных последовательных степеней двойки таковы:

$$\text{bCodeE}(2^{-16382}) = <0,1,1,0>,\ \mu = 1.0;$$

$$\text{bCodeE}(2^{-16383}) = <0,0,0,2^{62}>,\ \mu = 0.1_2.$$

Поделив ещё раз на 2, получим следующее, ещё меньшее денормализованное число 2^{-16384} с кодом $\text{bCodeE}(2^{-16384}) = <0,0,0,2^{61}>$ и мантиссой $\mu = 0.01_2$. Такое деление можно продолжать до указанного выше минимального $2^{-16445} = 2^{-16382-63}$, у которого $\text{bCodeE}(2^{-16445}) = <0,0,0,1>$, $\mu = 2^{-63}$.

Результаты экспериментального исследования типа Extended в среде Turbo Pascal. Экспериментально установлено, что интерпретация хранимого кода $<s,e,i,f>$ осуществляется по правилам, представленным в таблице 2.

Таблица 2 - Интерпретация Extended-кодов различными компонентами системы программирования

№ п/п	Элементы кода $<s,e,i,f>$	Процедуры вывода, конвертации в строку	Отладчик	Вычислительные процедуры
1	$(1 \le e < e_{\max})$ & $(i = 1)$ & $(f = *)$	$(-1)^s \cdot i.f \cdot 2^{e-b}$		
2	$(1 \le e < e_{\max})$ & $(i = 0)$ & $(f = *)$	$(-1)^s \cdot ?.? < 00...$	$(-1)^s \cdot i.f \cdot 2^{e-b}$	Error 207
3	$(e = 0)$ & $(i = *)$ & $(f = *)$	$(-1)^s \cdot 0\,0$		$(-1)^s \cdot i.f \cdot 2^{-b+1}$
4	$(e = e_{\max})$ & $(i = 1)$ & $(f = 0)$	$(-1)^s \cdot Inf$	0.0E32766	$(-1)^s \cdot Inf$
5	$(e = e_{\max})$ & $(i = 1)$ & $(0 < f < 2^{48})$	$(-1)^s \cdot Inf$	0.0E32766	Error 207
6	$(e = e_{\max})$ & $(i = 0)$ & $(f = *)$	NaN	0.0E32766	Error 207
7	$(e = e_{\max})$ & $(i = 1)$ & $(f \ge 2^{48})$	NaN	0.0E32766	Error 207

Строка 1 таблицы 2. Нормализованные коды в типе Extended интерпретируются по той же формуле, что и в типах Single и Double. Запись формулы слегка отличается, поскольку цифра целой части является не виртуальной, а реально хранимой в бите i единицей. Наименьшим по модулю числом в нормализованной форме является

$$1.0 \cdot 2^{-b+1} = 2^{-16382} \approx 3{,}3621 \cdot 10^{-4932}.$$

Наибольшим по модулю числом в нормализованной форме является

$$1.f_{\max} \cdot 2^{e_{\max}-1-b} = (2 - 2^{-63}) \cdot 2^{16383} \approx 1{,}18973 \cdot 10^{4932}.$$

Строка 2 таблицы 2. Недопустимые коды. В отличие от типов Single и Double тип Extended имеет подмножество недопустимых комбинаций своих составляющих s, e, i, f. Это подмножество определяется предикатом

$$(1 \le e < e_{\max}) \,\&\, (i = 0) \,\&\, (f = *),$$

и не предназначено для хранения информации. Его возникновение обусловлено физическим хранением целой части мантиссы. В типах Single и Double справедливо утверждение: $(1 \le e < e_{\max}) \to (1 \le \mu < 2)$. Справедливость этого утверждения легко обеспечивается, поскольку единица в мантиссе $\mu = 1.f$ формируется алгоритмически, а не извлекается из бита целой части. В Extended мантисса равна $\mu = i.f$, и в бит i можно поместить и единицу, и ноль независимо от значения характеристики e.

Недопустимые коды – это некорректно денормализованные коды, в которых мантисса меньше единицы при ненулевой (минимальной по арифметическому значению) характеристике.

Формально «недопустимые» коды не так уж и недопустимы. По математическому значению они могут рассматриваться как простые дубли разрешённых нормализованных и денормализованных кодов. Например, если дробная часть мантиссы содержит в начале k нулей, то денормализованное представление $0.f \cdot 2^{e-b}$ эквивалентно по вычисляемому значению нормализованному $1.f \cdot 2^{e-k-1-b}$.

Реакция различных компонентов среды Turbo Pascal на недопустимые коды различна.

Наиболее естественна реакция вычислительных процедур – они возбуждают исключение, свидетельствуя об ошибке кодирования данных «Error 207: Invalid floating point operation». Это формальная проверка элементов кода на корректность и реализация простого правила $(1 \le e < e_{\max}) \,\&\, (i = 0) \to Error$.

Процедуры вывода в разных форматах поразному отображают значение недопустимого кода. Очень затейлив вывод в E-формате – в начале появляются символы «?» и «<», совершенно не свойственные числам.

Очень оригинальна реакция на недопустимые коды отладчика – он игнорирует факт недопустимости кодов, у которых $(1 \le e < e_{\max}) \,\&\, (i = 0)$, вычисляет значение по формуле, установленной для нормализованных кодов, и отображает вполне корректные числа, однако вычислительные процедуры «капризничают» и не обрабатывают их, поскольку они не согласуются с используемыми схемами выравнивания порядков операндов.

Строка 3 таблицы 2. Ноль, критически нормализованные и денормализованные коды. Третья строка таблицы 2 представляет три группы кодов – код точного нуля, критически нормализованные и денормализованные коды. *Нулю* соответствуют коды $< s,0,0,0 >$, определяемые предикатом $(e = 0) \,\&\, (i = 0) \,\&\, (f = 0)$, они задают нулевую мантиссу «0.0», но дело не только в мантиссе (в кодах второй и шестой строк тоже могут быть нулевые мантиссы), – процедуры вывода и отладчик воспринимают код как ноль, по признаку $e = 0$.

Критически нормализованные коды – это подмножество кодов, определяемых предикатом $(e = 0) \,\&\, (i = 1) \,\&\, (f = *)$. Формально эти коды нормализованы, – они определяют значение мантиссы, не меньшее единицы. Однако равенство нулю характеристики выводит эти коды за рамки обычных нормализованных кодов. Процедуры вывода и отладчик интерпретируют их как нули, вычислительные процедуры работают с ними абсолютно корректно, рассматривая их как полноценный эквивалент обычных нормализованных кодов, у которых $e = 1$. Таким образом, экспериментально установлена справедливость следующего утверждения.

□***Утверждение 1***:

для вычислительных процедур Turbo Pascal коды $< s,1,1,f >$ и $< s,0,1,f >$ эквивалентны по арифметическим значениям и обрабатываются одинаково.■

Заметим, что при сложении числа, код которого имеет вид $< s,0,1,f >$, с нулём формируется эквивалентное по значению число, но уже с кодом $< s,1,1,f >$, который правильно воспринимают и процедуры вывода, и отладчик.

Денормализованные коды – это подмножество кодов, определяемых предикатом $(e = 0) \,\&\, (i = 0) \,\&\, (f = *)$. Компоненты Turbo Pascal интерпретируют их так же, как и критически нормализованные коды: процедуры вывода и отладчик интерпретируют их как нули, а вычислительные процедуры как полноценные корректные числа. Наименьшим по модулю, но отличным от нуля, числом в денормализованной форме является $2^{-16445} \approx 3.6452 \cdot 10^{-4951}$.

Машинный ноль и машинная эпсилон. Указанное выше ближайшее к нулю число в денормализованной форме является тем, что часто называют термином «***машинный ноль***» [8]. Для обозначения машинного нуля мы используем символ δ. Формула, по которой может быть вычислен машинный ноль для кодов, удовлетворяющих стандарту IEEE 754, имеет вид:

$$\delta = 2^{-b+1-N}, \qquad (3)$$

где b – смещение порядка (exponent bias); N – количество бит, выделяемых под дробную часть мантиссы f.

Как уже указывалось, машинный ноль определяет окрестность нуля, не представимую в машинных кодах. Это величина зазора между двумя ближайшими друг к другу последовательными числами 0 и δ, представимыми в кодах. И вообще, δ – это шаг, с которым на числовой оси располагаются вещественные числа, представимые в типах IEEE 754, принадлежащие полуоткрытому интервалу $[0;1)$. Для других интервалов шаг представимых чисел иной.

Величину зазора между двумя ближайшими друг к другу последовательными числами 1 и $1+\varepsilon$, представимыми в машинных кодах, принято называть машинной эпсилон (ε). Формула, по которой может быть вычислена машинная эпсилон для кодов, удовлетворяющих стандарту IEEE 754, имеет вид:

$$\varepsilon = 2^{-N}. \qquad (4)$$

Можно доказать, что справедливо следующее утверждение, определяющее множество точно представимых вещественных чисел.

□***Утверждение 2***:

1) представимые в типах IEEE 754 числа принадлежат отрезку $[2^{-b+1-N}; 2^{e_{\max}-1-b}]$;

2) числа, представляющие собой целые степени двойки 2^k, $-b+1-N \le k \le e_{\max}-1-b$, точно представимы в типах IEEE 754;

3) числа, принадлежащие интервалу $[0;1)$, точно представимые в типах IEEE 754, располагаются на числовой оси с шагом δ;

4) числа, принадлежащие полуоткрытому интервалу $[2^k; 2^{k+1})$, $0 \le k < e_{\max}-1-b$, точно представимые в типах IEEE 754, располагаются на числовой оси с шагом $\Delta_k = 2^k \cdot \varepsilon$.■

В связи с указанным утверждением укажем на досадную ошибку в материалах [5]: таблица «*Some example range and gap values for given exponents*» для типа Double содержит неверные значения шага (gap) значений, представимых в машинных кодах, из интервалов $[2;4)$ и $[4;8)$: во второй строке вместо

8.8817841970012523233890533447266e-16

должно быть

4.4408920985006261616945266E-0016;

в третьей строке вместо

3.5527136788005009293556213378906e-15

должно быть

8.8817841970012523233890533447266e-16.

Важным следствием указанного утверждения являются широко известные свойства ошибок представления чисел в формате «с плавающей точкой». Ошибочно кодируются числа, не представимые точно в соответствии с утверждением 2, т.е. находящиеся в промежутках между значениями, определяемыми шагами δ и Δ_k. Они округляются до ближайшего точно представимого значения. Максимальная ошибка, естественно, соответствует числу, находящемуся в середине пошаговых промежутков.

□***Утверждение 3***:

1) *абсолютная* погрешность $\mathbf{e}_{abs}(v)$ представления положительного числа v в типах IEEE 754 равна нулю в точках $v = 2^k \cdot (1 + n \cdot \varepsilon)$, где $k = \lfloor \log_2(v) \rfloor$ – целая часть двоичного логарифма числа v; $n = 0, 1, 2, \ldots, \frac{1}{\varepsilon} - 1$, т.е.

$$\mathbf{e}_{abs}\left(2^k \cdot (1 + n \cdot \varepsilon)\right) = 0;$$

2) *абсолютная* погрешность $\mathbf{e}_{abs}(v)$ имеет локальные максимумы $\mathbf{e}_{abs}^{[\max]}(v)$ в точках $v = 2^k \cdot (1 + n \cdot \varepsilon + \frac{\varepsilon}{2})$, где $n = 0, 1, 2, \ldots, \frac{1}{\varepsilon} - 1$; величина локальных максимумов нарастает по закону

$$\mathbf{e}_{abs}^{[\max]}\left(2^k \cdot (1 + n \cdot \varepsilon + \frac{\varepsilon}{2})\right) = \frac{\Delta_k}{2} = 2^{k-1} \cdot \varepsilon; \qquad (5)$$

3) *относительная* погрешность представления положительного числа v в типах IEEE 754 в точках экстремума абсолютной погрешности равна

$$\mathbf{e}_{rel}\left(2^k \cdot (1 + n \cdot \varepsilon + \frac{\varepsilon}{2})\right) = \frac{\varepsilon}{2 + 2 \cdot n \cdot \varepsilon + \varepsilon},$$

она максимальна при $n = 0$ – в середине первого шагового промежутка:

$$\mathbf{e}_{rel}^{[\max]}\left(2^k \cdot (1 + \frac{\varepsilon}{2})\right) = \frac{\varepsilon}{2 + \varepsilon} == \frac{\varepsilon}{2}, \qquad (6)$$

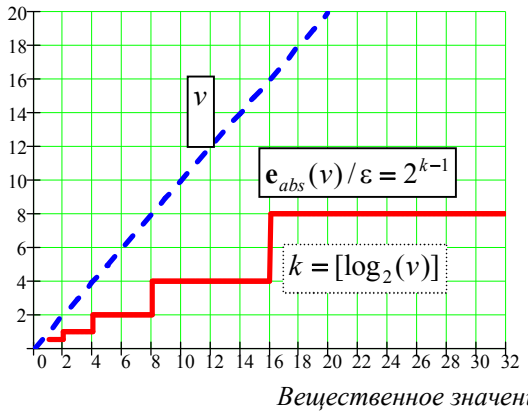
и минимальна при $n = \frac{1}{\varepsilon} - 1$ – в середине последнего шага:

$$\mathbf{e}_{rel}^{[\min]}\left(2^k \cdot (2 - \frac{\varepsilon}{2})\right) = \frac{\varepsilon}{4 - \varepsilon} == \frac{\varepsilon}{4}.■ \qquad (7)$$

В формулах (6) и (7) заимствованный из языка C знак операции отношения == означает следующий факт: результаты вычислений выражений левой и правой частей в типе Extended совпадают, поскольку за двойкой следующим представимым значением является $2 + 2 \cdot \varepsilon$, а четвёрке предшествует представимое $4 - 2 \cdot \varepsilon$. Числа же 2 и $2 + \varepsilon$, а также 4 и $4 - \varepsilon$ не различимы в типе Extended.

Формула (5) описывает ломано-линейный (ступенчатый) закон нарастания $\mathbf{e}_{abs}^{[\max]}(v)$ с рос-

том v, показанный на рисунке.



Максимальная ошибка представления чисел в типах IEEE 754

Строки 4 и 5 таблицы 2. Коды бесконечности. ***Процедуры вывода*** и конвертации числовых данных в строки интерпретируют как значение *Inf* коды, множество которых, согласно таблице 2, определяется предикатом:

$$(e = e_{\max}) \,\&\, (i = 1) \,\&\, (0 \le f < 2^{48}).$$

Этот факт находится в существенном противоречии с официальной формулой (2), которая не устанавливает связи *NaN* и разрядом i целой части мантиссы. В то же время нулевое значение i переводит код $< s, e_{\max}, i, 0 >$ из неопределённости $(-1)^s \cdot Inf$ в неопределённость *NaN*. Кроме этого, как $(-1)^s \cdot Inf$ интерпретируются коды $< s, e_{\max}, 1, f >$, у которых f не просто больше, а существенно больше нуля – вплоть до значения $2^{48} - 1$.

Отладчик Turbo Pascal, к сожалению, «не знает» о существовании кода бесконечности, и вместо него всегда выдаёт крайне неожиданное значение 0.0E32766, которое можно квалифицировать только как недоразумение.

Вычислительные процедуры Turbo Pascal правильно реагируют на коды неопределённостей $(-1)^s \cdot Inf$ только в случае $f = 0$. «Правильная» реакция выражается в том, что «значения» $(-1)^s \cdot Inf$ могут перемножаться, делиться и складываться с числами, синтаксически представленными величинами любого вида (константами, переменными, элементами структур, обращениями к функциям и выражениями). При этом умножение или деление на число с противоположным знаком меняет знак у *Inf*. Само же *Inf* как результат вычислений сохраняется, как и положено бесконечности.

Рекомендация относительно использования бесконечности в прикладных программах такова: представляйте бесконечность двумя комбинациями: $< s, e_{\max}, 1, 0 > \rightarrow (-1)^s \cdot Inf$, поскольку они правильно интерпретируются и процедурами вывода, и вычислительными процедурами. Коды $< s, e_{\max}, 1, f >$, у которых $1 \le f \le 2^{48} - 1$, рассматривайте как «лишние» и не используемые. (Всё лишнее, естественно, можно утилизировать в рамках собственной арифметики!)

Строки 6 и 7 таблицы 2. Коды «не чисел» в Turbo Pascal так же существенно дистанцируются от официальной формулы (2) – для получения *NaN* при $i = 1$ (это основное, концептуальное значение целой части мантиссы) вовсе не достаточно $f <> 0$, – требуется существенное превышение нуля, а именно $f \ge 2^{48}$. К сожалению, ни отладчик, ни вычислительные процедуры Turbo Pascal *NaN* не поддерживают.

Подлинная интерпретационная формула типа Extended в среде Turbo Pascal. Поскольку имеют место три интерпретации одной и той же комбинации кода, то интерпретационную формулу представим в виде:

$$\text{Predicat}(s,e,i,f) \rightarrow \begin{cases} \text{F}_{\text{IO}} = \text{F}_3(s,e,i,f); \\ \text{F}_{\text{Debugger}} = \text{F}_4(s,e,i,f); \\ \text{F}_{\text{Calculation}} = \text{F}_5(s,e,i,f), \end{cases} \qquad (8)$$

где $\text{Predicat}(s,e,i,f)$ – предикат, размещенный во втором столбце таблицы 2; IO, Debugger, Calculation – квалификаторы интерпретационных формул процедур вывода, отладчика и вычислительных процедур соответственно; $\text{F}_3(s,e,i,f)$, $\text{F}_4(s,e,i,f)$, $\text{F}_5(s,e,i,f)$ – интерпретационные формулы, размещенные соответственно в третьем, четвёртом и пятом столбцах таблицы 2.

Выводы

Научные результаты, представленные в настоящей статье:

1) методика экспериментального исследования способов интерпретации внутримашинных кодов чисел стандарта IEEE 754, основанная на искусственной генерации маргинальных кодов и анализе реакций на эти коды различных составляющих системы программирования – процедур преобразования числа в строку, отладчика и вычислительных процедур;

2) формулы для вычисления машинного нуля (3), машинной эпсилон (4), шага представимых значений по числовой оси (утверждение 2), максимальной абсолютной погрешности (5), максимальной (6) и минимальной (7) относительной погрешности представления чисел в кодах IEEE 754 *через параметры* этих кодов; в

литературе и Интернет-источниках вместо перечисленных формул представлены *константные* выражения для конкретных кодов и *алгоритм* для определения машинной эпсилон [8];

3) интерпретационная формула (8) кода данных типа Extended для системы программирования Turbo Pascal.

Прикладное значение полученных результатов состоит в том, что они позволят повысить правильность программных кодов, создаваемых для решения задач с превышенными требованиями к точности вычислений, помогут программистам избежать многочисленных недоразумений, возникающих при выполнении операций на уровне машинных нуля и эпсилон.

Можно утверждать, что изложенные в настоящей статье материалы войдут в арсенал средств специалистов, занятых практическим программированием и обучением программированию.

Библиографический список

1. *Архангельский А.Я.* Программирование в Delphi 7. – М.: ООО «Бином-Пресс», 2003 г. – 1152 с.: ил.

2. Extended precision // Wikipedia – The free Encyclopedia [Электронный ресурс] – Электрон. дан. – URL: http://en.wikipedia.org/wiki/Extended_precision.

3. *Хомоненко А.Д. и др*. Delphi 7 / под общ. ред. А.Д. Хомоненко. – СПб.:БХВ-Петербург, 2003 г. – 1216 с.: ил.

4. *Белов В.В., Чистякова В.И.* Программирование в Delphi: процедурное, объектно-ориентированное, визуальное: учебное пособие для вузов. – М.: Горячая линия–Телеком, 2009. – 240 с.

5. IEEE 754-1985 // Wikipedia – The free Encyclopedia [Электронный ресурс] – Электрон. дан. – URL: http://en.wikipedia.org/wiki/IEEE_754-1985.

6. IEEE Standard for Binary Floating-Point Arithmetic. Copyright 1985 by The Institute of Electrical and Electronics Engineers, Inc. 345 East 47th Street, New York, NY 10017, USA. [Электронный ресурс] – Электрон. дан. – URL: http://kfe.fjfi.cvut.cz/~klimo/nm/ieee754.pdf.

7. Turbo Pascal®. Version 7.0: Language Guide. BORLAND INTERNATIONAL INC. 1800 GREEN HILLS ROAD P.O. BOX 660001. SCOTTS VALLEY.CA 95067-0001. [Электронный ресурс] – Электрон. дан. – URL: http://www.bitsavers.org/pdf/borland/Turbo_Pascal_Version_7.0_Language_Guide_1992.pdf.

8. Машинный ноль // Википедия – свободная энциклопедия [Электронный ресурс] – Электрон. дан. – URL: http://ru.wikipedia.org/wiki/Машинный_ноль.

УДК 519.6

В.Е. Борзых, А.В. Морозова

СПОСОБ ОПРЕДЕЛЕНИЯ ПОГРЕШНОСТИ ЧИСЛЕННОГО ДИФФЕРЕНЦИРОВАНИЯ СИГНАЛОВ С ОГРАНИЧЕННЫМИ ЗНАЧЕНИЯМИ ПРОИЗВОДНЫХ

Предложен способ определения максимальной погрешности численного дифференцирования сигналов с ограниченными значениями производных. Сущность способа заключается в построении и анализе структурной схемы инверсной системы измерения погрешности. Приведены примеры расчета погрешностей, которые показывают преимущества предложенного способа по сравнению с классическими способами оценки погрешностей.

Ключевые слова: *погрешность, дифференцирование, сигнал, система.*

Введение. Задача численного дифференцирования состоит в приближенном вычислении производных функции по заданным в конечном числе точек значениям этой функции [1].

Пусть на интервале $[a,b]$ задана последовательность узлов $t_i = a + iT, i = 0,1...N,\ TN = b - a$ и для нее определены значения $y_i = y(t_i)$.

В качестве приближенного значения $y^{(1)}(t_i)$ можно взять, например, простейшее двухточечное разностное выражение

$$y_i^{(1)} = y^{(1)}(t_i) \approx (y_i - y_{i-1}) / T, \qquad (1)$$

где T – шаг дискретизации функции.

Возникающая при этом методическая погрешность дифференцирования характеризуется следующим соотношением

$$\left| y_i^{(1)} - (y_i - y_{i-1}) / T \right| \le (T/2) \max_{[t_{i-1}, t_i]} \left| y^{(2)}(t) \right|. \qquad (2)$$

Это выражение можно использовать в том случае, когда функция $y(t)$ на отрезке диф-

ференцирования будет иметь ограниченную по величине вторую производную. Однако при оценке погрешностей численного дифференцирования реальных сигналов возникают проблемы, связанные с возможностью использования соотношения вида (2). Например, при достижении входным сигналом некоторого электронного блока уровня ограничения выходной сигнал блока перестает изменяться и остается равным определенному порогу. В этот момент времени первая производная выходного сигнала скачком изменяется, а вторая производная сигнала обращается в бесконечность, поэтому для такого сигнала соотношение (2) не может быть использовано для расчета погрешности дифференцирования. Иногда операции дифференцирования может быть подвергнута кусочно-заданная функция. График этой функции обычно выглядит как гладкая кривая, но на самом деле состоит, например, из компонентов $y_i(t)$ и $y_{i+1}(t)$, сопрягающихся в точке t_i в соответствии со следующими условиями:

$$y_i(t_i) = y_{i+1}(t_i), y_i^{(1)}(t_i) \neq y_{i+1}^{(1)}(t_i).$$

Следовательно, и в этом случае нарушается непрерывность первой производной, поэтому ее вторая производная обращается в бесконечность.

Таким образом, при нарушении гладкости дифференцируемой функции использовать соотношение (2) для оценки погрешности численного дифференцирования нельзя.

Целью статьи является разработка способа расчета максимальной погрешности численного дифференцирования при нарушении гладкости дифференцируемой функции.

Постановка задачи. Разделим дифференцируемые функции по характеру нарушения гладкости на три множества: D1 - множество функций с ограниченной первой производной, D2 – множество функций с ограниченной второй производной и D12 – множество функций с ограниченной первой и второй производной. Для каждого множества функций требуется найти наихудший вариант реализации функции (принадлежащей этому множеству), при дифференцировании которой методическая погрешность, соответствующая двухточечному разностному соотношению (1), будет максимальна.

Способ решения задачи. Для генерирования реализаций функций, принадлежащих множествам D1, D2 и D12, будем использовать следующие формирователи сигналов (ФС):

- $ФС_1$ – интегратор сигнала $x(t)$, ограниченного по амплитуде (рисунок 1, а):

$$|x_1(t)| = |y^{(1)}(t)| \le M_1; \quad (3)$$

- $ФС_2$ – двойной интегратор сигнала $x(t)$, ограниченного по амплитуде (рисунок 1, б):

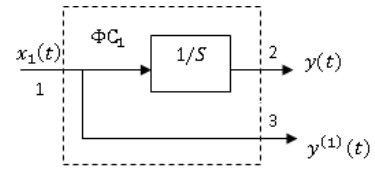
$$|x_2(t)| = |y^{(2)}(t)| \le M_2; \quad (4)$$

- $ФС_{12}$ – интегратор сигнала $x(t)$, ограниченного по амплитуде и по скорости изменения (рисунок 1, в):

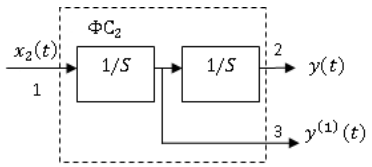
$$|x_{12}(t)| = |y^{(1)}(t)| \le M_1;$$
$$|x_{12}^{(1)}(t)| = |y^{(2)}(t)| \le M_2. \quad (5)$$

В этих выражениях M_1, $i = 1,2$ – максимально допустимое значение соответствующей производной дифференцируемого сигнала.

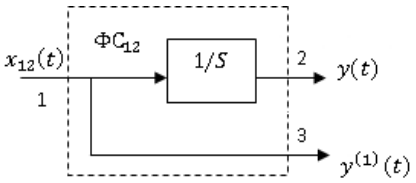
Каждый генератор имеет два выхода. На одном из выходов формируется сигнал, который будет подвергаться операции дифференцирования, а на другом – эталонный сигнал, соответствующий первой производной сигнала.



а



б



в

Рисунок 1 - Схемы формирования сигналов:
а – ограничение первой производной,
б – ограничение второй производной,
в – ограничение первой и второй производной

Схема формирователя сигнала $ФС_{12}$ похожа на схему $ФС_1$. Отличие заключается в том, что входной сигнал $x_{12}(t)$ должен быть ограничен не только по амплитуде, но и по скорости изменения (5). Такой сигнал может быть сформирован на выходе дополнительной цепочки элементов: первого ограничителя, интегратора и второго ограничителя сигнала. На рисунке 1, б эта цепочка, содержащая два нелинейных элемента, не показана. Такой прием позволяет

решить поставленную задачу, используя математический аппарат теории линейных импульсных систем.

Способ основан на применении метода инверсных систем [2] к анализу погрешностей численного дифференцирования сигналов, получаемых на выходе соответствующего формирователя.

Сущность способа заключается в построении и анализе структурных схем прямой и инверсной систем измерения погрешности численного дифференцирования.

В структурной схеме прямой системы измерения погрешности (рисунок 2) выходной сигнал $y(t)$ формирователя сигналов ФС (выход 2) подвергается следующей обработке. В соответствии с соотношением (1) реальное дифференцирующее устройство (РДУ) определяет приближенное значение первой производной $\tilde{y}^{(1)}(t)$ сигнала. Сигнал погрешности дифференцирования $\varepsilon^{(1)}(t)$ определяется в результате нахождения разности между сигналом $y^{(1)}(t)$, сформированным на выходе с номером 3 блока ФС, и сигналом $\tilde{y}^{(1)}(t)$ на выходе РДУ. Дискретные значения сигнала ошибки запоминаются в ступенчатом интерполяторе (СИ).

Требуется найти такой наихудший вид $\hat{x}(t)$ входного сигнала формирователя, для которого в момент наблюдения погрешность дифференцирования выходного сигнала достигает своего максимального значения.

Процесс определения максимальной погрешности сводится к построению соответствующей структурной схемы инверсной системы измерения погрешности, определению ее импульсной переходной функции $\bar{k}_\varepsilon(t)$ и вычислению максимальной погрешности

$$\varepsilon^{(1)}_{\max} = \int_0^\infty \hat{x}(t) k_\varepsilon(t) dt \ . \qquad (6)$$

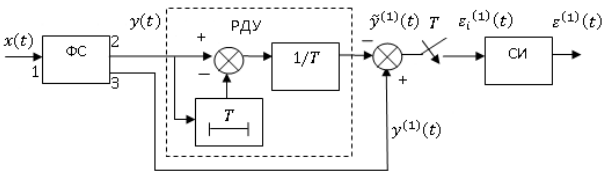


Рисунок 2 – Структурная схема прямой системы измерения погрешности численного дифференцирования

Расчет погрешностей. Рассмотрим три примера расчета погрешности (в соответствии с тремя типами ФС) при определении первой производной сигнала в соответствии с соотношением (1).

Пример 1. Дифференцируемый сигнал с ограниченной первой производной формируется на выходе устройства, схема которого показана на рисунке 1, а. Структурная схема соответствующей инверсной системы измерения погрешности приведена на рисунке 3.

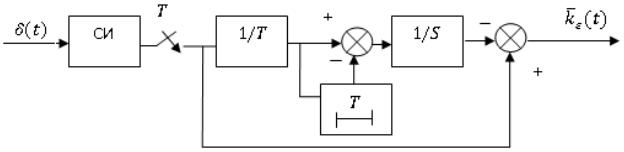
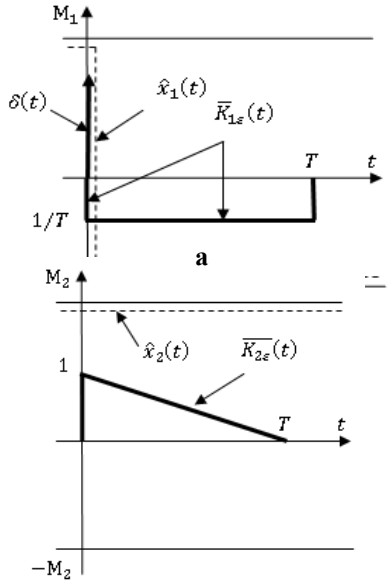


Рисунок 3 - Структурная схема инверсной системы измерения погрешности численного дифференцирования



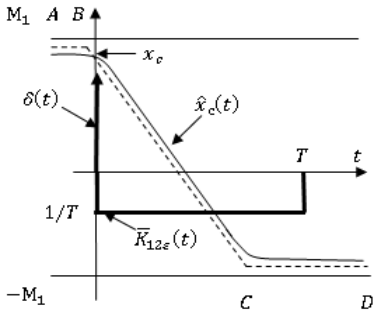


Рисунок 4 - Определение наихудших вариантов изменения сигналов на входах формирователей: а - $ФС_1$, б - $ФС_2$, в - $ФС_{12}$

Подав на вход этой схемы дельта-функцию и, проанализировав реакцию схемы на ее воздействие получим выражение импульсной переходной функции инверсной системы

$$\bar{k}_{1\varepsilon}(t) = \delta(t) - (1(t) - 1(t-T))/T \ . \qquad (7)$$

График этой функции (рисунок 4, а) состоит из дельта-функции и прямоугольника. Знак фун-

кции изменяется в точке $t=+0$, то есть после окончания дельта-функции.

Задача состоит в подборе наихудшего входного сигнала $\hat{x}_1(t)$, максимизирующего функционал (6) при известной функции $\bar{k}_{1\varepsilon}(t)$ вида (7).

Из ограничения (3) следует, что $\hat{x}_1(t)$ может быть разрывной функцией, поэтому наихудший вид сигнала на входе $ФС_1$, максимизирующего функционал (6), соответствует релейному сигналу с одним переключением (на рисунке 4, а он показан пунктирной линией). Учитывая это, упростим выражение (6)

$$\varepsilon_{1\max}^{(1)} = M_1 \int_0^{\infty} \left|\bar{k}_{1\varepsilon}(t)\right| dt . \qquad (8)$$

Подставив (7) в (8) и выполнив интегрирование, получим

$$\varepsilon_{1\max}^{(1)} = 2M_1 . \qquad (9)$$

Такая погрешность может появиться, например, при дифференцировании сигнала кусочно-линейного вида: в течение интервала времени $[t_{i-1},(t_i-0)]$ значения сигнала увеличиваются с максимальной скоростью M_1, а в точке (t_i-0) начинают уменьшаться со скоростью $-M_1$.

Пример 2. Дифференцируемый сигнал с ограниченной второй производной генерируется формирователем, схема которого показана на рисунке 1, б. В этом случае структурная схема инверсной системы измерения погрешности получается из схемы, изображенной на рисунке 3, путем включения на ее выходе второго интегратора. Выражение импульсной переходной функции инверсной системы определяется в результате интегрирования (7)

$$\bar{k}_{2\varepsilon}(t) = \begin{cases} (1-t/T), \text{для } 0 \le t \le T, \\ 0, \text{ для других значений } t. \end{cases} \qquad (10)$$

Эта функция на интервале $[0,T]$ знак не меняет (рисунок 4, б), следовательно, из всего множества сигналов, удовлетворяющих ограничению (4), наихудшим является сигнал $\hat{x}_2(t)$ с постоянной амплитудой M_2. Таким образом, подставив (10) в (6) и выполнив интегрирование, получим

$$\varepsilon_{2\max}^{(1)} = M_2 T/2 . \qquad (11)$$

Этот результат совпадает с известной оценкой погрешности (2). Такая величина погрешности достигается на сигнале параболической формы, у которого вторая производная на интервале имеет постоянное значение на уровне M_2 или $-M_2$.

Пример 3. Во многих задачах реальные сигналы имеют ограниченные величины первой и второй производных. Схема $ФС_{12}$ изображена на рисунке 1, в. Выражения для ограничений на первую и вторую производные приведены в (5). Структурная схема инверсной системы измерения погрешности соответствует схеме, показанной на рисунке 3, а ее импульсная переходная функция соответствует выражению (7).

В этом случае функция $\bar{k}_{12\varepsilon}(t)$ (рисунок 4, в) меняет знак в момент $t=+0$, поэтому нужно искать наихудший сигнал $\hat{x}_{12}(t)$ на интервалах $[0,+0],[+0,T]$ с учетом следующих условий сопряжения для непрерывной функции $x_{12}(t)$:

$$x_{12}(0) = x_{12}(+0) = x_c = \beta M_1 ,$$

где $0 \le \beta \le 1$.

Среди всех сигналов, проходящих в момент $t=+0$ через точку сопряжения x_c, наихудший сигнал $\hat{x}_c(t)$ должен максимизировать функционал

$$\varepsilon_{1\max}^{(1)} = \int_0^{+0} \delta(t) x_{12}(t) dt - (1/T) \int_{+0}^{T} x_{12}(t) dt . \qquad (12)$$

Максимум функционала достигается на сигнале $\hat{x}_c(t)$, имеющем вид ломаной линии ABCD (рисунок 4, в), пересекающей ось ординат в точке x_c с максимально допустимой скоростью $-M_2$. Действительно, другая кривая будет либо противоречить условиям (5), либо даст меньшее значение функционала (12).

Введем обозначение $k = M_1/M_2 T$. Рассмотрим два варианта продолжения решения задачи, каждый из которых зависит от диапазона изменения k.

Вариант 1. Пусть $0 \le k \le 1/2$. В этом случае при изменении величины β в пределах от 0 до 1 абсцисса точки C процесса $ABCD$ не выходит за границы интервала $[0,T]$, поэтому выражение для абсолютной погрешности имеет вид:

$$\varepsilon_{\max\limits_x}^{(1)}(\beta,k) = M_1\left(-\frac{\beta^2 k}{2} + \beta(1-k) - \frac{k}{2} + 1\right), \qquad (13)$$

где $0 \le \beta \le 1, 0 \le k \le 1/2$.

Проанализируем, как изменится погрешность дифференцирования при наложении ограничения на первую и вторую производные сигнала по сравнению со случаем, когда ограничение наложено только на первую производную.

Разделив левую и правую части выражения (13) на $2M_1$ [это максимальное значение абсолютной погрешности, определенное соотношением (9)], получим выражение относительной погрешности. Относительная погрешность достигает максимума при $\beta = 1$, поэтому

$$P_{\max\limits_{x,\beta}} = 1 - k, \text{ где } 0 \le k \le 1/2. \qquad (14)$$

Вариант 2. Пусть $1/2 \le k \le \infty$. В этом варианте относительная погрешность дифференцирования определяется по двум разным исходным формулам.

Первая формула [она соответствует выражению (12)] используется при изменении параметра β в пределах от 0 до β_1, где $\beta_1 = \frac{1}{k} - 1$, то есть рассматривается случай, когда абсцисса точки C ломаной линии $ABCD$ может достичь правой границы интервала $[0, T]$. Максимум погрешности определяется подстановкой в (12) значения $\beta_1 = \beta$, то есть

$$P_{\max\limits_{x,\beta}} = 1/4k \text{, где } 1/2 \le k \le \infty. \qquad (15)$$

Вторая формула вычисления погрешности используется при изменении параметра β в пределах от β_1 до 1, то есть рассматривается случай, когда абсцисса точки C ломаной линии $ABCD$ выходит за пределы правой границы интервала $[0, T]$. Исследования показывают, что величина относительной погрешности в этом случае не зависит от параметра β и определяется по формуле (15).

Объединив результаты (14) и(15), получим

$$P_{12\max} = \begin{cases} (1-k), \textit{для } 0 \le k \le 1/2; \\ 1/4k, \text{ для } 1/2 \le k \le \infty. \end{cases} \qquad (16)$$

Таким образом, по известным значениям M_1 и M_2 можно определить значение k, а затем вычислить относительную погрешность. Можно решить и обратную задачу: по заданной погрешности найти k, а затем определить допустимые значения M_1 и M_2.

Результаты моделирования. Целью экспериментальных исследований является подтверждение наиболее общих теоретических результатов, полученных в примере 3. Схема MATLAB - модели приведена на рисунке 5.

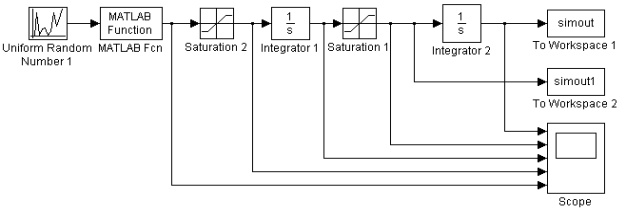


Рисунок 5 – Схема моделирования

Блоки Uniform Random Number 1 и MATLAB Fcn выполняют роль источника сигнала, амплитуда которого является равномерно распределенной случайной величиной. С помощью блоков Saturation 1, Saturation 2 и Integrator 1 моделируется формирователь сигналов ($ФС_{12}$) с ограничителями первой и второй производной. Значения выходного сигнала блока Integrator 2 сохраняются в рабочей области MATLAB (массив simout). Эти значения используются для приближенного вычисления первой производной. Точные значения первой производной запоминаются в массиве simout 1.

Параметры моделирования: время моделирования изменяется от 0 до 10, шаг моделирования 0.01, шаг дискретизации дифференцируемой функции $T = 1$. После окончания выполнения всех шагов моделирования запускается на выполнение MATLAB - функция, текст которой приведен ниже.

```
for k=101:1:1001
  sout(k,1)=simout(k,1)-simout(k-100,1);
  sout1(k,1)=abs(sout(k,1)-simout1(k,1));
end
   p=max(sout1(:,1))/8
```

Функция вычисляет приближенные значения первой производной (массив sout), сравнивает их с точными значениями (массив sout1) и определяет максимальное значение (из 901 результата) относительной погрешности (переменная p).

В таблице 1 приведены теоретические $P_{12\max}$ и экспериментальные результаты определения погрешностей численного дифференцирования реализаций случайных сигналов с различными значениями параметров M_1 и M_2. Анализ данных таблицы 1 показывает, что максимальное значение относительной погрешности, полученной экспериментально, достигает теоретического значения, но никогда не превышает его. Отдельные факты несовпадения теоретических и экспериментальных результатов свидетельствуют не об ошибках предложенной методики расчета, а соответствуют тому, что в реали-

зациях случайных сигналов отсутствовали аномальные участки, на которых относительная погрешность должна максимизироваться.

Таблица 1

M_1	M_2	k	$P_{12\max}$	Экспериментальные значения погрешности				
				Номер запуска программы на выполнение				
				1	2	3	4	5
4	16	0.25	0.750	0.7500	0.7456	0.7485	0.7500	0.7289
4	8	0.5	0.500	0.4944	0.3750	0.5000	0.4803	0.5000
4	4	1	0.2500	0.2500	0.2164	0.2500	0.2500	0.2500
4	2	2	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250
4	1	4	0.0625	0.0625	0.0625	0.0625	0.0625	0.0625
2	8	0.25	0.7500	0.7500	0.7492	0.7500	0.5368	0.7468
2	4	0.5	0.5000	0.4964	0.5000	0.5000	0.3750	0.5000
2	2	1	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500

Заключение. Предложенный способ имеет более широкие возможности для расчета погрешностей численного дифференцирования сигналов. Он позволяет получить не только известные ранее расчетные соотношения (как частные случаи), но и новые.

Например, если $k = 0$, то абсолютная погрешность, рассчитанная в соответствии с выражением $\varepsilon_{\max} = (1-k)2M_1$, будет равна $2M_1$. Этот результат соответствует известной расчетной формуле (9). Такой вариант оценки погрешности можно рекомендовать, если известно, что на некоторых участках гладкость сигнала существенно нарушается вследствие скачкообразного изменения первой производной.

Рассмотрим случай, когда $k \geq 1/2$. Подставив $k = M_1 / M_2 T$ во вторую строку выражения (16) и умножив результат на $2M_1$, получим $\varepsilon_{\max} = M_2 T / 2$. Получившийся результат совпадает с известной оценкой погрешности (11). Этот вариант расчета погрешности необходимо использовать при дифференцировании гладкого сигнала (его вторая производная в пределах шага дискретизации постоянна).

Если $0 \leq k \leq 1/2$, то погрешность

$$\varepsilon_{\max} = (1-k)2M_1 = (1 - M_1 / M_2 T)2M_1 .$$

Это новый результат. Такая погрешность появляется при дифференцировании сигналов, первая производная которых на отрезке времени Δt $(0 < \Delta t < T)$ переходит по линейному закону с уровня $+M_1$ на уровень $-M_1$.

Библиографический список

1. *Демидович Б.П., Марон И.А.* Основы вычислительной математики: учебное пособие. 6-е изд., стер.– СПб.: Издательство «Лань», 2007.– 672 с.

2. *Борзых В.Е.* К определению погрешности численного дифференцирования выходных сигналов динамических систем // Вестник РГРТУ. №2 (выпуск 28). Рязань, 2009. – С. 37-41.

УДК 81'322, 004.588

А.В. Пруцков, А.К. Розанов

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ МЕТОДОВ ОБРАБОТКИ ФОРМ СЛОВ И ЧИСЛИТЕЛЬНЫХ

Рассматриваются принципы работы программных систем, разработанных на основе предложенных методов генерации и определения форм слов и обработки числительных. Системы проверки знаний по морфологии, правилам образования количественных числительных и функционированию алгоритмических моделей позволяют автоматизировать данный процесс. Система сравнения статей трудовых коллективных договоров позволяет ускорить анализ данных документов и повысить его качество.

Ключевые слова*: автоматическая обработка текста, автоматизированные обучающие системы, анализ трудовых договоров.*

Введение. Автоматизация обучения с помощью программных средств удешевляет и упрощает процесс обучения, делает его более доступным для потребителей образовательных услуг. Автором статьи разработаны методы обработки форм слов и перевода числительных,

которые могут послужить основой для построения систем проверки знаний. Дадим краткую характеристику предложенным методам.

Метод генерации и определения форм слов включает модель формообразования и алгоритмы генерации и определения форм слов [1]. Генерация формы слова – процесс получения формы с использованием в качестве начальных параметров основы и грамматического значения. Определение формы слова заключается в нахождении по данной словоформе ее нормальной формы (основы) и грамматического значения. Модель формообразования предполагает, что получение любой формы слова любого естественного языка можно представить последовательностью простейших преобразований двух типов:

1) добавление подстроки S к строке слева (префикс) (обозначается S+) или справа (постфикс) (+S) без изменения самой строки;

2) замена в строке первого слева вхождения подстроки S на подстроку P (S → P).

Последовательность преобразований называется цепочкой преобразований.

На основе данной модели разработаны алгоритмы генерации и определения форм слов. Алгоритм генерации заключается в применении цепочки преобразований к основе и получении формы с заданным грамматическим значением. Алгоритм определения заключается в переборе цепочек преобразований и нахождении такой цепочки, которая позволит получить из формы основу слова.

Данный метод используется на морфологическом уровне автоматической обработки текстов, включающей также синтаксический и семантический уровни.

Метод обработки количественных числительных позволяет преобразовывать числительные в числа и обратно, переводить числительные на другие языки, определять падеж числительного и выявлять ошибки в числительных [2]. В основе этого метода лежит трехуровневая обобщенная модель числительного, которая является промежуточным этапом в перечисленных операциях. Алгоритмы этих преобразований записаны с помощью нормальных алгоритмов Маркова. Данный метод также используется на морфологическом уровне.

Показано, что метод генерации и определения форм слов и метод обработки количественных числительных применимы к русскому, английскому, немецкому, испанскому и финскому языкам.

Целью работы является разработка систем проверки знаний на основе методов обработки форм слов и числительных, а также системы сравнения статей трудовых договоров на основе этих же методов.

Системы проверки знаний. Процесс обучения можно подразделить на собственно обучение и проверку знаний. Использование автоматизированных обучающих систем (АОС) позволяет повысить эффективность этих этапов обучения. Предлагаются системы проверки знаний по морфологии, правилам образования количественных числительных и функционированию алгоритмических моделей, в основе которых лежат предложенные методы. Все они работают по одному принципу (см. рисунок) [3]. Процесс проверки знаний начинается с запроса системы проверки знаний к системе (1), в основе которой лежит один из предложенных методов. В ответ на запрос система проверки знаний получает задание обучаемому и ответ к нему (2). Система проверки знаний выдает полученное задание обучаемому (3), сравнивает ответ обучаемого с правильным ответом (4) и выдает результат: ответ правильный или ответ неправильный (5).

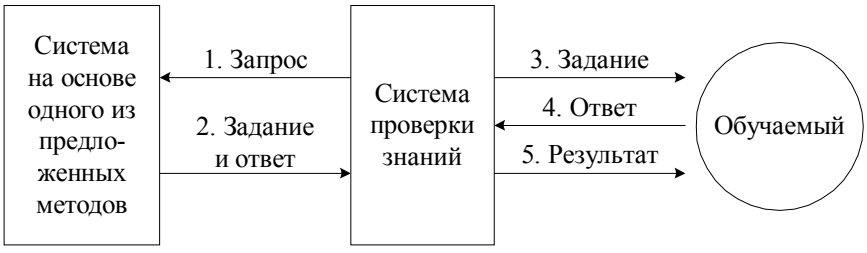


Схема работы системы проверки знаний

Формулировки заданий подготавливаются заранее преподавателем. Однако часть задания генерируется с помощью датчика случайных чисел, что увеличивает количество вариантов заданий и гарантирует, что каждый обучаемый получит уникальный вариант задания. Такая генерация задания называется динамической. Если задание не изменяется в процесс проверки знаний и заранее записано преподавателем, то такая генерация задания называется статической. Очевидно, что динамическая генерация позволяет получить большое число однотипных заданий, но требует разработки более сложного программного обеспечения по сравнению со

статической генерацией вопросов.

Этап обучения этим темам может быть реализован в виде электронных учебников.

Система проверки знаний по морфологии. Метод генерации и определения форм слов используется в системе проверки знаний по морфологии естественных языков для генерации форм слов, используемых в задании и ответе, а также для определения формы слова из ответа обучаемого, если ответ неверный. Примеры формулировок заданий:

1) запишите грамматическое значение глагола «*говорил*»;

2) запишите форму дат. п., мн. ч. слова «*дом*».

Параметры слов, такие как тип формообразования и грамматическое значение, определяются преподавателем в задании. С помощью метода генерации и определения генерируется форма слова для вопроса (задание 1) или для ответа (задание 2). Сгенерированный ответ сравнивается с ответом обучаемого. Если ответ неправильный, то с помощью метода генерации и определения система проверки знаний определяет, какая форма была указана в ответе обучаемого, и выдает поясняющий текст. Таким образом, система не только проверяет знания обучаемого, но и объясняет его неправильные ответы.

Система проверки знаний по морфологии и ее составные части зарегистрированы в РОСПАТЕНТе (свидетельства № 2010615326, № 2011611621, № 2011612830).

Система проверки знаний правил образования количественных числительных использует метод обработки числительных для генерации заданий и ответов к ним. Преподаватель задает диапазон чисел, используемый в задании. Из этого диапазона с помощью датчика случайных чисел выбирается число, которое затем будет использовано в задании. Примеры формулировок заданий:

1) какое число записано по-английски «*three hundred twenty two*»;

2) переведите числительное «*zweihundertdreiundzwanzig*» на испанский язык;

3) запишите число 123 по-фински.

Например, в задании 2 система обработки числительных получает в качестве входных данных число и преобразует его в числительные немецкого и испанского языков. Затем немецкое числительное подставляется в формулировку задания, а испанское числительное сравнивается с ответом обучаемого.

Данная информационная система зарегистрирована в РОСПАТЕНТе (свидетельство № 2011615475).

Система проверки знаний функционирования алгоритмических моделей. В данной системе задание состоит из программы одной из алгоритмических моделей и начального слова. Задачей обучаемого является определение результата работы алгоритма.

Для генерации задания необходимо выбрать случайное слово из заданного диапазона. Пусть диапазон задан словами A=«*ab*» и B=«*bab*» из алфавита, состоящего из n=2 букв *a* и *b*.

Рассмотрим алгоритм решения данной задачи.

1. Заменим буквы цифровыми эквивалентами: a – 1; b – 2; c – 3 и т. д. Тогда слово A будет представлено числом 12, а слово B – числом 212.

2. Считая эти числа числами двухцифровой системы счисления, переведем их в десятичную систему счисления по известному правилу перевода:

$$12 = 1\cdot 2^1 + 2\cdot 2^0 = 4_{10};$$

$$212 = 2\cdot 2^2 + 1\cdot 2^1 + 2\cdot 2^0 = 12_{10}.$$

3. Выберем случайное число из промежутка от 4_{10} до 12_{10}. Пусть это будет число 8_{10}.

4. Переведем число в двухцифровую систему счисления. Для этого используем правило перевода числа из десятичной системы счисления в двоичную систему счисления с одним отличием. Если остаток от деления числа равен нулю, то принять остаток равным n, а из результата деления вычесть 1 [4]. Переведем число 8_{10} в двухцифровую систему счисления:

8:2=3 (остаток 2); [до коррекции остатка 8:2=4 (остаток 0)]

3:2=1 (остаток 2);

1:2=0 (остаток 1).

В результате получим число 122 в двухцифровой системе счисления.

5. Заменим цифры буквами по тому же принципу, что и на шаге 1, и получим слово «*abb*», которое лежит диапазоне от A=«*ab*» до B=«*bab*». При этом для сравнения слов используются стандартные правила.

В результате получено случайное слово из заданного диапазона.

Система проверки знаний позволяет для любого начального слова и схемы нормального алгоритма определить результат работы алгоритма. Данный результат сравнивается с ответом обучаемого. В случае неправильного ответа система проверки знаний выдает обучаемому пошаговое выполнение алгоритма, чтобы пояснить принцип работы алгоритма.

Динамическая генерация вопросов в системах проверки знаний используется во многих

областях знаний (например [5]). Однако в морфологии авторам статьи известна система со статической генерацией заданий (см. [6]), а по теории алгоритмов такие системы неизвестны вообще.

Предложенные системы проверки знаний предназначены для самоконтроля или промежуточного контроля знаний. Также результаты проверки знаний могут использоваться при выставлении оценки при итоговом контроле. Был проведен эксперимент, в рамках которого одна группа студентов использовала для подготовки к экзамену систему проверки знаний функционирования алгоритмических моделей для подготовки к экзамену, а другая группа студентов – нет. Результаты проверки знаний студентов первой группы были выше на 7 %, чем результаты студентов второй группы. Очевидно, что этот показатель будет различаться для разных групп студентов.

Данная система проверки знаний зарегистрирована в РОСПАТЕНТе (свидетельство № 2011615476).

Предполагается интеграция предложенных систем проверки знаний с существующей в Рязанском государственном радиотехническом университете (РГРТУ) системой тестового контроля [7].

Система сравнения статей коллективных трудовых договоров. При принятии нового коллективного трудового договора необходимо определить, улучшает он права трудящихся по сравнению с предыдущей редакцией данного договора или нет. Для этого эксперт должен найти все статьи договора, которые были изменены, и выставить им экспертные оценки. Работу эксперта может упростить система анализа коллективно-договорных актов, разрабатываемая в РГРТУ. Одной из ее подсистем является система сравнения статей коллективных трудовых договоров. Сравнение статей должно ответить на вопрос: есть ли изменения в новой статье и, если статья обновилась, то в чем заключаются различия. В случае обновления статьи ее текст заново анализируется экспертом.

Для сравнения статей можно провести их семантический анализ. Однако «поиски универсальных путей автоматического анализа семантического содержания текстов пока не увенчались успехом» [8]. Поэтому задача сравнения статей трудовых договоров будет решаться на морфологическом уровне.

Предлагается сравнивать статьи по трем параметрам, позволяющим определить степень различия сравниваемых статей.

1. Количество одинаковых словоформ в сравниваемых статьях. Чем больше значение данного параметра относительно максимального количества слов, тем меньше различие между статьями.

2. Количество одинаковых слов (лексем) в сравниваемых статьях. Например, словоформы «*уволенный*» и «*увольняемый*» представляют одно слово. Данный параметр вместе с предыдущим параметром позволяет определить количество измененных словоформ.

3. Количество общих последовательностей словоформ. Например, в первой статье «*(расходование финансовых средств подразделениями) осуществляется (с учетом мнения профсоюзного) комитета*» и во второй статье «*выделение и (расходование финансовых средств подразделениями) происходит (с учетом мнения профсоюзного) органа подразделения*» присутствует две общие последовательности, выделенные скобками. Параметр позволяет определить, различны статьи или одна является модификацией другой.

Данные параметры представляются эксперту, который принимает решение о целесообразности анализа статьи.

Для определения количества одинаковых слов (лексем) используется разработанный метод генерации и определения форм слов.

В текстах статей могут встречаться как числительные, так и числа («*продолжительностью двадцать дней*» или «*продолжительностью 20 дней*»). Поэтому текст статьи нормализуется, и все числительные преобразуются в числа. Для нормализации текста используется метод обработки числительных.

Система сравнения коллективных трудовых договоров ускоряет анализ договоров. Величина сокращения времени анализа зависит от различия новой и предыдущей редакции договоров. Чем меньше изменился договор, тем больше сократится время анализа договора.

Для проверки работы системы были использованы старая и новая редакции тестовых трудовых договоров. В них 18 % статей были одинаковыми, а 26 % статей были переработанными вариантами старых статей. В результате время анализа договора с использованием данной системы сократилось на 34 %.

Автоматизация сравнения статей позволила упростить работу экспертов по анализу статей договоров, а значит, сократить время и повысить качество их работы.

Система сравнения статей коллективных трудовых договоров разработана в рамках нескольких НИР.

Заключение. Таким образом, разработаны системы проверки знаний по морфологии, правилам образования количественных числи-

тельных и функционированию алгоритмических моделей, в основе которых лежат предложенные методы обработки форм слов и количественных числительных. В данных системах используется динамическая генерация заданий по шаблону, позволяющая получить большое число вариантов одного задания. Системы проверки знаний являются частью АОС по этим темам и позволяют снизить затраты на обучение, упростить процесс обучения, сделать его автоматизированным и дистанционным. Использование системы проверки знаний функционирования алгоритмических моделей для подготовки к экзамену позволило повысить результаты сдачи экзамена на 7 %.

Система сравнения статей коллективных трудовых договоров позволяет упростить работу эксперта и повысить ее эффективность по времени (на 34 %) и качеству выполнения.

Можно предположить, что существуют аналогичные системы. Однако провести их сравнительный анализ с предлагаемыми системами затруднительно, так как аналогичные системы являются специализированными и, как правило, не выходят за пределы тех организаций, в которых или для которых они были разработаны.

Все описанные системы основаны на предложенных методах. Реализация данных систем позволила проверить работоспособность этих методов, показать их практическую значимость и актуальность.

Библиографический список

1. *Пруцков А.В.* Определение и генерация сложных форм слов естественных языков при морфологическом анализе и синтезе // Известия Таганрогского государственного радиотехнического университета. – Таганрог, 2006. – № 15 (70). – С. 10-14.

2. *Пруцков А.В.* Обработка числительных естественных языков с помощью формальных грамматик и нормальных алгоритмов Маркова // Вестник Рязанского государственного радиотехнического университета. – 2009. – Вып. 28. – С. 49-55.

3. *Пруцков А.В.* Применение информационных ресурсов для автоматизации обучения и проверки знаний // Информационные ресурсы России. – 2005. – № 1. – С. 18-20.

4. *Крупский В.Н., Плиско В.Е.* Теория алгоритмов: учеб. пособие для студ. вузов. – М.: Издательский центр «Академия», 2009. – 208 с.

5. *Жуков Д.* Тестирование знаний по физике с помощью интеллектуальных компьютерных «генераторов» // Информационные ресурсы России. – 2004. – № 4. – С. 22-25.

6. *Сулейманов Д.Ш., Гильмуллин Р.А., Сафина Л.Р.* Использование компьютерных технологий в обучении: на примере обучающе-тестирующей программы «Морфологический анализатор» // Educational Technology & Society. – 2006. – № 9 (4). – С. 293-305.

7. *Система внутреннего тестового контроля знаний РГРТУ*: метод. указ. / Рязан. гос. радиотехн. ун-т; сост.: С.А. Батуркин, А.М. Гостин, А.В. Пруцков и др.; под ред. проф. В.С. Гурова. – Рязань, 2007. – 68 с.

8. *Марчук Ю.Н.* Компьютерная лингвистика. – М.: АСТ; Восток-Запад, 2007. – 317 с.