МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК

МАТЕМАТИЧЕСКИЕ СТРУКТУРЫ и МОДЕЛИРОВАНИЕ

Выпуск 22

ОМСК
2011

Журнал «**Математические структуры и моделирование**». — Омск : Ом. гос. ун-т, 2011. — Вып. 22. — 139 с.

ISBN 978-5-7779-1244-2

Адрес научной редакции

Россия, 644053, Омск-53, ул. Грозненская, 11
Омский государственный университет
факультет компьютерных наук

E-mail: guts@omsu.ru
msm@cmm.univer.omsk.su

ISBN 978-5-7779-1244-2

МАТЕМАТИЧЕСКИЕ СТРУКТУРЫ и МОДЕЛИРОВАНИЕ

В журнале публикуются статьи, в которых излагаются результаты исследований по фундаментальной и прикладной математике, теоретической физике и размышления, касающиеся окружающей нас природы и общества.

Публикуются также статьи по информационным технологиям, компьютерным наукам, защите информации, философии и истории математики.

Объекты исследования должны быть представлены в форме некоторых математических структур и моделей.

Журнал является реферируемым. Рефераты статей публикуются в «Реферативном журнале» и в журналах «Zentralblat für Mathematik» (Германия) и «Mathematical Reviews» (США).

Электронная версия журнала представлена в сети Интернет по адресу:

`http://cmm.univer.omsk.su`

Журнал издается на коммерческие средства факультета компьютерных наук Омского государственного университета.

Наш E-mail:

`msm@cmm.univer.omsk.su`

Подробную информацию можно найти на Web-сервере:

`http://cmm.univer.omsk.su`

СОДЕРЖАНИЕ

Прикладная математика и моделирование

Теоретическая физика

Компьютерные науки

Продолжение на следующей странице

Информационная безопасность

Математические структуры и моделирование
2011, вып. 22, с. 5–14

УДК 517.91

ОЦЕНКА ПОГРЕШНОСТИ ВЫЧИСЛЕНИЯ ТОЧКИ ПЕРЕСЕЧЕНИЯ ПРОДОЛЖЕНИЯ КРИВОЙ РЕШЕНИЯ ЗАДАЧИ КОШИ С ПОВЕРХНОСТЬЮ РАЗРЫВА

В. В. Коробицын, Ю. В. Фролова

The method for computing an intersection of solution for system of ordinary differential equations with continuous break surface is suggested. The feature of algorithm is an ability to compute the near points on different sides of the surface. These points are the approximation of accurate solution point. The convergence of the numerical method is proven.

1. Введение

Поведение систем управления, скорость изменения которых зависит от переключательной функции, часто моделируют с помощью гибридных систем. Поведение решения таких систем более сложное, чем у систем обыкновенных дифференциальных уравнений. Это зачастую делает невозможным поиск аналитического решения. Поэтому приходится использовать численные методы, а для этого нужны эффективные методы численного интегрирования гибридных систем.

Простым примером гибридной системы является линейная система управления, представленная в работе [16]. Решение такой системы может быть представлено замкнутыми циклами, скользящей траекторией и вибрирующим режимом. Все эти эффекты требуют внимательного изучения в различных прикладных областях. Для решения подобных систем используются разные модификации методов Рунге–Кутты. Так, в работе [18] векторное поле функции правой части переопределяется на поверхности разрыва. А решение находится численно с помощью встроенных средств математического пакета MATLAB: решателя ОДУ (ODE solver) и детектора событий (event detector). Статья [19] посвящена локализации событий для систем ОДУ. Авторы обеих работ предполагают, что функции $\mathbf{f}^+$ и $\mathbf{f}^-$, определяющие правую часть системы до и после разрыва, определены на всей области определения системы. Однако это не всегда реализуется. Часто правые части определены только в соответствующих областях

Омский государственный университет им. Ф. М. Достоевского.
E-mail: korobits@rambler.ru

системы по разные стороны от поверхности разрыва. В этом случае использование предлагаемых методов является невозможным. Поэтому мы предлагаем метод, определяющий точки пересечения кривой решения системы с поверхностью разрыва, используя определение функций только в областях непрерывности.

2. Постановка задачи

Предположим, что в пространстве $\mathbb{R}^n$ задана поверхность $G = \{\mathbf{x} : \mathbf{x} \in \mathbb{R}^n, g(\mathbf{x}) = 0\}$, где $g : \mathbb{R}^n \to \mathbb{R}$ — гладкая функция, имеющая непрерывные частные производные. Поверхность G разделяет пространство $\mathbb{R}^n$ на две области $D^+ = \{\mathbf{x} : \mathbf{x} \in \mathbb{R}^n, g(\mathbf{x}) > 0\}$ и $D^- = \{\mathbf{x} : \mathbf{x} \in \mathbb{R}^n, g(\mathbf{x}) < 0\}$. Определим также замкнутые области $\mathcal{D}^+ = D^+ \cup G$, $\mathcal{D}^- = D^- \cup G$.

Пусть в областях $\mathcal{D}^+$ и $\mathcal{D}^-$ определены непрерывные функции $\mathbf{f}^+(\mathbf{x}) : \mathcal{D}^+ \to \mathbb{R}^n$, $\mathbf{f}^-(\mathbf{x}) : \mathcal{D}^- \to \mathbb{R}^n$ и функция $\mathbf{x}(t) : \mathbb{R} \to \mathbb{R}^n$ является решением задачи Коши

$$\frac{d\mathbf{x}}{dt} = \begin{cases} \mathbf{f}^+(\mathbf{x}), & \mathbf{x} \in \mathcal{D}^+, \\ \mathbf{f}^-(\mathbf{x}), & \mathbf{x} \in D^-, \end{cases} \tag{1}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0,$$

где $\mathbf{x}_0 \in \mathcal{D}^+$ — заданная начальная точка, соответствующая t_0.

Задача состоит в том, чтобы численно определить координаты точки $\mathbf{x}^*$ пересечения кривой $\mathbf{x}(t)$ с поверхностью G, если такое имеется.

Очевидным способом решения этой задачи представляется следующий: численно находим кривую решения задачи Коши до тех пор, пока она не пересечёт поверхность G. Тогда, имея две соседние точки кривой решения, находящиеся по разные стороны от поверхности G, уточняем положение точки пересечения с помощью интерполяции. Но проблема заключается в том, что правая часть задачи Коши состоит из двух функций, определённых в областях $\mathcal{D}^+$ и $\mathcal{D}^-$, которые пересекаются по G. Если решать задачу (1) традиционными методами без учёта разрыва, то на поверхности разрыва численное решение сильно отклоняется от точного.

Для решения этой проблемы предлагается метод, состоящий из трёх шагов (предполагается, что $\mathbf{x}_0 \in D^+$):

1) с помощью метода Рунге–Кутты с управлением длиной шага находим точки кривой решения в области $\mathcal{D}^+$ вплоть до точки, находящейся на расстоянии не более заданной точности Tol от поверхности G;

2) используя найденные точки на шаге 1, строим полином Ньютона, приближающий кривую решения вблизи поверхности G;

3) находим пересечение полинома Ньютона с поверхностью G.

В этом методе на шаге 1 используется правая часть $\mathbf{f}^+$. А если в схеме Рунге–Кутты требуется значение функции правой части в точке, принадлежащей области $\mathcal{D}^-$, то вычисления прерываются и решается альтернатива: уменьшить

шаг, чтобы приблизиться к G, или перейти к шагу 2, если расстояние до поверхности G мало (меньше Tol).

Далее вместо кривой решения используется её приближение полиномом Ньютона. Полином заменяет кривую решения вблизи поверхности G в областях $\mathcal{D}^+$ и $\mathcal{D}^-$. Но строится полином по точкам решения в области $\mathcal{D}^+$, а это означает, что метод будет успешен только тогда, когда полином будет построен по точкам, находящимся в достаточной близости к поверхности G.

Пересечение полинома Ньютона с поверхностью G находим с помощью итерационного метода Ньютона. Коэффициент в итерационной процедуре должен быть подобран таким образом, чтобы каждый шаг оказывался на противоположной стороне от поверхности G. Тем самым обеспечивается приближение к точке пересечения с обеих сторон от поверхности разрыва.

Доказательство сходимости представленного метода можно провести в несколько этапов:

1) доказать сходимость метода решения ОДУ, в частности, метода Рунге–Кутты четвёртого порядка. Решение ОДУ приближается конечными приращениями по схеме Рунге–Кутты. Сходимость обеспечивается уменьшением шага интегрирования;

2) доказать сходимость приближения полиномом Ньютона, построенного по заданным значениям точек кривой, к кривой функции в локальной области. Сходимость обеспечивается уменьшением шага между опорными точками;

3) доказать сходимость метода Ньютона для нахождения точки пересечения кривой, заданной гладкой функцией в явном виде, с поверхностью, заданной в виде уравнения $g(x) = 0$.

Доказательство сходимости трёх этапов и согласованность их даёт доказательство сходимости всего метода.

3. Первый этап: вычисление опорных точек

Задача состоит в том, чтобы найти опорные точки для дальнейшей аппроксимации. Опорные точки являются приближением к точкам кривой решения ОДУ

$$\frac{dx}{dt} = f(x),\ x(t_0) = x_0,\ x : \mathbb{R} \to \mathbb{R}^n,\ f : \mathbb{R}^n \to \mathbb{R}^n. \tag{2}$$

Необходимо вычислить точки $x_1 \approx x_1(t)$, $x_2 \approx x_2(t)$, где $t_1 = t_0 + \tau$, $t_2 = t_0 + 2\tau$. Будем использовать для этого метод Рунге–Кутты 4-го порядка:

$$x_1 = RK4(f, x_0, t_0, \tau), \quad x_2 = RK4(f, x_1, t_1, \tau). \tag{3}$$

Можно также вычислить решение с удвоенным шагом 2τ.

$$\widehat{x}_2 = RK4(f, x_0, t_0, 2\tau).$$

Оценка погрешности метода Рунге–Кутты задаётся неравенством

$$|x_2 - x(t_2)| \le C \cdot \tau^{k+1},$$

где k – порядок метода, в нашем случае равен 4.

На практике погрешность можно оценить либо вложенной схемой, либо оценкой Ричардсона, для чего и вычислялось значение $\widehat{x}_2$. Поскольку

$$|x_2 - x(t_2)| \le C \cdot \tau^{k+1}, \quad |\widehat{x}_2 - x(t_2)| \le C \cdot (2\tau)^{k+1} \tag{4}$$

и $|\widehat{x}_2 - x_2 + x_2 - x(t_2)| \le |\widehat{x}_2 - x_2| + |x_2 - x(t_2)|$, то получаем

$$C \cdot (2\tau)^{k+1} \le |\widehat{x}_2 - x_2| + C \cdot \tau^{k+1}.$$

Отсюда $|\widehat{x}_2 - x_2| \ge C \cdot \tau^{k+1}(2^{k+1} - 1)$. Следовательно,

$$C \cdot \tau^{k+1} \le \frac{|\widehat{x}_2 - x_2|}{2^{k+1} - 1}.$$

Тогда, подставив это выражение в (4), получим

$$|x_2 - x(t_2)| \le C \cdot \tau^{k+1} \le \frac{|\widehat{x}_2 - x_2|}{2^{k+1} - 1}.$$

При точном значении $x(t_0) = x_0$ отклонения x_2 от $x(t_2)$ ограничено величиной $\delta = C\tau^{k+1}$, которая стремится к нулю при $\tau \to 0$.

4. Второй этап: построение полинома

Заданы три точки $(t_1, x_1), (t_2, x_2), (t_3, x_3)$, причём $t_2 = t_1 + \tau$, $t_3 = t_2 + \tau$, кроме того, известны значения первой производной аппроксимируемой функции f_1, f_2, f_3 в этих точках (вектора $\frac{dx}{dt}(t_1), \frac{dx}{dt}(t_2), \frac{dx}{dt}(t_3)$). Необходимо построить интерполирующую функцию (многочлен Ньютона) по трём равноотстоящим узлам с кратностью 2. Таким образом, максимальная степень многочлена $s < 6$.

Условия построения многочлена:

$$N_s(t_1) = x_1, N_s'(t_1) = f_1, N_s(t_2) = x_2, N_s'(t_2) = f_2, N_s(t_3) = x_3, N_s'(t_3) = f_3.$$

Таблица разделённых разностей имеет вид

$$\begin{array}{llllll}
t_1 & x_1 & & & & \\
 & & f_1 & & & \\
t_1 & x_1 & & \frac{d_1 - f_1}{\tau} & & \\
 & & d_1 & & \frac{f_2 - 2d_1 + f_1}{\tau^2} & \\
t_2 & x_2 & & \frac{f_2 - d_1}{\tau} & & \frac{d_2 - 4f_2 + 5d_1 - 2f_1}{4\tau^3} \\
 & & f_2 & & \frac{d_2 - 2f_2 + d_1}{2\tau^2} & \\
t_2 & x_2 & & \frac{d_2 - f_2}{\tau} & & \frac{2f_3 - 5d_2 + 4f_2 - d_1}{4\tau^3} \\
 & & d_2 & & \frac{f_3 - 2d_2 + f_2}{\tau^2} & \\
t_3 & x_3 & & \frac{f_3 - d_2}{\tau} & & \\
 & & f_3 & & & \\
t_3 & x_3 & & & &
\end{array} \qquad \frac{2f_3 - 6d_2 + 8f_2 - 6d_1 + 2f_1}{8\tau^4}.$$

Здесь используются следующие обозначения $d_1 = \frac{x_2 - x_1}{\tau}$, $d_2 = \frac{x_3 - x_2}{\tau}$. Тогда полиномы 4-го и 5-го порядков будут иметь вид:

$$N_4(t_3 + \theta) = x_3 + \theta R_5^1 + \theta^2 R_4^2 + \theta^2(\theta + \tau) R_3^3 + \theta^2(\theta + \tau)^2 R_2^4,$$

$$N_5(t_3 + \theta) = N_4(t_3 + \theta) + \theta^2(\theta + \tau)^2(\theta + 2\tau) R_1^5.$$

После подстановки значений конечных разностей из таблицы получаем

$$\begin{gathered} N_4(t_3 + \theta) = x_3 + \theta f_3 + \theta^2 \cdot \tfrac{f_3 - d_2}{\tau} + \theta^2(\theta + \tau) \cdot \tfrac{f_3 - 2d_2 + f_2}{\tau^2} + \\ + \theta^2(\theta + \tau)^2 \cdot \tfrac{2f_3 - 5d_2 + 4f_2 - d_1}{4\tau^3}, \end{gathered} \tag{5}$$

$$N_5(t_3 + \theta) = N_4(t_3 + \theta) + \theta^2(\theta + \tau)^2(\theta + 2\tau) \tfrac{2f_3 - 6d_2 + 8f_2 - 6d_1 + 2f_1}{8\tau^4}.$$

Ошибки аппроксимации задаются выражениями:

$$N_4(t_3 + \theta) - x(t_3 + \theta) = \theta^2(\theta + \tau)^2(\theta + 2\tau) \cdot \frac{x^{(5)}(\xi_1)}{5!}, \quad t_1 \le \xi_1 \le t_3 + \theta,$$

$$N_5(t_3 + \theta) - x(t_3 + \theta) = \theta^2(\theta + \tau)^2(\theta + 2\tau)^2 \cdot \frac{x^{(6)}(\xi_2)}{6!}, \quad t_1 \le \xi_2 \le t_3 + \theta.$$

При $\theta \in [t_3; t_3 + \tau]$ получим

$$|N_4(t_3 + \theta) - x(t_3 + \theta)| \le \frac{M_5 \tau^5}{10}, \quad |N_5(t_3 + \theta) - x(t_3 + \theta)| \le \frac{M_6 \tau^6}{20}, \tag{6}$$

где $M_i = \sup |x^{(i)}(t)|,\ t_1 \le t \le t_1 + 3\tau$.

Два полинома приведены не случайно. Для практической оценки погрешности можно воспользоваться соотношением

$$|N_4(t_3 + \theta) - x(t_3 + \theta)| \approx |N_5(t) - N_4(t)|.$$

Использование полинома N_4 является более рациональным потому, что опорные узлы для аппроксимации получены на первом этапе численным методом 4-го порядка. Учитывая, что опорные точки были получены с помощью метода Рунге–Кутты 4-го порядка, оценим погрешность аппроксимации кривой решения полиномом Ньютона при заданном отклонении опорных точек.

Пусть значение в точке t_1 аппроксимируемой функции задано точно, а значения в точках t_2 и t_3 известны с некоторой погрешностью, не превосходящей $\delta = C \cdot \tau^5$. Запишем случайное смещение для опорных точек $x_2^* = x_2 + \Delta x_2$, $x_3^* = x_3 + \Delta x_3$ с заданным ограничением $|\Delta x_2| \le \delta$, $|\Delta x_3| \le \delta$. Тогда значения производных тоже будут получены с погрешностью $f_2^* = f(x_2^*)$, $f_3^* = f(x_3^*)$. А значит, аппроксимирующий полином будет отклонён от точной функции больше, чем полином, построенный по точным значениям опорных точек. В связи с этим существенным будет доказательство сходимости смещённого полинома к точному решению.

Лемма 1. *Если опорные точки x_i $(i = 2, 3)$ возмущены не более чем на δ $(|x_i^* - x_i| \le \delta)$ и производная аппроксимируемой функции ограничена величиной B в рассматриваемой области $(|f'(x)| \le B)$, то многочлен $N_4^*(t)$, построенный по возмущённым узловым точкам, будет равномерно сходиться к $N_4(t)$ при $\delta \to 0$ с оценкой погрешности $|N_4^*(t_3 + \theta) - N_4(t_3 + \theta)| \le \delta + \tau\delta(21 + 12B)$ при $\theta \le \tau$.*

Доказательство. Для оценки погрешности полинома N_4^* необходимо вычислить погрешность каждого слагаемого из (5) и сложить их.

Оценим ошибку значения функции в узлах $i = 2, 3$:

$$f_i^* = f(x_i^*) = f(x_i) + f'(x_i) \cdot (x_i^* - x_i) + o(x_i^* - x_i).$$

Отсюда получаем оценку $|f_i^* - f_i| \le B\delta$. Будем считать $f_i^* = f_i + \Delta f_i$, $|\Delta f_i| \le B\delta$.

Оценим отклонения d_1^* и d_2^*.

$$d_1^* = \frac{x_2^* - x_1}{\tau} = \frac{x_2 + \Delta x_2 - x_1}{\tau} = \frac{x_2 - x_1}{\tau} + \frac{\Delta x_2}{\tau} = d_1 + \Delta d_1.$$

Величина d_1^* имеет погрешность $\Delta d_1 = \frac{\Delta x_2}{\tau}$, которая оценивается по формуле

$$|\Delta d_1| = \left|\frac{\Delta x_2}{\tau}\right| \le \frac{\delta}{\tau}.$$

Для d_2^* получаем

$$d_2^* = \frac{x_3^* - x_2^*}{\tau} = \frac{x_3 + \Delta x_3 - x_2 - \Delta x_2}{\tau} = \frac{x_3 - x_2}{\tau} + \frac{\Delta x_3 - \Delta x_2}{\tau} = d_2 + \Delta d_2,$$

где

$$|\Delta d_2| = \left|\frac{\Delta x_3 - \Delta x_2}{\tau}\right| \le \frac{2\delta}{\tau}.$$

Оценим конечные разности из (5).

$$R_4^{2*} = \frac{f_3^* - d_2^*}{\tau} = \frac{f_3 + \Delta f_3 - d_2 - \Delta d_2}{\tau} = \frac{f_3 - d_2}{\tau} + \frac{\Delta f_3 - \Delta d_2}{\tau} = R_4^2 + \Delta R_4^2.$$

Отклонение ΔR_4^2 оценивается неравенством

$$\left|\Delta R_4^2\right| \le \left|\frac{\Delta f_3}{\tau}\right| + \left|\frac{\Delta d_2}{\tau}\right| = \frac{(2 + B)\delta}{\tau}.$$

Аналогично оценим отклонения ΔR_3^3 и ΔR_2^4.

$$\left|\Delta R_3^3\right| \le \left|\frac{\Delta f_3}{\tau^2}\right| + 2\left|\frac{\Delta d_2}{\tau^2}\right| + \left|\frac{f_2}{\tau^2}\right| = \frac{B\delta}{\tau^2} + \frac{4\delta}{\tau^2} + \frac{B\delta}{\tau^2} = \frac{(4 + 2B)\delta}{\tau^2},$$

$$\left|\Delta R_2^4\right| \le \left|\frac{2\Delta f_3}{4\tau^3}\right| + \left|\frac{5\Delta d_2}{4\tau^3}\right| + \left|\frac{4\Delta f_2}{4\tau^3}\right| + \left|\frac{\Delta d_1}{4\tau^3}\right| = \frac{(11 + 6B)\delta}{4\tau^3}.$$

Максимальная абсолютная погрешность полиномиальной интерполяции с заданными отклонениями опорных узлов оценивается неравенством

$$\left|N_4^*(t_3+\theta) - N_4(t_3+\theta)\right| \le \delta + \tau\delta(21+12B),$$

где δ — максимальное отклонение опорных точек; $\tau\delta(21+12B)$ — отклонение многочлена в точке, расположенной на расстоянии не более τ от опорной.

Из этого неравенства следует сходимость метода приближения кривой многочленом Ньютона при $\delta \to 0, \tau = const$ и $|B| < \infty$. Лемма 1 доказана. ■

Следствие 1. *Оценка погрешности аппроксимации точного решения $x(t)$ задачи (2) с помощью полинома (5), построенного по опорным точкам с помощью метода (3), удовлетворяет условию*

$$|N_4^*(t_3+\theta) - x(t_3+\theta)| \le K \cdot \tau^5,$$

где $K = \frac{M_5}{10} + C(1+\tau(21+12B))$ при $\theta < \tau$.

Доказательство следует из оценки погрешности метода Рунге–Кутты (4) и леммы 1.

5. Третий этап: поиск пересечения полиномиальной кривой с гладкой поверхностью

Задача третьего этапа состоит в том, чтобы вычислить координаты $\mathbf{x}^* \in \mathbb{R}^n$ точки пересечения кривой $\mathbf{x} = N_4(t_3+\theta)$ с поверхностью $g(\mathbf{x}) = 0$, где $g : \mathbb{R}^n \to \mathbb{R}$ — гладкая функция, имеющая непрерывные частные производные.

Введём функцию $h(\theta) := g(N_4(t_3+\theta))$. Задача нахождения x^* сводится к решению алгебраического уравнения

$$h(\theta) = 0. \tag{7}$$

В общем случае решение может не существовать или не быть единственным. Поэтому предположим, что на интервале $0 < \theta\tau$ имеется единственное решение θ^* уравнения (7). То есть $h(0)\cdot h(\tau) < 0$ и имеется единственное число θ^* такое, что $h(\theta^*) = 0$.

Применим итерационный метод Ньютона для решения уравнения (7), получим следующую процедуру для поиска параметра θ^*, соответствующего точке $\mathbf{x}^*$:

$$\theta_{i+1} = \theta_i - \frac{h(\theta_i)}{\frac{dh}{d\theta}(\theta_i)}. \tag{8}$$

В качестве начального приближения возьмём $\theta_0 = \tau/2$.

Сходимость итерационной процедуры может быть обеспечена, если будут выполнены условия следующей теоремы.

Теорема 1 (о сходимости метода Ньютона). *Если выполнены условия:*

1) $\left|\frac{dh}{d\theta}\right|^{-1} \le a_1 < \infty \quad при \quad 0 < \theta < \tau,$

2) $\left|h(\theta_1) - h(\theta_2) - \left(\frac{dh}{d\theta}\right)(\theta_2) \cdot \left(\theta_1 - \theta_2\right)\right| \le a_2|\theta_2 - \theta_1|^2$ *при* $0 < \theta_1, \theta_2 < \tau$,

3) $0 < \theta_0 < b$ *при* $b = \min\{\tau/2, \frac{1}{a_1 a_2}\}$,

то итерационный процесс Ньютона (8) сходится с оценкой погрешности

$$|\theta_i - \theta^*| \le \frac{1}{a_1 a_2}\left(a_1 a_2 |\theta_0 - \theta^*|\right)^{2^i}.$$

Условие теоремы переформулировано в принятых обозначениях. Доказательство теоремы приведено в [1]. Мы докажем, что условия этой теоремы выполняются для нашей задачи. Тем самым мы докажем сходимость итерационной процедуры (8).

Лемма 2. *Итерационная процедура (8) сходится к решению уравнения (7), если выполнены условия:*

1) $b_1 > 0$, *где* $b_1 = \inf\limits_{0<\theta<\tau} |\frac{dh}{d\theta}|$;

2) $b_2 < \infty$, *где* $b_2 = \sup\limits_{0<\theta<\tau} |\frac{d^2h}{d\theta^2}|$;

3) $\tau \le \frac{b_1}{b_2}$.

Доказательство. Доказательство леммы сводится к проверке выполнения условий теоремы 1, откуда будет вытекать сходимость итерационной процедуры (8).

Первое условие теоремы будет выполнено, поскольку $b_1 > 0$. Достаточно взять $a_1 = 1/b_1$.

Так как $b_2 < \infty$, то достаточно взять $a_2 = 2b_2$ и условие 2 теоремы будет выполнено.

Поскольку $\tau \le \frac{b_1}{b_2}$, то верно условие $\frac{1}{a_1 a_2} \ge \frac{\tau}{2}$. Следовательно, для $\theta_0 = \tau/2$ будет выполнено условие 3 теоремы 1. В итоге все условия теоремы 1 выполнены. Лемма 2 доказана. ■

Из теоремы 1 также следует, что достаточным условием достижения требуемой точности ε можно взять $|\theta_n - \theta_{n-1}| \le \varepsilon$, при этом будет выполнено условие $|\theta_n - \theta^*| \le \varepsilon$.

Итак, доказана сходимость численного метода. Сходимость метода Рунге–Кутты обеспечивает приближение опорных точек к точкам кривой решения. Сходимость метода приближения к кривой многочленом Ньютона даёт приближение аппроксимирующей кривой к решению. Сходимость итерационной процедуры обеспечивает нахождение точки пересечения кривой решения с заданной поверхностью. Полученный результат можно сформулировать в виде теоремы.

Теорема 2. *Численное решение системы (2), найденное по формулам (7), (5), (3), сходится к точному решению при* $\tau \to 0$, *если функция* f *имеет ограниченные производные в области определения и кривая решения* $x(t)$ *пересекает поверхность* $g(x) = 0$.

6. Выводы

Теорема 2 даёт теоретическое обоснование сходимости предложенного метода нахождения точки пересечения кривой решения с поверхностью разрыва.

Литература

1. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. М.: Бином. Лаборатория знаний, 2008.
2. Деккер К., Вервер Я. Устойчивость методов Рунге–Кутты для жестких нелинейных дифференциальных уравнений. М.: Мир, 1988. 334 с.
3. Коробицын В. В., Маренич В. Б., Фролова Ю. В. Исследование поведения явных методов Рунге–Кутты при решении систем обыкновенных дифференциальных уравнений с разрывной правой частью // Математические структуры и моделирование. 2007. Вып. 17. С. 19–25.
4. Коробицын В. В., Фролова Ю. В. Алгоритм численного решения дифференциальных уравнений с разрывной правой частью // Математические структуры и моделирование. 2005. Вып. 15. С. 46–54.
5. Коробицын В. В., Фролова Ю. В., Маренич В. Б. Алгоритм численного решения кусочно-сшитых систем // Вычисл. технологии. 2008. Т. 13. № 2. С. 70–81.
6. Новиков Е. А., Каменщиков Л. П. Реализация полуявного (4,2)-метода решения жёстких систем на параллельной ЭВМ // Вычислительные технологии. 2004. Т. 9. Вестник КазНУ им. аль-Фараби. Сер.: Математика, механика, информатика. № 3 (42). (Совм. выпуск. Ч. 1). С. 235–241.
7. Фельдман Л. П. Сходимость и оценка погрешности параллельных одношаговых блочных методов моделирования динамических систем с сосредоточенными параметрами // Научные труды ДонНТУ. Сер. Информатика, моделирование и вычислительные методы (ИКВТ-2000). Вып. 15. Донецк: ДонНТУ, 2000. С. 34–39.
8. Фельдман Л., Назарова И. А. Параллельные алгоритмы численного решения задачи Коши для систем обыкновенных дифференциальных уравнений // Матем. моделирование. 2006. Т. 18. № 9. С. 17–31.
9. Филиппов А. Ф. Дифференциальные уравнения с разрывной правой частью. М.: Наука, 1985. 224 с.
10. Хайрер Э., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Жёсткие и дифференциально-алгебраические задачи. М.: Мир, 1999. 685 с.
11. Хайрер Э., Нёрсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежёсткие задачи. М.: Мир, 1990. 512 с.
12. Cash J. R., Karp A. H. A variable order Runge–Kutta method for initial value problems with rapidly varying right-hand sides // ACM Transactions on Mathematical Software. 1990. Vol. 16. No. 3. P. 201–222.
13. Gear C. W., Osterby O. Solving ordinary differential equations with discontinuities // ACM Transactions on Mathematical Software. 1984. Vol. 10. P. 23–44.
14. Guglielmi N., Hairer E. Computing breaking points in implicit delay differential equations // Advances in Computational Mathematics. 2008. Vol. 29. No. 3. P. 229–247.

15. Jackson K., Norsett S. The potential for parallelism in Runge–Kutta methods. Part I: RK formulas in standard form // SIAM J. Numer. Anal. 1995. Vol. 32. P. 49–82.
16. Johansson K. H., Barabanov A. E., Astrom K. J. Limit cycles with chattering in relay feedback systems // IEEE Transations on Automatic Control. 2002. Vol. 247. P. 1414–1423.
17. Park T., Barton P. I. State event location in differential-algebraic models // ACM Transactions on Modeling and Computer Simulation. 1996. Vol. 6. No. 2. P. 137–165.
18. Piiroinen P. T., Kuznetsov Y. A. An event-driven method to simulate Filippov systems with accurate computing of sliding motions // ACM Transactions on Mathematical Software. 2008. Vol. 34. No. 13. P. 1–24.
19. Shampine L. F., Thompson S. Event location for ordinary differential equations // Computer and Mathematics with Application. 2000. Vol. 39. P. 43–54.

Математические структуры и моделирование
2011, вып. 22, с. 15–23

УДК 004.94, 303.094.7

КОМПЬЮТЕРНАЯ РЕАЛИЗАЦИЯ МОДЕЛИ РЕГИОНАЛЬНОГО РАЗВИТИЯ ГОСУДАРСТВА

А. А. Лаптев, Ю. С. Трофимчук

Описаны реализация и исследование математической модели регионального развития государства.

Введение

Модель регионального развития государства основана на идее о функционировании государства-империи посредством становления административных центров [2, 3, 9].

Модель построена на основе следующих предположений:

1. Развитие государства рассматривается через изменение численности населения, добычу и воспроизводство ресурсов, территориальное изменение, появление и распад административных центров.
2. Ресурс — пространственная характеристика. Это некоторое усреднённое значение по всем ресурсам. Не производится деление ресурсов на составляющие. Рассматриваются только природные (возобновляемые и невозобновляемые) ресурсы, их добыча и возобновление.
3. Количество административных центров ограничено. За каждым административным центром закреплена своя территория. Считается, что территория принадлежит государству, если на неё распространено влияние какого-либо административного центра данного государства.
4. Политическое влияние центра (управление периферией) — пространственная характеристика. Она показывает степень влияния центра на зависимые от него территории. Власть как бы распространяется по территории.
5. Сила, мощь административного центра — это характеристика каждого центра.
6. Изменение границ и изменение властных центров — некоторая внешняя функция центра государства-империи.
7. Внешние параметры для административного центра — это уровень развития политической и экономической систем. Они определяются с помощью математической модели социогенеза [9].

Омский государственный университет им. Ф. М. Достоевского.
E-mail: aalaptev@mail.ru

Математическая модель развития государства построена и описана ранее [6, 7, 9]. Требовалось реализовать программно данную математическую модель и исследовать готовую компьютерную модель. Первый из двух этапов включает реализацию решения системы дифференциальных уравнений с помощью численных методов, отображение изменения характеристик на географической карте и в виде графиков их зависимости от времени. После компьютерного моделирования необходимо также оценить адекватность модели исследуемым социальным процессам.

1. Компьютерная реализация модели

Для компьютерной реализации потребовалось несколько упростить математическую модель — функциональные коэффициенты заменены постоянными, некоторые коэффициенты и зависимости исключены (1). *Мощь административного центра* описывается функцией $M_i(t) : \mathbb{R}^+ \to \mathbb{R}$ (i — номер центра), *численность населения* — функцией $N_i(t) : \mathbb{R}^+ \to \mathbb{R}$, *природный ресурс* — функцией $R^*(x, y, t) : \mathbb{R}^3 \to \mathbb{R}$, *добытый ресурс* — функцией $R_i(t) : \mathbb{R}^+ \to \mathbb{R}$, *политическое влияние центра* — функцией $P_i(x, y, t) : \mathbb{R}^3 \to \mathbb{R}$.

Развитие государства рассматривается в некоторой области $\Omega \in \mathbb{R}^2$. $b_M^n, b_M^R, b_M^N, d_M^n, d_M^R, d_M^N, f_G, \alpha_{ij}, d_r^i, k_K, k_r, r_0, p, L$ — постоянные коэффициенты, характеризующие конкретный административный центр.

$$\begin{cases} \frac{dM_i}{dt} = b_M^n \cdot b_M^R \cdot b_M^N \cdot M_i - d_M^n \cdot d_M^R \cdot d_M^N \cdot M_i, \\ \frac{\partial P_i}{\partial t}(x, y, t) = \triangle P_i + f_G M_i - \nabla R^* \cdot \nabla P_i - \sum\limits_j \alpha_{ij} P_j \cdot P_i, \\ \frac{\partial R^*}{\partial t}(x, y, t) = \sum\limits_i (-d_r^i \cdot k_K \cdot P_i(x, y, t) \cdot R^*(x, y, t)) + k_r, \\ \frac{dR_i}{dt}(t) = \iint\limits_\Omega (d_r^i \cdot k_K \cdot P_i(x, y, t) \cdot R^*(x, y, t)) dxdy - r_0 \cdot N_i, \\ \frac{dN_i}{dt} = p \cdot N_i - L \cdot N_i^2. \end{cases} \tag{1}$$

Первое, четвёртое и пятое уравнения системы (1) решаются методом Рунге–Кутта четвёртого порядка [1]. Второе и третье уравнения системы решаются методом сеток (разностных схем) [4,5]. Граничные условия во втором уравнении задавались в виде $\frac{\partial P_i}{dn} = 0$. Использована явная схема с порядком аппроксимации по пространственной координате $0(h)$ и по времени $0(\tau)$.

Методы, необходимые для решения системы (1), и интерфейс программы выполнены с помощью пакета `Microsoft Visual Studio`.

Численные методы реализованы двумя основными функциями: решение систем дифференциальных уравнений методом Рунге–Кутта и метод сеток для параболического уравнения [4, 5, 8, 10].

Решение уравнений и вывод результатов на карте территорий выполняются параллельно и в программе реализованы отдельными потоками. В главном потоке выполняются все вычисления. Из него запускается интерфейсный поток, в котором создаётся отдельное диалоговое окно для вывода карты распределения ресурса R^* и отрисовки на ней изменений политического влияния

P_i для каждого центра. В то время как в интерфейсном потоке выполняется вывод в виде изображения числовых значений, записанных в память в предыдущий момент времени, в главном потоке вычисляются значения для вывода в интерфейсном потоке в следующий момент времени. Отрисовка графиков зависимости численности населения $N(t)$, мощи $M_i(t)$ и добытого ресурса $R_i(t)$ от времени t для каждого административного центра производится в главном потоке.

Ввод исходных данных осуществляется через интерфейс пользователя (см. рис. 1 и 2), который позволяет задать начальные условия и коэффициенты для уравнений системы (1), а также длительность эксперимента, число шагов по времени, масштаб для графиков, размер стороны квадрата для вывода карт. Ввод коэффициентов, начальных условий и загрузка карт осуществляется через диалоговые окна (см. рис. 2).

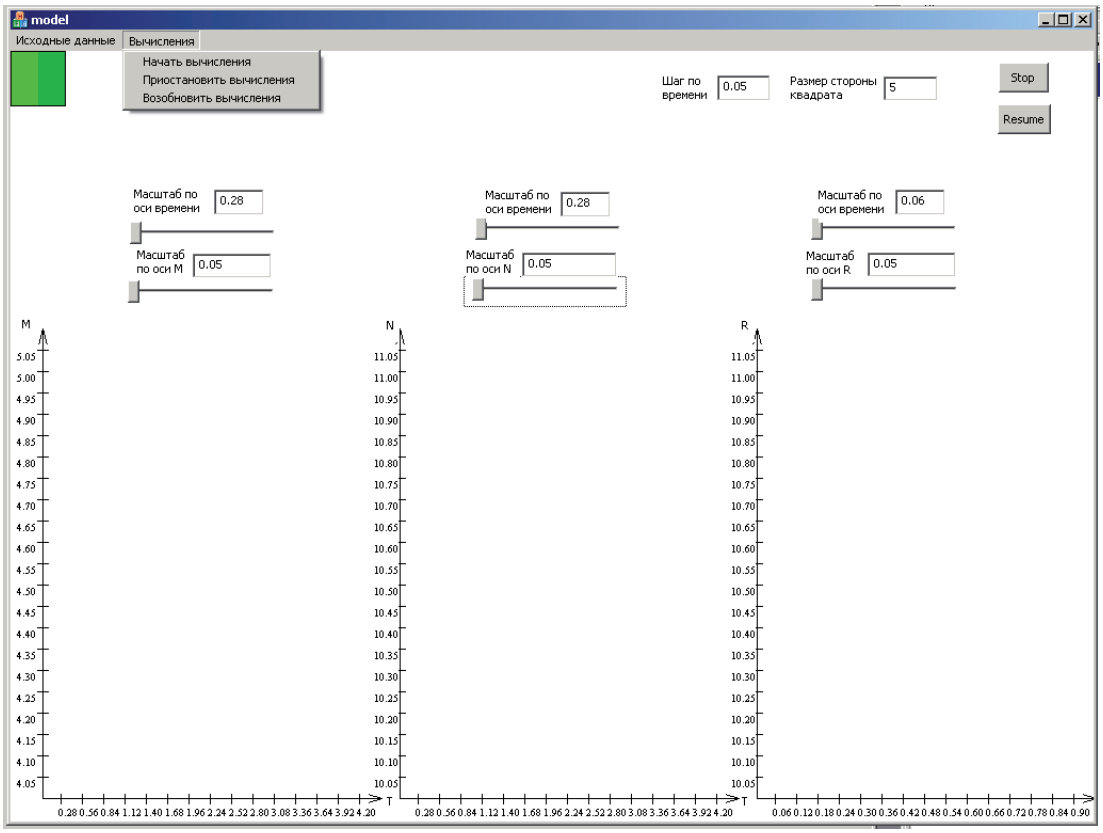

Рис. 1. Главное окно программы

При запуске вычислений в главном окне начинается отрисовка графиков. Одновременно открывается диалоговое окно, в которое выводится шкала значений ресурса R^*, карта распределения ресурса R^*, меняющаяся со временем. На ней осуществляется вывод распространяющегося политического влияния P_i для каждого центра разными цветами (см. рис. 3). Также для лучшей наглядности рядом (справа) снова отображается изменяющаяся карта ресурса.

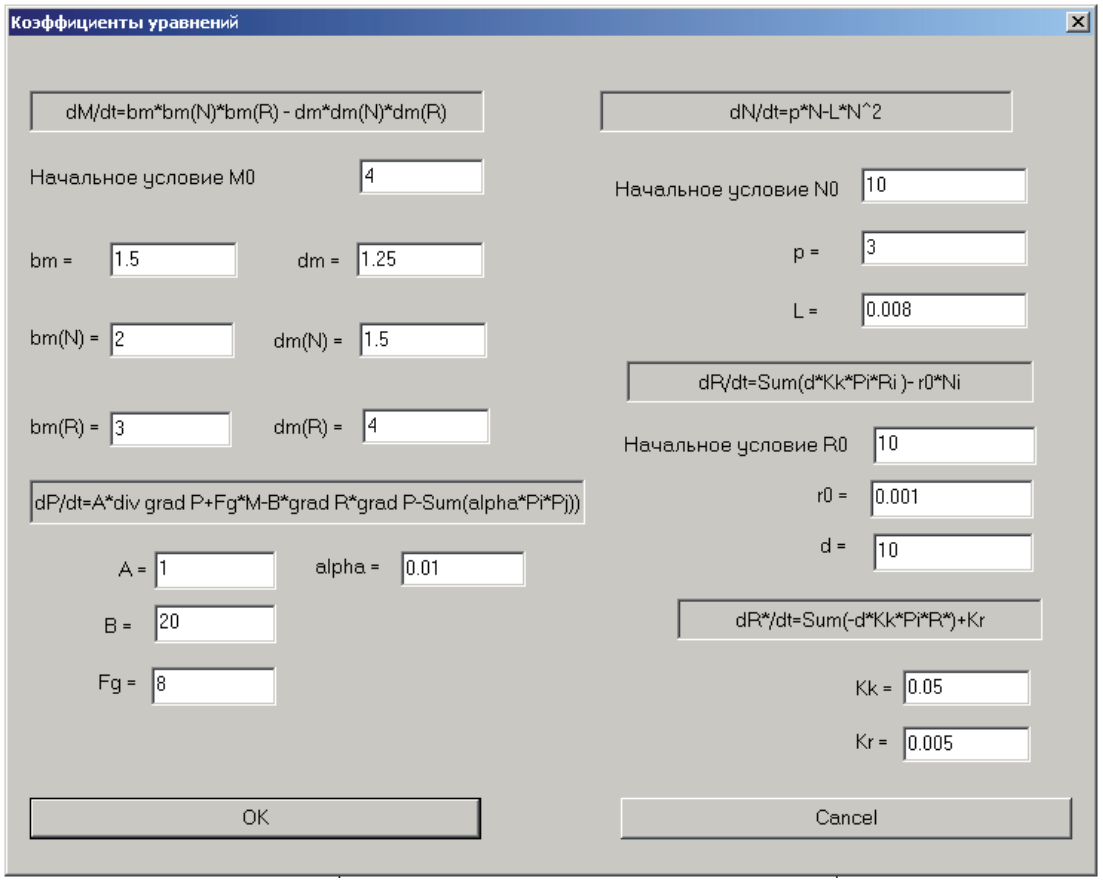

Рис. 2. Окно ввода коэффициентов

2. Моделирование

Для проверки адекватности модели были проведены эксперименты с одним и двумя политическими центрами.

Изначально тестирование проводилось на однородной карте (значение ресурса одинаково во всех точках). На этом этапе проводилась проверка работы сеточного метода для параболического уравнения: в начальный момент задавались ненулевые значения политического влияния только в центральной точке.

С течением времени эта область начинает разрастаться, а значения в центре — увеличиваться (цвета темнее). Поскольку ресурс распределён равномерно, влияние распространяется симметрично относительно начальных значений. При этом ресурс постепенно как бы «съедается», начиная от центра, в местах распространения политического влияния, т. е. значения ресурса R^* в этих местах уменьшаются. В то же время ресурс возобновляется, и области, на которые влияние ещё не распространено, закрашиваются в цвет, который в шкале соответствует большим значениям ресурса.

Так как ресурс по карте распределён равномерно на конечных этапах эксперимента (см. рис. 4), политическое влияние достигает всех четырёх границ квадратной карты практически одновременно.

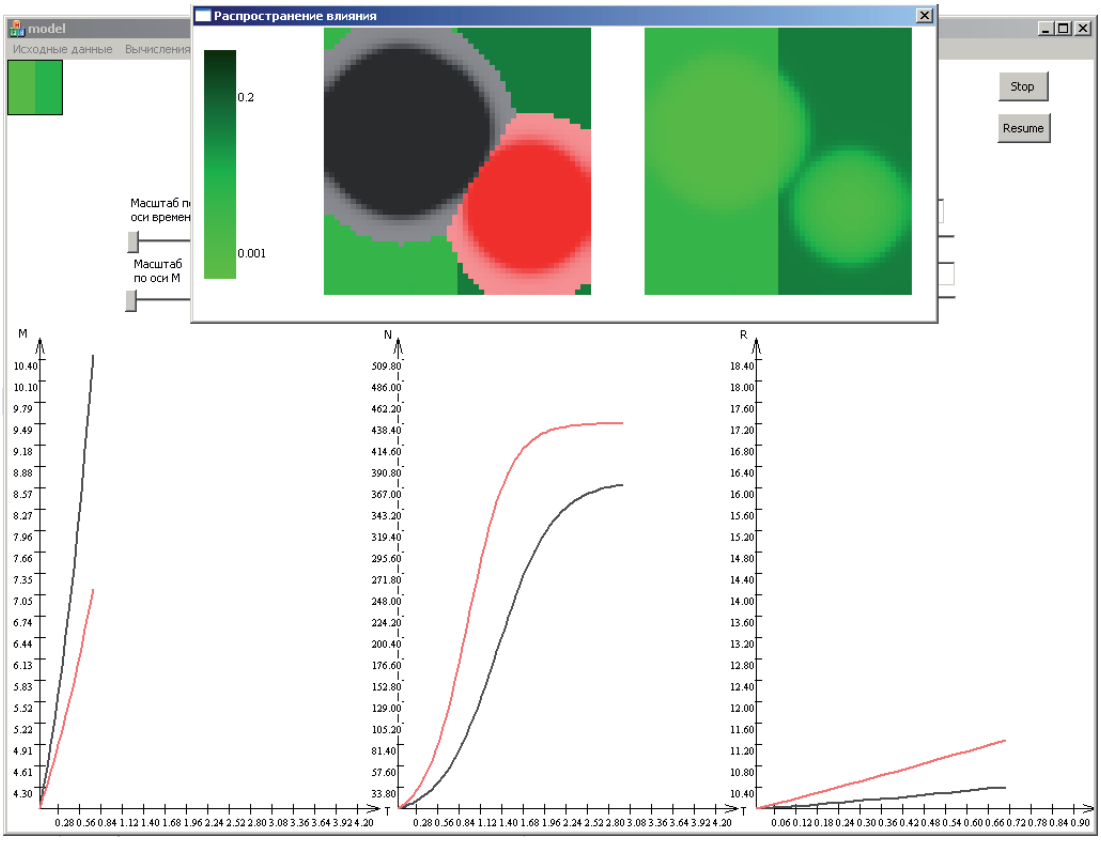

Рис. 3. Главное окно программы в процессе расчета

При неравномерном распределении начального ресурса распространение влияния больше направлено в ту сторону, где ресурсов больше.

Приведённые результаты позволяют сделать вывод об адекватности компьютерной модели математическим соотношениям: полученное решение параболического уравнения соответствует решениям в аналогичных задачах (например, в задачах теплопроводности).

Добавим в модель ещё один центр. Значения коэффициентов для первого и второго центров немного изменены. Карту ресурсов сделаем неоднородной.

На рис. 5 можно наблюдать начальные этапы эксперимента. У первого центра больше, чем у второго, значения начальных данных, также больше «подпитка» в центре и быстрее возрастает функция мощи. Второй центр начал своё развитие на той части карты ресурсов, где их значения больше. Поэтому второй центр на начальном этапе практически догнал первый по площади занятых территорий.

При столкновении влияние первого административного центра начинает распространяться на территории второго. Причём захватывается территория наиболее богатая природными ресурсами. Поскольку за время до столкновения ресурсы на территории второго центра заметно истощились, первый распространяет влияние по периферии второго в местах, ещё не до конца освоенных «хозяевами». Постепенно оба центра захватывают каждый свою часть территории. Первый — за счёт мощи центра и более высокого влияния вначале

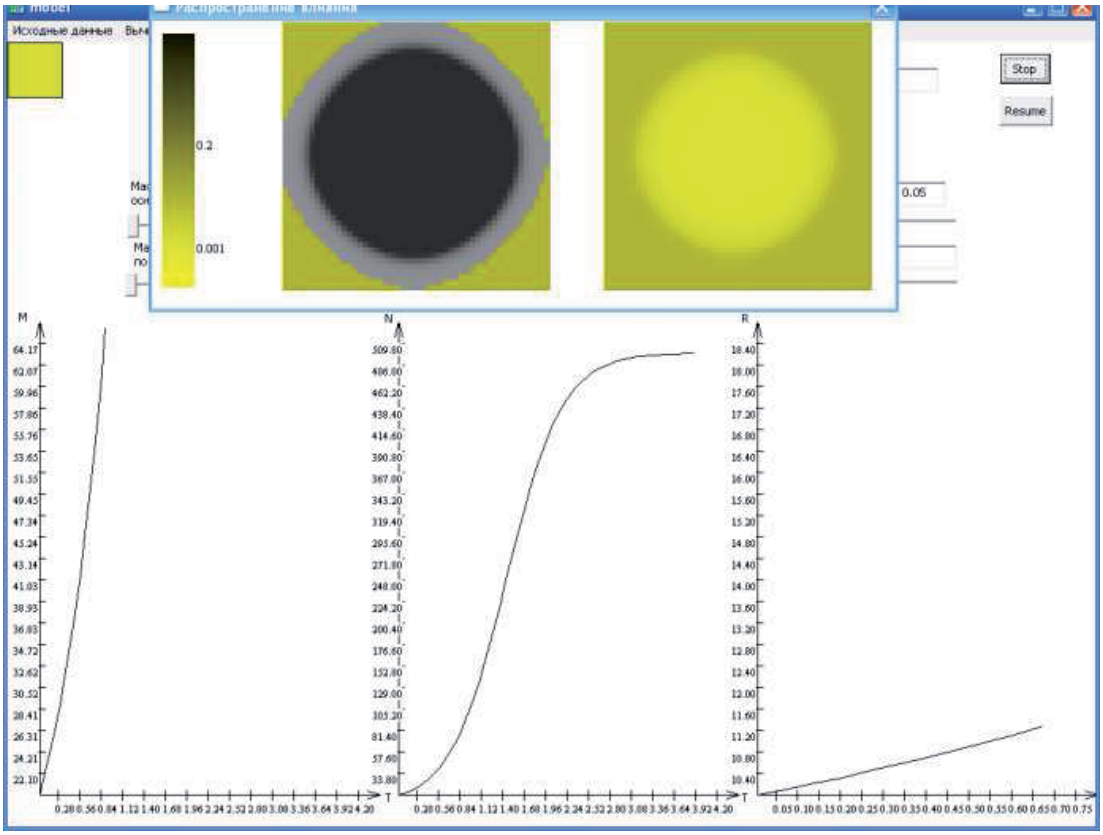

Рис. 4. Моделирование с равномерной картой ресурсов

развития центра, второй — за счёт близкой географической расположенности к территориям, богатым природными ресурсами. При этом первый центр всё-таки отвоёвывает большую часть территории (см. рис. 6).

3. Проверка адекватности и интерпретация работы модели

Строя компьютерную имитацию, мы неминуемо упрощаем и искажаем социальные явления. Построенная компьютерная модель была упрощена при постановке задачи.

Скорость распространения территориального влияния административного центра зависит от мощи центра, от распределения ресурса по территории и от существования или несуществования границ с другими административными центрами. Такая зависимость моделирует реальное развитие при разных территориальных характеристиках.

В модели учитываются пространственные (площадь территорий), политические (границы), частично географические (ресурсы) факторы. Но не учитывается плотность населения, поскольку из уравнения, описывающего изменение мощи административного центра, были исключены коэффициенты, определяющие функциональную зависимость от численности населения данного центра.

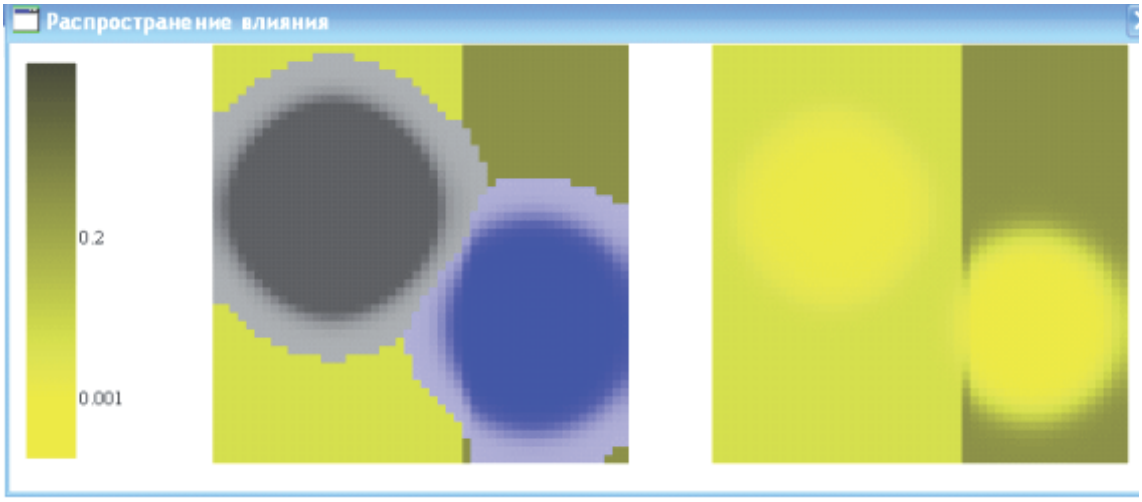

Рис. 5. Моделирование взаимодействия двух административных центров

Рис. 6. Финальная часть моделирования взаимодействия двух административных центров

В модели от численности населения центра зависит только скорость расхода ресурса. И действительно, в реальном мире рост населения вызывает рост потребностей в ресурсах. Но одновременно растущее население способствует распространению его по всё большей территории, расширению административных центров. Недостаток же людских ресурсов приводит к ограничению территориального роста центра.

Увеличение количества добытого ресурса должно приводить к поддержанию и росту населения, а значит, и к распространению территориального влияния, росту мощи центра. Но со временем природные ресурсы истощаются. Понятно, что распространение влияния происходит в направлении увеличения количества ресурса, если рассматривать предположение, что центральная власть империи ставит под контроль более важные ресурсы [3, с. 32]. Вышеупомянутый рост мощи и населения с ростом добытого ресурса в модели пока не отражён.

Возможные функциональные зависимости от добытого ресурса исключены из уравнения численности населения и мощи административного центра для упрощения модели.

В случае столкновения административных центров, попытки нескольких центров захватить влияние на одной территории происходит частичное или полное вытеснение одного из них. При этом территориальный рост центра-победителя замедляется. Такое поведение модели напоминает захват государством-империей заселённых территорий.

Функциональные зависимости от внешней функции центра $C(t)$ из компьютерной модели были исключены. Однако в центре распространения влияния постоянно происходит «подпитка», чтобы, даже если ресурсы в центре истощены, определённая степень влияния в нём сохранялась. Но центр в империи должен не только сохранять, но и оказывать влияние на развитие регионов, допустим, определять, сколько природных ресурсов добывать.

Функциональная зависимость, отражающая влияние центра на количество добываемых природных ресурсов, из уравнения ресурса также исключена. В компьютерной модели не учитывается экономическое и политическое развитие системы, которое в административном центре может влиять на численность населения и на мощь центра. В действительности же при определении исходных данных из реальных исторических источников рост населения может оказаться незначительным, если не принимать во внимание уровень развития экономической системы.

Таким образом, в модели можно наблюдать лишь тенденцию к территориальному расширению центров, захват новых территорий, освоение и добычу ресурсов. Но сохранение или потерю влияния внутри центра при изменении численности населения, его ресурсообеспеченности, политической или экономической ситуации в нём увидеть в данной компьютерной модели нельзя.

Центр существует и развивается только благодаря максимизации объёма контролируемых ресурсов. Влияние в местах, где ресурсы истощены, поддерживается в модели искусственно (задана возрастающая функция мощи центра). Количество добытого ресурса центром не регулируется. Нельзя увидеть сохранение или потерю влияния в зависимости от численности населения, уровня развития экономической и политической системы.

Поэтому для большей адекватности модели реальному процессу развития административных центров необходимо определить и добавить в модель функциональные зависимости от численности населения, добытого ресурса, уровня развития экономической и политической системы, внешней функции центра туда, откуда они были исключены.

Тогда, если проводить компьютерное моделирование поведения нескольких центров, можно будет увидеть различия (возможно, значительные) в их развитии, в зависимости от всех этих факторов, взятых из реальных исторических источников.

Тем не менее модель соответствует нашим представлениям о территориальном развитии административных центров империи, их столкновении между собой и выборочном освоении природных ресурсов.

Литература

1. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. М.: Лаборатория базовых знаний, 2003. 624 с.
2. Замятина Н. Ю. Модели политического пространства // Полис. 1999. № 4. С. 29–41.
3. Каспэ С. И. Империи: генезис, структура, функции // Полис. 1997. № 5. С. 31–48.
4. Кирьянов Д. В., Кирьянова Е. Н. Вычислительная физика. М.: Полибук Мультимедиа, 2006. 352 с.
5. Коробицын В. В. Математическое моделирование этнических полей: дис.... канд. физ.-мат. наук. Тюмень: Тюмен. гос. ун-т, 2002. 141 с.
6. Лаптев А. А. Подходы к построению пространственной модели развития государства // Шестые апрельские экономические чтения. Омск: Изд-во ОмГПУ, 2001. С. 29–32.
7. Лаптев А. А. Пространственная модель развития государства // Развитие оборонно-промышленного комплекса на современном этапе: материалы научно-технической конференции. Часть 1. Омск: ОмГТУ, 2003. С. 124–126.
8. Марчук Г. И. Методы вычислительной математики. М.: Наука, 1977. 456 с.
9. Математическое моделирование социальных систем / А. К. Гуц [и др.]. Омск: Ом. гос. ун-т, 2000. 256 с.
10. Мудров А. Е. Численные методы для ПЭВМ на языках Паскаль, Фортран и Бейсик. Томск: Раско, 1991. 272 с.

Математические структуры и моделирование
2011, вып. 22, с. 24–27

УДК 519.688:534.4

АЛГОРИТМ ВЫДЕЛЕНИЯ ОСНОВНОГО ТОНА И ДЕТЕКТИРОВАНИЯ ТОН/НЕ ТОН ПО МИНИМУМАМ РАЗНОСТНОЙ ФУНКЦИИ НА УЧАСТКЕ МИНИМАЛЬНОГО ПЕРИОДА

Е. А. Первушин, Д. Н. Лавров

Описывается алгоритм нахождения мгновенных значений периодов основного тона речевого сигнала, использующий кратковременную функцию среднего значения разности. Решение о наличии тона принимается при сравнении значений минимумов разностной функции. Предлагается альтернативный выбор начала периода.

Процедура выделения основного тона является одной из важнейших задач в области анализа речи. Выделение основного тона используется в вокодерах, системах синтеза речи, системах распознавания по голосу и других приложениях. В речевом сигнале период основного тона соответствует периоду колебаний голосовых связок и является одной из основных характеристик источника возбуждения голосового тракта.

Существует ряд алгоритмов, каждый из которых имеет свои преимущества и недостатки. Пиковые алгоритмы выделения основного тона используют амплитудно-временные характеристики сигнала, выделяя точки локальных максимумов и/или минимумов речевого сигнала и используя их для определения периода. Данные методы чувствительны к появлению ложных максимумов.

Суть спектрального метода заключается в построении спектра сигнала и нахождении максимума в пределах допустимых частот. При этом для вычисления требуется относительно большое количество операций.

Алгоритмы, использующие автокорреляционную или кратковременную функцию среднего значения разности (КФСР) [4], вычисляют некоторую интегральную для заданного интервала характеристику и по ней оценивают значение периода. При этом вычисленное значение при соответствующем нормировании является также мерой вокализованности. Алгоритмы данного класса обладают приемлемым сочетанием простоты и точности, однако чувствительны к изменениям формы речевого сигнала. Поэтому в данной работе предлагается модификация алгоритма, основанного на вычислении КФСР. Выбор между

Омский государственный университет им. Ф. М. Достоевского.
E-mail: pervushin_evgen@mail.ru

автокорреляционной функцией и КФСР основан на более простом вычислении последней. Более детальное сравнение этих функций можно найти в [2].

В общем виде КФСР (иногда эту функцию называют разностной или сдвиговой) определяется как

$$S_n(h) = \sum_{m=-\infty}^{\infty} |s(n+m)w_1(m) - s(n+m-h)w_2(m-h)|,$$

где $s(t)$ — функция сигнала; w_1, w_2 — оконные функции. Очевидно, что если участок $s(t)$, попадающий в окно, имеет квазипериодический характер с периодом T, то функция S_n будет иметь ярко выраженные минимумы при $h = T, 2T, \ldots$ Условие наличия такого минимума будет использоваться для определения тон/не тон, т. е. является ли участок вокализованным или невокализованным. Пример вокализованных и невокализованных участков речи, а также соответствующих им функций разности, приведён на рис. 1.

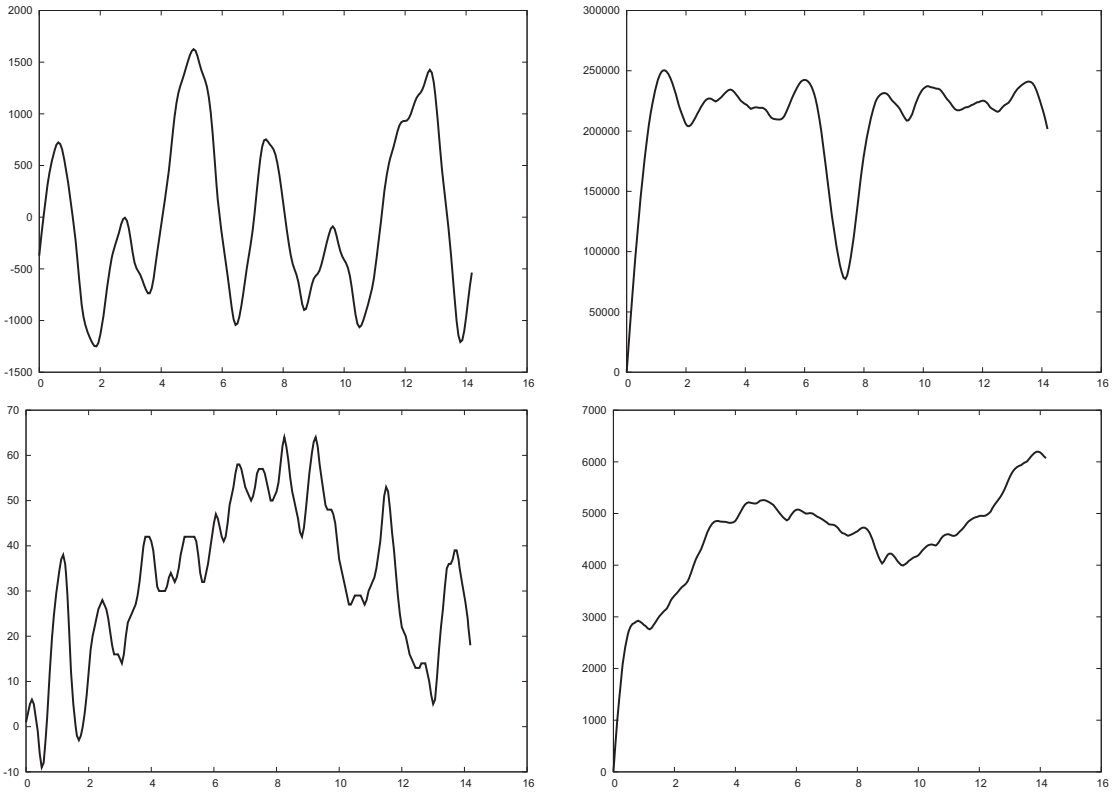

Рис. 1. Пример участков сигналов и их разностных функций: вокализованный участок (сверху), невокализованный (снизу)

Для работы алгоритма потребуется зафиксировать несколько констант:

T_{max} — максимальное значение периода, выраженное в отсчётах;

(P_1, P_2) — интервал, использующийся для проверки на наличие тона;

A_1 — коэффициент для задания порога на наличие тона;

A_2 — коэффициент для задания порога, использующийся в проверке, не найден ли период, кратный основному.

На каждой итерации алгоритма исследуется часть сигнала, попадающая в прямоугольное окно размера T_{max}

$$w_1(t) = w_2(t) = \begin{cases} 1, & 0 \le t \le T_{max} \\ 0, & \text{в противном случае} \end{cases}$$

1. Пусть $s_i,\ i = 1, \ldots, T_{max}$ — отсчёты сигнала, попадающие в окно на данной итерации.
2. Вычисляются значения разностей

$$S_h = \sum_{i=1}^{T_{max}} |s_i - s_{i+h}|,\ h = P_1, \ldots, T_{max}.$$

3. Вычисляется наиболее вероятное значение периода, выраженное в отсчётах

$$T = \arg \min_{h \in (P_1, T_{max})} S_h.$$

4. Если $T < P_2$, то полагаем, что на данном участке нет основного тона; конец итерации.
5. Если существует i такой, что

$$S_i \le A_1 S_T,\ i \in (P_1, P_2),$$

также полагаем, что на данном участке нет основного тона, и переходим к следующей итерации.
6. Обозначим $S = S_T$.
7. Находим минимум среди значений $S_{P_2}, \ldots, S_{T-P_1}$. Пусть T_1 — точка, в которой достигается минимум. Если $S_{T_1} \le A_2 S$, то положим $T = T_1$ и вернёмся к началу шага 7.
8. Считаем полученное значение T периодом основного тона.

В случае нахождения периода, окно сдвигается на величину T, в противном случае — на величину $T_{max}/2$.

Константы, необходимые для работы алгоритма, определим следующим образом. Положим, что алгоритм должен искать периоды основного тона, соответствующие диапазону частот F_{0min} — F_{0max} Гц, тогда $T_{max} = 1/F_{0min}rate$, где $rate$ — частота дискретизации сигнала. Определим теперь $T_{min} = 1/F_{0max}rate$ — минимальное значение периода основного тона. Положим $P_1 = 1/3T_{min}$, $P_2 = 2/3T_{min}$, тогда (P_1, P_2) — подынтервал интервала $(0, T_{min})$, в котором маловероятно появление минимума значений S_h. При фиксированных значениях P_1 и P_2 настраиваются параметры A_1, A_2 для удовлетворения целей приложения, в котором используется данный алгоритм.

Так, например, для системы идентификации дикторов, предложенной в [3], алгоритм выделения основного тона должен быть настроен таким образом, чтобы, с одной стороны, он не выделял периоды, на которых нельзя с уверенностью утверждать наличие тона, и с другой стороны, должно быть выделено достаточное количество периодов для представления шаблона диктора. Эффективность

выбранных значений параметров в данном приложении оценивается по итоговому проценту верных идентификаций при данной базе. При заданных значениях $F_{0min} = 70$ Гц, $F_{0max} = 450$ Гц в результате экспериментов над тестовой базой были выбраны значения порогов $A_1 = 1,3$ и $A_2 = 1,15$.

Помимо определения мгновенных значений периодов основного тона, часто требуется указать начало периода, например, для системы синтеза речи. Распространёнными являются следующие два подхода. Один из них в качестве начала периода определяет точку максимального значения сигнала в пределах найденного периода. Такой подход проще в реализации и, возможно, способен на более точное определение начала периода ввиду определённой выраженности максимума речевого сигнала.

Однако более часто, особенно в системах синтеза (см., например, [1]), используется другой подход, который определяет в качестве начала периода точку перехода сигнала через ноль слева от точки максимума. В своей работе автор использует и другие подходы, при которых сначала находится точка максимума, а затем определяется начало периода, отстоящее слева от точки максимума на расстояние, фиксированное по времени либо зависящее от длины найденного периода.

Предложенный в данной работе алгоритм выделения основного тона, принятия решения тон/не тон и разметки на периоды использовался в системе распознавания дикторов. Алгоритм не требует сложных вычислений и многочисленных настроек и может быть применён в системах анализа и синтеза речи, распознавании речи, распознавании дикторов и других приложениях.

Литература

1. Бабкин А. В. Автоматический синтез речи — проблемы и методы генерации речевого сигнала // Труды Международного семинара по компьютерной лингвистике и её приложениям «Диалог'98». М., 1998.
2. Баронин С. П. Автокорреляционный метод выделения основного тона речи. Пятьдесят лет спустя // Речевые технологии. 2008. Вып. 2. С. 3–12.
3. Первушин Е. А., Лавров Д. Н. Система идентификации диктора на основе выделения информативных участков речевого сигнала // Материалы II межвузовской научно-практической конференции ОмГТУ «Информационные технологии и автоматизация управления». 2010. С. 188–189.
4. Рабинер Л. Р., Шафер Р. В. Цифровая обработка речевых сигналов: пер. с англ. М., 1981. 496 с.

Математические структуры и моделирование
2011, вып. 22, с. 28–32

УДК 519.1

МОДЕЛИРОВАНИЕ ДЕМОГРАФИЧЕСКОЙ СИТУАЦИИ В ОМСКОЙ ОБЛАСТИ

А. К. Ракша

Освещается демографическая ситуация в Омской области за последние 40 лет. Строятся математические модели, отражающие зависимость рождаемости и смертности населения от времени.

Любое демографическое исследование начинается со сбора данных. Рассмотрим динамику смертности и рождаемости в Омской области с 1970 по 2009 гг. [1–7]. Исходные данные представлены в виде двух таблиц (см. табл. 1 и 2).

Таблица 1

Показатели рождаемости в Омской области

Год	Рождаемость (чел.)	Год	Рождаемость (чел.)	Год	Рождаемость (чел.)	Год	Рождаемость (чел.)	Год	Рождаемость (чел.)
1970	29 012	1978	36 883	1986	42 146	1994	22 935	2002	20 406
1971	30 611	1970	37 506	1987	40 669	1995	22 313	2003	22 363
1972	32 030	1980	37 800	1988	37 862	1996	21 457	2004	21 928
1973	32 733	1981	39 145	1989	34 387	1997	19 861	2005	21 282
1974	34 170	1982	42 722	1990	32 191	1998	19 960	2006	21 424
1975	34 950	1983	42 895	1991	29 916	1999	17 756	2007	23 627
1976	36 066	1984	41 983	1992	26 538	2000	18 363	2008	25 050
1977	36 654	1985	40 876	1993	23 324	2001	18 235	2009	25 708

Соответствующие этим таблицам графики рождаемости и смертности населения даны на рис. 1 и 2 соответственно. Исходя из рис. 1, можно предположить, что рождаемость в Омской области отображается полиномиальной функцией. Проверим это предположение.

Обозначим через $R(t)$ функцию, зависящую от времени t и отражающую динамику рождаемости, через $S(t)$ — функцию, зависящую от времени t и отражающую динамику смертности.

Построим полиномиальную функцию шестой степени, зависящую от времени t, подобрав коэффициенты таким образом, чтобы значения производной этой функции были близки показателям рождаемости в Омской области.

Омский государственный университет им. Ф. М. Достоевского.
E-mail: ale2007xa@mail.ru

Рис. 1. Иллюстрация рождаемости в Омской области

Таблица 2

Показатели смертности в Омской области

Год	Смертность (чел.)	Год	Смертность (чел.)	Год	Смертность (чел.)	Год	Смертность (чел.)	Год	Смертность (чел.)
1970	14 105	1978	17 669	1986	19 039	1994	28 126	2002	30 401
1971	14 264	1970	18 664	1987	19 114	1995	26 747	2003	30 796
1972	14 846	1980	19 507	1988	19 536	1996	26 640	2004	30 557
1973	15 133	1981	18 898	1989	20 101	1997	26 600	2005	31 686
1974	15 329	1982	18 934	1990	20 110	1998	25 061	2006	30 159
1975	16 481	1983	19 733	1991	21 321	1999	27 951	2007	29 578
1976	16 849	1984	20 927	1992	22 013	2000	28 713	2008	28 819
1977	17 238	1985	20 344	1993	25 797	2001	28 539	2009	27 352

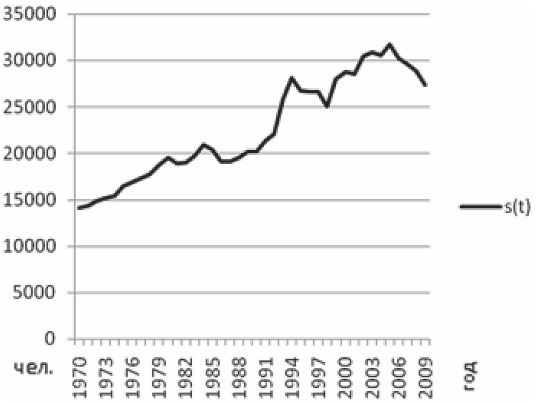

Рис. 2. Иллюстрация смертности в Омской области

После тщательного перебора коэффициентов получаем следующее уравнение:

$$R(t) = 0,0188t^5 - 1,08t^4 + 19,22t^3 - 192t^2 + 2100t + r, \qquad (1)$$

где r — рождаемость в начальный момент времени (для данного исследования 1970 г.); t — период времени; $t = 1, 2, 3, \ldots$

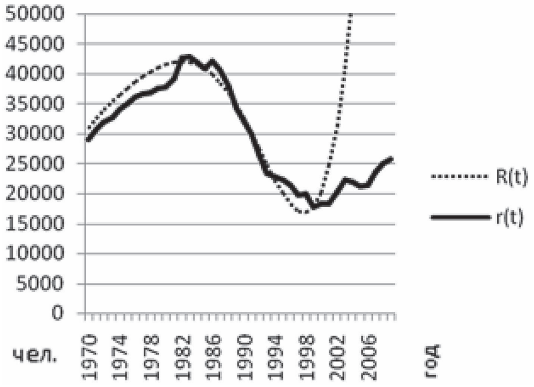

Рис. 3. Иллюстрация полученной полиномиальной функции (1)

Сравнение графиков $R(t)$ и $r(t)$ показывает (см. рис. 3), что на отрезке от 1970-го до 2000 г. значения рождаемости почти совпадают, но после 2000 г. полиномиальная функция $R(t)$ большими темпами устремляется к бесконечности, что противоречит поведению функции $r(t)$. Следовательно, модель $R(t)$ не является удовлетворительной. Наше предположение не подтвердилось.

Построим другую модель для показателей рождаемости в Омской области. Используем тригонометрические функции. Исходя из рис. 1 видно, что график рождаемости по своей форме напоминает функцию синуса. Возьмём начальное значение r_0 и прибавим к нему $h\sin(zt)$, т. е.

$$K(t) = r_0 + h\sin(zt), \tag{2}$$

где r_0 — рождаемость в начальный момент времени; t — период времени; h и z — коэффициенты, найденные путём подбора. При подборе коэффициентов мы пришли к выводу, что для большей схожести исходных значений рождаемости с функцией, которую мы хотим построить, необходимо в эту функцию добавить значение косинуса, т. е.

$$K(t) = r_0 + h\sin(zt)\cos(yt), \tag{3}$$

где y — коэффициент, найденный путём подбора. После подбора коэффициентов найденная нами функция примет вид:

$$K(t) = r_0 + 12510\sin(0,1494t)\cos(0,0122t). \tag{4}$$

Сравнение графиков функций $r(t)$ и $K(t)$ показывает (рис. 4), что $K(t)$ неплохо аппроксимирует $r(t)$. Если сравнивать по табличным данным, а не графически, то данные также окажутся примерно идентичными. Отсюда следует вывод, что данное уравнение отражает демографическую ситуацию в Омской области.

Найдём теперь функцию, характеризующую смертность в Омской области. Из рис. 2 видно, что смертность на протяжении многих лет с каждым годом всё

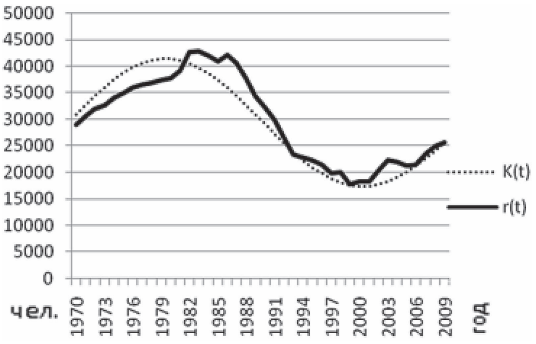

Рис. 4. Иллюстрация полученного уравнения (4), отражающего рождаемость в Омской области

увеличивалась, но в последние годы уменьшается. Следовательно, чем больше период времени, тем выше смертность, поэтому

$$S(t) = s_0 + vt, \tag{5}$$

где s — начальный уровень смертности; v — коэффициент, найденный подбором. Чтобы отобразить характер волнообразности уровня смертности, к (5) прибавим функцию синуса от показателя времени:

$$S(t) = s_0 + vt + q\sin(mt), \tag{6}$$

где m и q — также коэффициенты, найденные подбором. После перебора коэффициентов получим:

$$S(t) = s_0 + 427t + 1000\sin(1370t). \tag{7}$$

Сравнение данных по смертности с полученной моделью — графиком функции $S(t)$ — показывает неплохое их совпадение.

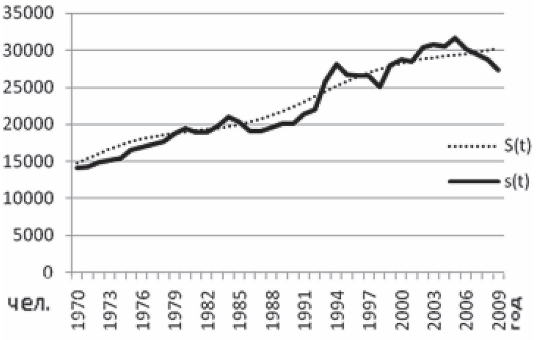

Рис. 5. Иллюстрация полученного уравнения (7), отражающего смертность в Омской области

Если сравнивать табличные данные, а не графические, то они также окажутся примерно одинаковыми. Значит, можно сделать вывод, что данная функция

$S(t)$ характеризует уровень смертности в Омской области. Отсюда следует вывод, что динамику рождаемости и смертности в Омской области за последние 40 лет можно отразить с помощью полученных моделей (4) и (7), содержащих тригонометрические функции.

Литература

1. Демографический ежегодник: стат. сб. / Федеральная служба гос. статистики, территориальный орган Федеральной службы гос. статистики по Омской области. Омск, 2006. 98 с.
2. Демографический ежегодник: стат. сб. / Федеральная служба гос. статистики, территориальный орган Федеральной службы гос. статистики по Омской области. Омск, 2007. 89 с.
3. Демографический ежегодник: стат. сб. / Федеральная служба гос. статистики, территориальный орган Федеральной службы гос. статистики по Омской области. Омск, 2008. 89 с.
4. Омский областной статистический ежегодник [2000 год]: в 2 ч. / Ом. обл. ком. гос. статистики. Омск, 2001. Ч. 1. 235 с.; Ч. 2. 419 с.
5. Омский областной статистический ежегодник [2001 год]: в 2 ч. / Ом. обл. ком. гос. статистики. Омск, 2002. Ч. 1. 232 с.; Ч. 2. 377 с.
6. Омский областной статистический ежегодник [1990, 1995, 1999–2002 гг.]: в 2 ч. / Ом. обл. ком. гос. статистики. Омск, 2003. Ч. 1. 238 с.; Ч. 2. 353 с.
7. Омский областной статистический ежегодник «2009»: крат. стат. сб. / Омскстат. Омск, 2009. 39 с.

Математические структуры и моделирование
2011, вып. 22, с. 33–37

УДК 539.1.01

ГЕОМЕТРОГИДРОДИНАМИКА: ПОПЫТКА ГЕОМЕТРИЧЕСКОЙ ИНТЕРПРЕТАЦИИ ЭЛЕКТРОМАГНИТНОГО ПОЛЯ И КВАНТОВОЙ МЕХАНИКИ

С. А. Сошников

Предлагается простая и наглядная интерпретация уравнений квантовой механики и электромагнитного взаимодействия на основе представлений о динамике искривлённого пространства.

Введение

В 1935 г. Эйнштейн и Розен в работе «Проблема частиц в ОТО» [2] предложили рассматривать соединение «мостом» двух конгруэнтных асимптотически плоских «листов» (решение Шварцшильда) как незаряженную элементарную частицу. В дальнейшем такие «мосты» послужили прототипом известных объектов типа *wormholes*, «кротовые норы», червоточины, горловины. В той же работе для описания заряженных частиц авторами была предложена метрика в виде:

$$ds^2 = \left(1 - \frac{r_g}{r} - \frac{a^2}{r^2}\right) c^2 dt^2 - \frac{dr^2}{1 - \frac{r_g}{r} - \frac{a^2}{r^2}} - r^2(d\theta^2 + sin(\theta)^2 d\varphi^2). \qquad (1)$$

По мнению авторов, такая метрика могла быть положена в основу общей релятивистской теории вещества и квантовой теории. Однако для получения такой метрики потребовался тензор энергии–импульса электромагнитного поля с обратным знаком, т. е., по сути дела, с мнимым зарядом.

1. Основная гипотеза

Предлагается следующая гипотеза для обоснования метрики Эйнштейна–Розена (1):

Предполагается, что само Пространство обладает свойствами инерции и текучести (т. е. его поведение должно быть аналогично поведению идеальной

ООО «Омсктехуглерод» (г. Омск).
E-mail: jin@carbonblack.ru

несжимаемой жидкости, в которой отсутствует давление), при этом плотность его является постоянной и отрицательной (по знаку) величиной.

Пространство, протекающее через «мост» в том или ином направлении, способно создавать дополнительные взаимодействия между «мостами»-частицами, на порядки по величине отличающиеся от гравитационного и полностью аналогичные в частности взаимодействию при помощи классических полей. Кроме того, сам объект-частица получает при этом дополнительные свойства и особенности в поведении.

2. Электрическое взаимодействие

Для обоснования электрического взаимодействия (закон Кулона) можно провести аналогию между точечными источниками (стоками) в гидродинамике идеальной жидкости и точечными электрическими зарядами. Для последних силовые линии напряжённости электрического поля — точная картинка линий тока жидкости вблизи источника. Скорость движения жидкости вблизи источника обратно пропорциональна квадрату расстояния до его центра — так же ведёт себя напряжённость электрического поля точечного заряда. Для нескольких источников и стоков поле скоростей подчиняется принципу суперпозиции — аналогично ведёт себя электрическое поле системы точечных зарядов. Кроме того, сила взаимодействия источников будет кулоновской, если расстояние между ними намного превосходит размеры самих источников.

В нашем случае «мост», через который протекает Пространство, можно со стороны одного «листа» считать источником, со стороны второго — стоком. Однако следует иметь в виду, что в гидродинамике (и это экспериментально подтверждённый факт [1]) два источника притягиваются. Это происходит из-за особенностей потоков импульса жидкости, исходящей от источников. Пространство же обладает отрицательной плотностью, что при тех же условиях приводит к их отталкиванию.

3. Магнитное взаимодействие

Для наглядности рассмотрим процесс нерелятивистского взаимодействия двух заряженных движущихся частиц. Первая частица создаёт вокруг себя электрическое $\overrightarrow{E}$ и магнитное поле $\overrightarrow{B} = \frac{1}{c^2}[\overrightarrow{V_1} \times \overrightarrow{E}]$. Соответственно, сила, действующая на вторую частицу, может быть записана в виде:

$$\overrightarrow{F} = q(\overrightarrow{E} + [\overrightarrow{V_2} \times \overrightarrow{B}]) = q\left(\overrightarrow{E} + \frac{1}{c^2}[\overrightarrow{V_2} \times [\overrightarrow{V_1} \times \overrightarrow{E}]]\right) =$$
$$= q\left(\overrightarrow{E}\left(1 - \frac{(\overrightarrow{V_1}\overrightarrow{V_2})}{c^2}\right) + \overrightarrow{V_1}\frac{(\overrightarrow{V_2}\overrightarrow{E})}{c^2}\right).$$

В итоге мы получаем, что с учётом движения частиц несколько меняется «электрическая» составляющая взаимодействия и появляется сила, действующая

в направлении движения первой частицы. Такая добавка представляется естественной, если принять изложенную гипотезу.

Из сказанного следует, что поле скоростей движущегося Пространства имеет не три, а шесть степеней свободы. Три из них отвечают за прохождение через «мосты», а три — за движение самих «мостов»-частиц. В этом случае полю локальных скоростей Пространства следует сопоставить вектор Римана–Зильберштейна [3]:

$$\overrightarrow{F} = \overrightarrow{E} + \frac{i}{\sqrt{\varepsilon_0 \mu_0}} \overrightarrow{B}$$

Соответственно, если такое движение описывать на языке потенциалов, то их оказывается четыре. В итоге лучший претендент на описание движения Пространства — лагранжиан электромагнитного поля, взятый с обратным знаком (напомним, что плотность постоянна и отрицательна).

4. Квантовая механика

Пусть мы имеем дело с гравитирующим телом, которое издали можно считать сферически симметричным. Тогда элемент пространственной радиальной длины вдали от него можно определить формулой:

$$dl \approx \left(1 - \frac{\Phi}{c^2}\right) dr.$$

Здесь Φ — потенциал гравитационного поля. Он и определяет искривление пространства. Если это отклонение интерпретировать как «неопределённость» радиальной координаты, то можно построить выражение для импульса этой неопределённости, который будет вносить некоторую поправку в выражение для радиальной длины. Учесть это можно с помощью замены:

$$\Phi \to \Phi' = \Phi + \Delta\Phi.$$

Величина $\Delta\Phi$ представляет собой добавку, связанную с градиентом потенциала (силой) и с параметром самой задачи — её характерным гравитационным радиусом:

$$\Delta\Phi = -\frac{\partial\Phi}{\partial r}\frac{a^2}{R_g}.$$

Поскольку $R_g = -\frac{2\Phi}{c^2} r$, тогда

$$\Delta\Phi = \Gamma_r \frac{a^2 c^2}{\hbar r}.$$

Постоянная Планка здесь введена искусственно и использовано обозначение:

$$\Gamma_r = -\frac{\hbar}{2\Phi}\frac{\partial\Phi}{\partial r}. \tag{2}$$

Величину a^2 определим как произведение гравитационного и комптоновского радиусов:

$$a^2 = \frac{2Gm}{c^2}\frac{\hbar}{mc} = \frac{2G\hbar}{c^3}.$$

Произведённая модификация потенциала приводит и к модификации гравитационного радиуса:

$$r_g = -\frac{2\Phi'}{c^2}r = \frac{2Gm}{c^2}\left(1 + \frac{\Gamma_r}{mc}\right).$$

Такой переход по сути и есть переход к метрике Эйнштейна–Розена (1). Величина Γ_r (назовём её геометрическим импульсом) своим происхождением, очевидно, обязана энергии движущегося Пространства. Из этого, кстати, следует и «происхождение» постоянной Планка. Она связана с плотностью Пространства:

$$\rho_0 \sim \frac{c^5}{\hbar G^2}.$$

Теперь тройке фундаментальных физических констант $c, G, \hbar$ можно противопоставить другую тройку констант — c, G, ρ_0.

Рассмотрим релятивистское уравнение для скалярной частицы. Его классическим пределом является уравнение Гамильтона–Якоби. Последнее, в свою очередь (хотя и включает действие), может быть получено из вариационного принципа, если использовать лагранжиан в виде:

$$L = \rho\left(\eta^{\mu\nu}\frac{\partial S}{\partial x^\mu}\frac{\partial S}{\partial x^\nu} - m^2c^2\right).$$

Здесь ρ можно интерпретировать как обычную плотность вероятности, а можно о ней говорить как о функции «представления» частицы, т. е. как «представлена» частица в той или иной области Пространства своими физическими характеристиками. В частности, она может быть «представлена» своими геометрическими импульсами:

$$\Gamma_\mu = -\frac{\hbar}{2\rho}\frac{\partial\rho}{\partial x^\mu}. \tag{3}$$

Если теперь функцию Лагранжа (используя стандартное обозначение для 4-импульса $P_\mu = \frac{\partial S}{\partial x^\mu}$) записать в виде:

$$L = \rho(\eta^{\mu\nu}\Gamma_\mu\Gamma_\nu + \eta^{\mu\nu}P_\mu P_\nu - m^2c^2), \tag{4}$$

то становится очевидным, что речь на самом деле идёт о минимизации разности между суммами квадратов всех видов импульсов частицы и квадрата её собственного импульса mc. Последнее выражение и есть, очевидно, лагранжиан для уравнения скалярной частицы. Если воспользоваться стандартным представлением волновой функции

$$\Psi = \sqrt{\rho}\exp\left(i\frac{S}{\hbar}\right),$$

то лагранжиан (4) принимает вид:

$$L = \hbar^2\eta^{\mu\nu}\frac{\partial\Psi^*}{\partial x^\mu}\frac{\partial\Psi}{\partial x^\nu} - m^2c^2\Psi^*\Psi.$$

Выводы

Если принять предложенную модель, то наряду с достаточно простой интерпретацией электромагнитных и квантово-механических свойств частиц можно также говорить и о рамках применимости тех или иных теорий. Однако наиболее интересным представляется то обстоятельство, что на достаточно малых расстояниях ($r < \frac{a^2}{r_g}$) взаимодействие между частицами может приобрести новые характеристики, что даст возможность интерпретировать и описать другие виды взаимодействий.

Литература

1. Станюкович К. П. Взаимодействие двух тел, «излучающих» потоки газа // Докл. АН СССР. 1958. № 4. С. 119.
2. Einstein A., Rosen N. The Particle Problem in the General Theory of Relativity // Phys. Rev. 1935. Vol. 48. P. 73–77.
3. Silberstein L. Electromagnetische Grundgleichungen in bivectorieller Behandlung // Annalen der Physik. 1907. Vol. 22. P. 579.

Математические структуры и моделирование
2011, вып. 22, с. 38–47

УДК 004.38:004.94

ФИЗИЧЕСКИЕ ОСНОВЫ И ПРОБЛЕМЫ ТЕХНИЧЕСКОЙ РЕАЛИЗАЦИИ КВАНТОВОГО КОМПЬЮТЕРА

Т. В. Вахний, А. К. Гуц

Даётся обзор различных подходов к созданию квантового компьютера. Обсуждаются основные проблемы технической реализации и перспективы каждого из подходов. Обращено внимание на исследования поведения квантовых компьютеров с помощью симуляторов, базирующихся на классических компьютерах.

Введение

Развитие технологий высокого уровня контроля над объектами наноразмеров могут открыть новые возможности по созданию не только более миниатюрных и быстродействующих микропроцессоров, но и принципиально иных способов вычислений, которые не могут быть реализованы в нынешних классических компьютерах. Компьютеры, существенно использующие в своей работе квантовые эффекты путём выполнения квантовых алгоритмов, получили название *квантовых* [1, 3, 6–8].

Перспективы проведения вычислений на квантовых компьютерах обычно связывают с ожидаемым экспоненциальным уменьшением времени решения сложных задач [1, 3, 5–8]. К таким задачам можно отнести разложение больших чисел на множители, теоретико-числовые задачи, связанные с абелевыми группами, поиск нужной записи в неупорядоченной базе данных и другие. Задачи такого типа не могут быть решены на классических компьютерах за время, полиномиально зависящее от числа битов, используемых для представления этих задач в памяти компьютеров.

Возможно, одним из важных приложений квантовых вычислений окажется моделирование поведения широкого класса многочастичных квантовых систем [6]. Это позволит предсказывать свойства молекул и кристаллов, а также проектировать микроскопические электронные устройства размером в несколько десятков ангстрем. Актуальность решения задач такого рода связана не только с быстрым продвижением современной нанотехнологии всё глубже в область

Омский государственный университет им. Ф. М. Достоевского.
E-mail: vahniytv@mail.ru, guts@omsu.ru

нанометровых масштабов, но и с необходимостью прямого моделирования электронных процессов в приборах наноэлектроники, в том числе и в многокубитовых квантовых устройствах. Кроме того, представляет огромный интерес возможность моделирования физических свойств различных сложных органических молекулярных и биологических систем, искусственных полупроводниковых и магнитных материалов и структур.

1. Основные идеи и принципы, лежащие в основе квантового компьютера

1.1. Элементная база

Наименьшим элементом для хранения информации в квантовом компьютере является *кубит*. В качестве кубитов могут быть использованы различные квантовые двухуровневые системы, в частности [1]:

— ионы или нейтральные атомы с двумя низколежащими колебательными или сверхтонкими уровнями, удерживаемые в силовых ловушках, созданных в вакууме с помощью электрических и магнитных полей, при лазерном охлаждении до микрокельвиновых температур;

— сверхпроводниковые структуры с переходами Джозефсона;

— отдельные электронные и ядерные спины в магнитном поле;

— квантовые точки с двумя электронными орбитальными и спиновыми состояниями;

— определённые состояния квантованного электромагнитного поля в электродинамических резонаторах и фотонных кристаллах.

1.2. Кубиты и запутанные состояния

В отличие от единицы классической информации бита, который принимает только два возможных значения (0 и 1), квантовый бит может находиться в суперпозиции этих состояний [1, 3, 6–8]. Как и бит, кубит допускает два собственных состояния, обозначаемых $|0\rangle$ и $|1\rangle$, но при этом может находиться и в их суперпозиции, т. е. в состоянии $\alpha|0\rangle + \beta|1\rangle$, где α и β — любые комплексные числа, удовлетворяющие условию $|\alpha|^2 + |\beta|^2 = 1$. При любом измерении состояния кубита он случайно переходит в одно из своих собственных состояний $|0\rangle$ или $|1\rangle$. Вероятности перехода в эти состояния равны, соответственно, $|\alpha|^2$ и $|\beta|^2$.

Преимущество работы квантового компьютера над классическим основывается на наличии *запутанных (сцепленных) состояний* между кубитами. Запутанность выражается в том, что при всяком изменении состояния одного из кубитов остальные меняют своё состояние согласованно с ним. Причём это происходит не посредством обычных классических взаимодействий (классических корреляций), ограниченных, например, скоростью света, а посредством *нелокальных квантовых корреляций*, когда изменение сказывается в тот же самый момент времени независимо от расстояния между сцепленными кубитами [3].

Если до измерения, т. е. операции вывода данных, m-разрядный квантовый регистр находится в состоянии

$$\sum_{n_{m-1}=0}^{1}\sum_{n_{m-2}=0}^{1}...\sum_{n_0=0}^{1} c_{n_{m-1}n_{m-2}...n_0}|n_{m-1}n_{m-2}...n_0\rangle, \tag{1}$$

$$c_{n_{m-1}n_{m-2}...n_0} \in \mathbb{C},$$

являющемся суперпозицией базовых состояний $|n_{m-1}\rangle|n_{m-2}\rangle...|n_0\rangle$[1], где каждое $|n_j\rangle = |0\rangle$ или $|1\rangle$, то при невозможности быть представленным в виде

$$|x_1\rangle...|x_m\rangle, \tag{2}$$

где $|x_j\rangle = \alpha_j|0\rangle + \beta_j|1\rangle$ $(j = 1, ..., m)$, состояние (1) называется *сцепленным*.

Совокупность сцепленных между собой кубитов может интерпретироваться как заполненный квантовый регистр. Как и отдельный кубит, квантовый регистр (1) гораздо более информативен, чем классический. Он может находиться не только во всевозможных комбинациях составляющих его битов, но и реализовывать различные тонкие зависимости между ними.

1.3. Квантовый параллелизм

Наличие запутанных состояний между кубитами является основным фактором, отвечающим за *квантовый параллелизм*. Этот эффект не имеет аналога при работе классических компьютеров. Если в классическом компьютере вычисляется единственное выходное значение для одного входного, то в квантовом компьютере вычисляются выходные значения сразу для всех входных состояний. Таким образом, благодаря квантовому параллелизму, квантовый компьютер на каждом такте свой работы преобразует **сразу все** базовые состояния. В результате этого квантовые вычисления являются параллельными, что должно позволить получить значительное увеличение скорости и эффективности вычислений квантовых компьютеров по сравнению с классическими.

2. Варианты исполнения квантовых компьютеров

В настоящее время наиболее широко обсуждаются следующие основные направления в развитии элементной базы квантовых компьютеров [1, 6].

2.1. Квантовый компьютер на ионах в ловушках

Для реализации таких квантовых компьютеров в качестве кубитов используются уровни энергии ионов (основное и возбуждённое состояние ионов соответствуют значениям $|0\rangle$ и $|1\rangle$), захваченных ионными ловушками, создаваемыми в вакууме определённой конфигурацией электрического поля в условиях

[1]Обозначение $|n_{m-1}\rangle|n_{m-2}\rangle...|n_0\rangle$ для состояния m-разрядного регистра — это сокращение записи $|n_{m-1}\rangle \otimes |n_{m-2}\rangle \otimes ... \otimes |n_0\rangle$. Другая используемая запись, причём самая простая, — $|n_{m-1}n_{m-2}...n_0\rangle$.

лазерного охлаждения их до микрокельвиновых температур. Взаимодействие между заряженными ионами в одномерной цепочке этих ловушек осуществляется посредством возбуждения их коллективного движения, а индивидуальное управление ими — с помощью лазеров инфракрасного диапазона.

Преимущество такого подхода состоит в сравнительно простом индивидуальном управлении отдельными кубитами. Основными недостатками этого типа квантовых компьютеров являются [1]:

— необходимость создания сверхнизких температур;

— обеспечение устойчивости состояний ионов в цепочке;

— решение проблемы декогерентизации квантовых состояний, определяемой сильным взаимодействием заряженных частиц с окружением;

— ограниченность возможного числа кубитов значением 40 (реально достигнутое число кубитов в таких системах, по-видимому, не может в ближайшее время быть существенно увеличено).

В результате системы из ионов на ловушках без дальнейшего развития принципов работы кубитов на ионах не могут рассматриваться в качестве аппаратного средства для квантового суперкомпьютера, хотя в качестве модельных структур они несомненно имеют определённые перспективы [1].

2.2. Квантовый ЯМР-компьютер на органической жидкости

Эти квантовые компьютеры базируются на использовании в качестве кубитов спинов ядра (состояниям $|0\rangle$ и $|1\rangle$ могут соответствовать направления спина атомного ядра вверх и вниз) атомов, принадлежащих молекулам органических жидкостей, с косвенным скалярным взаимодействием между ними, и методов ядерного магнитного резонанса (ЯМР) для управления кубитами.

В области квантовых ЯМР-компьютеров на органических жидкостях к настоящему времени достигнуты наибольшие успехи, на этом пути удалось реализовать простейшие квантовые компьютеры [1, 6]. Экспериментально на них были осуществлены алгоритм Гровера поиска данных, квантовое фурье-преобразование, квантовая коррекция ошибок, квантовая телепортация, квантовое моделирование и другие операции.

Квантовый компьютер на органической жидкости имеет ряд значительных преимуществ [1]:

— компьютер может работать при комнатной температуре;

— для управления кубитами и измерения их состояний может быть использована хорошо развитая техника ЯМР;

— физической системой, представляющей кубиты, является макроскопический объем органических практически независимых молекул жидкости, содержащих атомы с ядерными спинами, различающимися по резонансной частоте;

— время декогерентизации квантовых состояний ядерных спинов в жидкости достаточно велико (может составлять несколько секунд).

Основными ограничениями для этого направления являются [1]:

— смешанный характер исходного квантового состояния кубитов при температурах жидкого состояния, что требует разработки специальных методов;

— для жидкостных квантовых ЯМР-компьютеров на сегодняшний день число кубитов не может превышать двух десятков из-за экспоненциального уменьшения интенсивности измеряемого сигнала с ростом числа кубитов (это требует экспоненциального роста чувствительности измеряющего оборудования);

— число различающихся по резонансной частоте ядерных спинов-кубитов в отдельной молекуле не может быть произвольно большим;

— однокубитовые и двукубитовые квантовые операции являются относительно медленными.

Таким образом, этот вариант не может лечь в основу для создания квантового суперкомпьютера, превосходящего возможности современного классического компьютера [1].

2.3. Полупроводниковый квантовый ЯМР-компьютер с индивидуальным обращением к кубитам при низких температурах

В этих квантовых компьютерах роль кубитов играют ядерные спины одинаковых донорных атомов в полупроводниковой структуре, для электрического управления которыми и измерения их состояний должна быть создана структура из затворов нанометрового масштаба. Учитывая достижения современной нанотехнологии, в этом варианте можно создать систему из многих тысяч кубитов [1]. Кроме того, полупроводниковые квантовые ЯМР-компьютеры с индивидуальным обращением к кубитам при работе в условиях низких температур способны решить проблемы инициализации и экспоненциального уменьшения интенсивности сигнала с ростом числа кубитов [1].

Основной проблемой полупроводникового варианта является необходимость измерения состояния отдельного кубита. Для решения этой проблемы предложен ряд способов, однако ни один из них пока ещё не реализован. Другая трудность связана с наличием управляющих затворов, шумовое напряжение на которых является существенным источником декогерентизации. Этот вариант квантового компьютера, несмотря на имеющиеся трудности, несомненно заслуживает дальнейшей разработки [1].

2.4. Твердотельный квантовый ЯМР-компьютер с ансамблевым обращением к кубитам

Является более перспективным, чем предыдущая модель [1]. Появляется возможность существенно упростить управление кубитами и измерение их состояний. Кроме того, при использовании в ансамблевых вариантах принципов квантового клеточного автомата можно в значительной степени упростить и систему управляющих затворов и даже, возможно, отказаться от них совсем. Это позволило бы исключить связанные с ними существенные механизмы декогерентизации.

2.5. Полупроводниковый квантовый ЯМР-компьютер с использованием СВЧ и лазерных импульсов

Полупроводниковый квантовый ЯМР-компьютер, в котором для управления кубитами и измерения их состояний, наряду с электрическими, используются СВЧ и лазерные импульсы, является альтернативой по отношению к модели полупроводникового квантового ЯМР-компьютера с индивидуальным обращением к кубитам. Этот вариант облегчает решение задач, связанных с измерениями состояний отдельных кубитов, но при этом сохраняются недостатки, связанные с наличием системы затворов. Здесь также является перспективным использование ансамблевого подхода [1].

2.6. Квантовые компьютеры на квантовых точках с электронными орбитальными и спиновыми состояниями

Эти компьютеры имеют ряд преимуществ перед квантовыми ЯМР-компьютерами [1]:

— они способны работать при более высоких температурах, чем полупроводниковые ЯМР-квантовые компьютеры;

— имеют значительно более высокие тактовую частоту и величину измеряемого сигнала;

— современная нанотехнология позволяет создавать квантовые структуры с практически неограниченным числом квантовых точек.

Основной трудностью для них является относительно быстрая декогерентизация квантовых состояний, связанная с электрическим зарядом электрона и электрическими методами управления кубитами, для подавления которой нет хорошо разработанных методов. Выходом может быть использование для этого не электрических, а оптических ультраскоростных методов [1].

2.7. Квантовые компьютеры на переходах Джозефсона

В таких компьютерах в качестве кубитов используются зарядовые состояния куперовских пар в квантовых точках, связанных переходами Джозефсона. Перспективность этого направления состоит в возможности создания электронных квантовых устройств высокой степени интеграции на одном кристалле, при этом для управления кубитами не требуются громоздкие лазерные или ЯМР-установки [1].

Сверхпроводниковый вариант квантового компьютера, несмотря на уже имеющиеся достижения в реализации отдельного кубита, имеет ряд трудностей [1]. Они связаны с необходимостью жёсткого контроля за совершенством изготовления туннельных джозефсоновских переходов, за временными характеристиками импульсных воздействий, с использованием для управления отдельными кубитами электрических схем, флуктуации напряжений, которые являются основной причиной декогерентизации. Система большого числа кубитов, связанная с электромагнитным окружением, представляет собой сложную

нелинейную систему, в которой могут проявляться многие нежелательные нелинейные эффекты.

2.8. «Необычные» квантовые компьютеры

Рассмотренные выше варианты квантовых компьютеров относятся к группе обычных квантовых компьютеров. Они предполагают использование достаточно изученных квантовых явлений. В последнее время обсуждаются так называемые необычные квантовые компьютеры [1], в которых в качестве базовых носителей информации — кубитов — рассматриваются частицы, подчиняющиеся фермиевской статистике — фермионы. Это могут быть, например, электроны и дырки в полупроводнике. Состояние, занятое фермионом в таких системах, может представлять состояние $|1\rangle$, а незанятое — состояние $|0\rangle$.

В качестве другого варианта необычных квантовых компьютеров рассматриваются также бозонные компьютеры, использующие свойства бозе-конденсата с нелинейным взаимодействием между бозонами, проявляющиеся, например, в фотонных кристаллах, в системах из нейтральных атомов в оптических ловушках [1].

Интересны квантовые компьютеры, использующие экзотические частицы, не подчиняющиеся ни фермиевской, ни бозоновской статистике. Был предложен компьютер на анионах [1,6] — квазичастицах, отличающихся тем, что волновая функция аниона при обходе другого аниона приобретает произвольную фазу. Примером анионов являются элементарные возбуждения в двухмерном электронном газе в условиях квантового эффекта Холла. Привлекательным свойством анионных квантовых систем является возможность при использовании нелокальной природы аниона обеспечить защиту квантовых операций от влияния случайных помех.

Возможно, в будущем появятся новые идеи для физической реализации квантового суперкомпьютера или комбинации уже известных вариантов, в которых будут одновременно учитываться их преимущества.

3. Наиболее перспективные направления конструирования квантового компьютера

Из всех известных направлений в развитии элементной базы квантовых компьютеров наиболее привлекательными с точки зрения создания суперкомпьютеров в настоящее время представляются следующие [1]:

— полупроводниковые ЯМР-квантовые компьютеры,

— квантовые компьютеры на переходах Джозефсона,

— квантовые компьютеры на квантовых точках.

Указанные перспективные направления конструирования квантового компьютера допускают произвольно большое число кубитов и для них существует множество наработанных приёмов микро- и нанотехнологий создания полупроводниковых и сверхпроводниковых интегральных схем.

Наиболее важными преимуществами обладает твердотельный ЯМР-квантовый компьютер. Основные из них следующие [1]:

— ядерные спины сами по себе являются кубитами;

— при низких температурах они характеризуются очень большими временами релаксации (и, соответственно, временами декогерентизации) по сравнению с электронными спинами;

— технологические структуры нанометрового масштаба в полупроводниковых квантовых ЯМР-компьютерах предназначаются не для создания самих кубитов, как в случае сверхпроводниковых устройств, а лишь для задач управления кубитами и измерения их состояний.

В настоящее время состояние современной высокоточной технологии и технологии высокочистых материалов уже сейчас позволяют экспериментально создавать простейшие варианты квантовых компьютеров. Создание же многокубитовых образцов пока находится в далёкой перспективе. Для преодоления существующих трудностей физической реализации квантового суперкомпьютера потребуется привлечение технологических и схемотехнических достижений современной микро- и наноэлектроники, а также разработка программ математического моделирования физических процессов, и в частности процессов декогерентизации в многокубитовых квантовых системах [1].

4. Общие требования к элементной базе квантового компьютера

Считается, что для реализации полномасштабного квантового компьютера, превосходящего по производительности любой классический компьютер, на каких бы физических принципах он не работал, необходимо выполнение следующих основных требований [1]. Квантовый компьютер должен:

1. Содержать более 10^3 хорошо различаемых кубитов;

2. Обеспечить условия для приведения входного регистра в исходное основное базисное состояние, т. е. возможность процесса инициализации;

3. Обеспечить максимальное подавление эффектов декогерентизации квантовых состояний, обусловленных взаимодействием системы кубитов с окружающей средой, что приводит к разрушению суперпозиций квантовых состояний и может сделать невозможной выполнение квантовых алгоритмов. Время декогерентизации должно существенно превышать (более чем в 10^4 раз) время выполнения основных квантовых операций (времени такта). Для этого система кубитов должна быть достаточно слабо связана с окружением;

4. Обеспечить выполнение за время такта требуемой совокупности квантовых логических операций, определяющей унитарное преобразование [3, 4];

5. Обеспечить с достаточно высокой надёжностью измерение состояния квантовой системы на выходе. Проблема измерения конечного квантового состояния является одной из основных проблем квантовых вычислений.

5. Методы построения симуляторов квантового компьютера

Помимо технологической сложности устройства квантового компьютера перед разработчиками стоит также проблема, заключающаяся в невозможности просчитать поведение квантовой системы при помощи классических компьютеров. Для решения этой проблемы строятся модели квантовых вычислителей [2].

При моделировании статистики системы обычно используется матричное представление операторов и векторов квантовых состояний. Для реализации таких моделей чаще всего используются языки высокого уровня (Delphi, C++) и квантовые подключаемые библиотеки, например QDD [10]. Также создаются специальные языки программного описания квантовых процессов, такие как QASM — квантовый ассемблер, созданный как простое средство описания действия квантовых цепочек, состоящих из кубитов и однокубитных вентилей, или Qgol [9]. Существуют подключаемые библиотеки для классических моделирующих программ типа Matlab и Maple [11], которые содержат набор интуитивно понятных квантовых операторов.

Ещё одним направлением является использование стандартных языков аппаратного описания (*HDL) за их возможность компактно раскрыть структурную и функциональную составляющие архитектуры систем [2]. Здесь же предложена технология ухода от необходимости дополнительно описывать квантовую сцепленность кубитов. Для снижения временных затрат и необходимого объёма памяти при моделировании применяется методика QuIDD, которая рассматривает новый класс матриц и векторов, специально разработанных и используемых для описания состояний и поведения квантовых информационных систем [2].

Заключение

Появление полноценных квантовых компьютеров даст мощный толчок не только развитию вычислительной техники, но и техники передачи информации, организации принципиально новых систем связи типа квантового Интернета и может оказаться началом развития новых, пока ещё неизвестных областей науки и техники. Кроме того, исключительные возможности квантовых суперкомпьютеров, несомненно, будут способствовать более глубокому пониманию физических законов.

Литература

1. Валиев К. А., Кокин А. А. Квантовые компьютеры: надежды и реальность. Ижевск: РХД, 2001. 352 с.
2. Гузик В. Ф., Гушанский С. М. Методы построения симуляторов квантового компьютера // Изв. ТРТУ. 2006. Т. 64. № 9–1. С. 92.

3. Гуц А. К. Основы квантовой кибернетики: учеб. пособие. Омск: КАН, 2008. 204 с.
4. Дуплий С. А., Калашников В. В., Маслов Е. А. Квантовая информация, кубиты и квантовые алгоритмы. 2005. URL: homepages.spa.umn.edu/∼duplij/publications/Duplij_vest_sd-kal-mas.pdf (дата обращения: 30.03.2010).
5. Ежов А. А. Некоторые проблемы квантовой нейротехнологии // Научная сессия МИФИ–2003. V Всероссийская научно-техническая конференция «Нейроинформатика–2003»: лекции по нейроинформатике. Часть 2. М.: МИФИ, 2003. С. 3–79.
6. Китаев А., Шень А., Вялый М. Классические и квантовые вычисления. М.: МЦНМО, ЧеРо, 1999. 192 с.
7. Кокин А. А. Твердотельные ядерные магнито-резонансные (ЯМР) ансамблевые квантовые компьютеры (исследования физических основ и проблем реализации): дис. . . . д-ра физ.-мат. наук. М., 2003. 187 с.
8. Ожигов Ю. И. Квантовые вычисления / ВМиК. М.: МГУ, 2003. 104 с.
9. Baker G. Qgol. A system for simulating quantum computations: Theory, Implementation and Insights. 1996. 62 p. URL: http://ifost.org.au/∼gregb/q-gol/index.html (дата обращения: 30.03.2010).
10. Greve D. QDD: A quantum computer emulation library. URL: http://thegreves.com/david/QDD/qdd.html (дата обращения: 30.03.2010).
11. OpenQUACS — Open-source Quantum Computer Simulation. URL: http://userpages.umbc.edu/∼gcmccub1/quacs/quacs.html (дата обращения: 30.03.2010).

Математические структуры и моделирование
2011, вып. 22, с. 48–54

УДК 534.4

ПОДХОДЫ К ЗАДАЧЕ ИДЕНТИФИКАЦИИ ДИКТОРА

О. А. Вишнякова, Д. Н. Лавров

Представлен обзор систем идентификации по голосу, алгоритмов распознавания дикторов. Проводится анализ информативных признаков, способов построения эталонов и алгоритмов принятия решения.

Введение

Интерес к системам идентификации обусловлен широким кругом практических приложений: проверка прав доступа к различным системам (базам данных, каналам связи, помещениям, устройствам и механизмам, банковским счетам и т. д.), криминалистическая экспертиза.

Преимущества именно голосовой идентификации следуют из характеристик голоса: не отчуждаем от человека; не требует непосредственного контакта; не требует сложных технических устройств. Голос диктора, а как следствие и сам речевой сигнал уникален ввиду специфики физиологического строения его артикуляторного аппарата и специфики его речи. Это обусловливает интерес к нему как биометрическому объекту. Только в некоторых весьма редких случаях указанная уникальность может не иметь места (например, однояйцовые близнецы, воспитанные в одинаковых условиях).

1. Классификация задач определения диктора

Классически задача идентификации выглядит следующим образом: имеется ограниченная и строго контролируемая группа пользователей системы. Системы идентификации диктора выносят решение о том, что диктор принадлежит к выбранной группе (или подгруппе) дикторов, модели которых хранятся в её базе моделей, и указывают конкретного диктора (рис. 1). При этом качество системы характеризуется средней вероятностью правильной идентификации. При такой формулировке задачи исключена ситуация возможного злоумышленника, однако среди возможных применений систем распознавания дикторов ситуации с замкнутыми группами возникают редко [2].

Омский государственный университет им. Ф. М. Достоевского.
E-mail: olga@infotekorg.ru

Рис. 1. Общая схема идентификации диктора

На практике чаще встречается задача верификации, для решения которой строится система, которая выносит решение о том, что диктор, голосовой сигнал которого предъявлен системе в качестве образца, соответствует его модели голоса, хранимой в базе голосовых моделей зарегистрированных пользователей и обозначенной уникальным PIN-кодом, или не соответствует. Диктор в первом случае может быть обозначен как «свой», а во втором — как «чужой» (рис. 2). Качество принимаемого системой решения характеризуется ошибками 1-го и 2-го рода (FRR и FAR) [5].

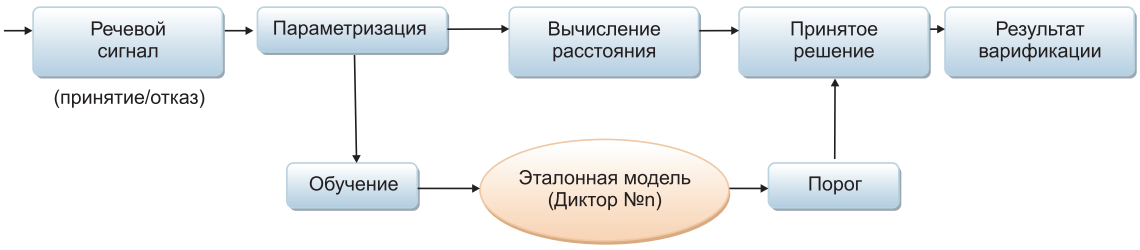

Рис. 2. Общая схема верификации диктора

В общем же случае задача распознавания диктора сводится к открытой идентификации, при которой пользователь не объявляет свою индивидуальность. Система должна сверить поступивший речевой сигнал со всеми речевыми эталонами зарегистрированных пользователей. Таким образом, задача открытой идентификации совпадает с задачей многократной верификации.

В настоящее время используются как текстозависимые, так и текстонезависимые системы распознавания дикторов. Текстозависимая система работает по парольным фразам (статический режим). Обобщённым случаем для этого режима является тексто-подсказанный режим (динамический), когда система в случайном порядке предлагает диктору сказать фразу из заданного набора, причём диктор на этапе обучения ввёл соответствующие фразы в систему. Содержание фраз может выбираться пользователем или системой. Это свойство системы позволяет пользователю периодически менять свой голосовой пароль, обеспечивая ещё большую надёжность верификации. В отличие от предыдущей, текстонезависимая система работает с использованием произвольной речи. Диктору не нужно помнить какую-то определённую парольную фразу. Очевидно, что совпадение лингвистической формы двух сравниваемых речевых сообщений облегчает процесс идентификации. В процессе решения задачи идентификации каждому диктору ставится в соответствие некоторый эталон — набор уникальных признаков — для дальнейшей классификации и принятия решения алгоритмами распознавания.

Таким образом, задача разбивается на три относительно независимые части:

1. Выделение информативных признаков (параметризация речевого сигнала);
2. Процедуры построения эталона для данного диктора;
3. Принятие решения на основе сравнения с эталонами.

2. Выделение информативных признаков

Важнейшим элементом успешного распознавания дикторов является выбор информативных признаков, способных эффективно представлять информацию об особенностях речи конкретного диктора. Требования к ним таковы:

- эффективность представления информации об особенностях речи конкретного диктора;
- простота измерения;
- стабильность во времени;
- частое и естественное появление в речи;
- практическая независимость от акустической среды;
- невосприимчивость к имитации.

Индивидуальные характеристики голоса определяются уникальностью строения артикулярного аппарата человека: строение голосовых связок, степень натяжения голосовых связок, скорость открывания и закрывания голосовой щели, объем и конфигурация речевого тракта.

Так, одним из ключевых признаков является частота основного тона F_0 — частота импульсов голосового источника, возникающая в результате колебания голосовых связок. При этом периодичность колебаний может нарушаться вследствие изменений амплитуды, частоты, фазы колебаний, наличия шума, поэтому под частотой основного тона понимают среднюю оценку на некотором интервале.

Одной из лучших характеристик гласоподобных звуков считаются формантные частоты (форманты), которые являются проявлениями резонансных частот речевого тракта диктора в акустическом сигнале [7]. Таким образом, они являются важнейшим параметром, характеризующим спектр (распределение энергии или амплитуды по частотам) речевого сигнала, которые определяют как концентрацию энергии в ограниченной частотной области. Форманта характеризуется частотой, шириной и амплитудой. За частоту форманты принимают частоту максимальной амплитуды в пределах форманты. Другими словами, форманта — это некоторый амплитудный всплеск на графике спектра, а его частота — частота пика этого всплеска (рис. 3).

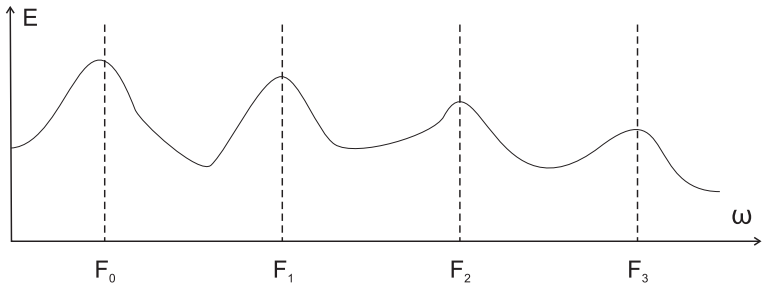

Рис. 3. Форманты речевого сигнала

Следует учитывать, что характерные признаки голоса должны вычисляться на определённых сегментах речевого сигнала. Частота основного тона — на гласоподобных участках; форма речевого тракта, которая характеризуется формантными частотами, — на гласных звуках; скорость артикуляции определяется по длительностям переходных процессов между артикуляторно-акустическими сегментами. Для выделения индивидуальных характеристик голоса целесообразно использовать только гласные и сонорные согласные звуки, хотя есть работы по идентификации на базе шипящих [3]. Таким образом, одной из основных задач является разработка надёжного алгоритма сегментации речевого сигнала и определения типа сегмента [4]. Методы сегментации в свою очередь можно условно разделить на две группы: основанные и не основанные на моделях. Методы, основанные на моделях, в основном используют кодирование по линейному предсказанию (LPC) и меры близости для оценивания спектральных изменений между последовательными кадрами речевого сигнала. При использовании модели LPC вычисляется гауссовская функция правдоподобия для обеих гипотез, и для каждого n-го кадра находится отношение правдоподобия $LR(n)$. Если на определённом кадре n_0 отношение $LR(n_0)$ превышает некий фиксированный порог, детектируется изменение. Методы, не основанные на моделях, использу-

ют различные алгоритмы обработки, такие как параметрическая фильтрация. Эти методы пытаются обнаружить спектральные изменения речевого сигнала, непосредственно рассматривая речевой спектр с использованием нескольких мер спектральных расстояний. Это означает, что если $S_1(f)$ и $S_2(f)$ — спектры двух соседних кадров сигнала, то изменение будет отмечено, если расстояние между ними превысит заданный порог [8].

3. Построение эталона

Для параметризации речевого сигнала с целью построения эталона используют две основные группы представлений — на базе преобразований Фурье и на базе линейного предсказания. Однако есть исследования альтернативных походов, на базе вейвлет-преобразовний, частотных цифровых фильтров, гауссовых смесей.

Одним из способов параметризации с использованием кепстральных характеристик при построении эталона служит модель:

$$M = \{K12, \Delta K12, E, \Delta E\},$$

где $K12$ — двенадцать мел-частотных кепстральных коэффициентов (MFCC); $\Delta K12$ — двенадцать характеристик дельты MFCC; E — энергетическая характеристика; ΔE — дельта-характеристика энергии. Итого: 26 характеристик на эталон [6].

Модель также можно представить в виде набора средних значений энергии вейвлет-коэффициентов для каждого уровня детализации:

$$M = \{W_n, \Delta W_n\},$$

где W_n — значения средней энергии вейвлет-коэффициентов для десяти уровней детализации; ΔW_n — значения среднего квадратического отклонения вейвлет-коэффициентов для десяти уровней детализации; n — число уровней детализации вейвлет-преобразования. Итого: 20 характеристик на каждый эталон.

Также используются комбинации вейвлет и Фурье-анализа превосходящих по итогам экспериментов моделей на базе мел-частотных кепстральных коэффициентов и линейного предсказания [9].

4. Алгоритмы принятия решения

При принятии решения, в простейшем случае, вычисляется вероятность или расстояние от тестовых векторов информативных признаков для образца, поданного на вход системы, до эталонных векторов (моделей дикторов) и сравнивается полученное значение с порогом, часто фиксированным для всех дикторов. Могут быть использованы также механизмы нормализации для повышения устойчивости при наличии таких мешающих факторов, как: вариабельность произнесения парольной фразы одного и того же диктора, настроение диктора,

разные манеры и интонации произнесения, болезнь горла, громкость произнесения (шёпот или громкий голос) и т. д. Наиболее широко для этих целей используется когортный метод и метод мировых моделей.

Также для классификации вводимого речевого образца могут использоваться следующие методы собственно распознавания: HMM (скрытые марковские модели), моделирующие речевой сигнал на основе теоретико-вероятностных схем, DTW (динамическое программирование), базирующийся на евклидовой метрике, ANN (нейронные сети), в основе которых лежит процедура предварительного обучения, байесовский классификатор, FRIS-функции [1].

При верификации осуществляется оценка близости предъявляемого образца к эталону (модели данного диктора) и производится сравнение этой оценки с порогом, который может изменяться, чтобы осуществлять обмен между ошибками FAR и FRR. При равнозначности ошибок ($FAR = FRR = EER/2$) порог фиксирован. Критерий качества связан с ошибками 1-го и 2-го родов при проверке простой гипотезы при простой альтернативе.

При идентификации критерий качества определяется вероятностью отнесения диктора к заданной группе, правильного распознавания диктора из группы, вероятностью перепутать диктора при отнесении его к заданной группе, принять чужака за своего в группе и отождествить с конкретным диктором из группы, что связано с более общей ситуацией проверки гипотезы при сложной альтернативе. Принятие решения производится по минимуму расстояния между предъявляемым образцом и ближайшей моделью из набора моделей голосов дикторов, входящих в заданную группу. Отбор на предмет принадлежности к группе осуществляется путём сравнения указанного расстояния с порогом.

5. Проблемы голосовой идентификации

Основной проблемой для систем идентификации являются изменчивость речевого сигнала, связанная с произношением самого диктора, различия в условиях записи при регистрации пользователей и идентификации, шумы и искажения в каналах связи. Так, при хорошем отношении сигнал/шум (SNR +20 dB) достигнута точность идентификации 98 %, но уже 81 % — при +10 dB, что говорит о необходимости дальнейшего исследования в пользу способов предобработки и помехоустройчивых методов распознавания.

Таким образом, по-прежнему перспективными направлениями исследования остаются:

- поиск новых помехоустойчивых информативных признаков, связанных с характеристиками голосового источника, и формы артикуляционного тракта;
- новые решающие правила, минимизирующие ошибки 1-го и 2-го рода;
- создание полноценной речевой базы для тестирования систем идентификации с большим числом дикторов различного возраста, акцента, особенностями произношения, эмоционального состояния, записанных на различных условиях записи и с разной частотой дискретизации.

Литература

1. Борисова И. А. Алгоритм таксономии FRiS-Tax // Научный вестник НГТУ. 2007. № 3. С. 3–12.
2. Галунов В. И. Верификация и идентификация говорящего. СПб.: СПбГУ, 2002. URL: http://www.auditech.ru/article/ver_obz.doc (дата обращения: 20.10.2010).
3. Криводубский О. А., Федоров Е. Е. Моделирование особенностей речи диктора // Математичні машини і системи. 2008. № 1.
4. Леонов А. С., Макаров К. С., Сорокин В. К., Цыплихин А. К. Кодовая книга для речевых обратных задач // Информационные процессы. 2005. Т. 5. № 2. С. 101–119.
5. Мартынович П. А., Свириденко В. А. Системы верификации и идентификации диктора от SPIRIT Corp. // Доклады на конференции «BIOMETRICS 2003 AIA RUII». М., 2002.
6. Медведев М. С. Фонемная сегментация речевого сигнала с использованием вейвлет-преобразования // Доклады на конференции ИВТ СО РАН. Новосибирск, 2004.
7. Цыплихин А. И. Анализ и автоматическая сегментация речевого сигнала: дис. ... канд. техн. наук. М., 2006. 149 с.
8. Li T. H., Gibson J. D. Speech Analysis and Segmentation by Parametric Filtering // IEEE Transactions on Speech and Audio Processing. 1996. Vol. 4 (3). P. 203–213.
9. Mporas J., Ganchev T. Comparison of speech features on the speech recognition task // Journal of Computer Science. 2007. Vol. 3 (8). P. 608–616.

Математические структуры и моделирование
2011, вып. 22, с. 55–68

УДК 004.4'24

РАЗРАБОТКА СЕМЕЙСТВА ПРОГРАММНЫХ СИСТЕМ В СПЕЦИФИЧЕСКОЙ ПРЕДМЕТНОЙ ОБЛАСТИ

С. В. Гусс

Рассматриваются общие вопросы разработки семейства программных систем в рамках линейки программных продуктов на основе систематического подхода к повторному использованию производственных активов (программных компонентов, специализированных инструментов). Проводится обзор оснований, лежащих в корне данной практики. Обращается внимание на профессиональные качества разработчиков конечных продуктов и требуемый уровень знаний для их эффективной работы.

Введение

Согласно [13, 15, 19, 21], специально созданные или отобранные, адаптированные и настроенные на определённое семейство программных систем, средства разработки в сочетании с определённой профессиональной подготовкой исполнителей проекта способны увеличить производительность процесса создания компьютерных приложений и обеспечить высокое качество разрабатываемого продукта.

К таким средствам относятся модели, каркасы, шаблоны и предметно-ориентированные языки моделирования/программирования, способные связать составные части приложения в рамках общей архитектуры семейства программных систем [2, 5, 19]. В моделях отображаются общие очертания программных единиц. Каркасы определяют характерные для программных единиц повторноиспользуемые абстракции. Предметно-ориентированные языки делают процесс работы с этими абстракциями производительным, а шаблоны повышают эффективность процесса, ограничивая возможность совершения некорректных действий, в то же время указывая верный путь работы с абстракциями. Добавив к этому автоматические программные инструкции, оказывающие помощь разработчикам, непосредственно во время создания приложений, можно объединить всё вместе в специальный пакет, расширяющий возможности интегрированной среды разработки. К примеру, среда Microsoft Visual Studio 2010

Омский государственный университет им. Ф. М. Достоевского.
E-mail: InfoGuss@Gmail.com

имеет в своём арсенале все необходимые для расширения средства, включая специальные шаблоны проектов, есть нечто подобное и в средах IDE Eclipse и Embarcadero RAD Studio. Особое выделение продуктов компании Microsoft вызвано тем, что в рамках одной из её исследовательских инициатив проводятся поиск и развитие методов автоматизации, а в конечном счёте и индустриализации разработки программных продуктов, что должно повысить профессиональный уровень зрелости разработчиков программного обеспечения. Эта инициатива, названная фабриками программного обеспечения [19], рассматривается автором статьи в своём теоретическом и практическом аспекте.

Наряду с общими рекомендациями по составлению пакета вспомогательных средств, куда входят библиотеки программных компонентов, специализированные инструменты, документация и автоматические руководства, описанными в [19, 21, 25], необходимы также конкретные рекомендации по созданию пакетов для конкретной предметной области [15]. В качестве предметной области в представленном в статье проекте программного обеспечения выступают игровые обучающие программные системы с лингвистической составляющей [6–8]. Такие программные системы оптимально подходят, с точки зрения автора (подкреплённой исследованиями учёных в области педагогики и психологии [9]), для учебного процесса.

О явлении компьютерных игр в целом можно узнать из исследований в области культурологии, представленных в работе [14]. Вопрос влияния игр на психологию играющих и обучающихся с помощью них с разных сторон рассмотрен в [9]. Что же касается внутренней (не только технической) составляющей компьютерных игр, то довольно полно информация по данному вопросу представлена в [18].

О необходимости и практичности конкретизации современных методов разработки компьютерных приложений для отдельных предметных областей говорится во множестве работ, часть из которых будет упомянута и коротко рассмотрена в тексте статьи.

Кроме того, также рассматриваются и этические вопросы, касающиеся профессиональной разработки приложений для специальной предметной области. Автор упомянет и об исторических моментах, а также основах, лежащих в затронутых темах. Не упущен из виду и такой аспект профессиональной разработки, как человеческий фактор [13], связанный с психологическими особенностями разработчиков, их профессиональными качествами и набором необходимых им знаний [17]. Данные вопросы рассматриваются в общих чертах, они основываются на существующих сегодня знаниях, тем не менее для тех, кто хочет узнать об этом подробней, приводятся литературные источники.

1. Профессиональная разработка программного обеспечения

В наше время от специалистов в области программной инженерии, непосредственно от профессиональных разработчиков требуются определённые знания о производстве программных систем «в определённой тематической области их

применения» [13]. В [15] говорится, что для эффективного внедрения инноваций инженерии программного обеспечения необходимо выразить их в терминах, «понятных для каждой конкретной подотрасли». Это подразумевает необходимость конкретизации знаний в области разработки приложений для конкретной прикладной области [34]. Согласно [17], от сегодняшних разработчиков требуется больше, чем просто написание программного кода. Необходимо иметь представления обо всём жизненном цикле программного обеспечения [4,11] и в соответствии со своей специализацией уметь пользоваться специальными знаниями. Что самое главное, там также упоминается о том, что «эффективное использование специфических для предметной области методов, средств и компонент в большинстве случаев обеспечивает успешность разработок с использованием программной инженерии». Таким образом, аналитики, архитекторы, проектировщики и разработчики помимо знаний из инженерии программного обеспечения должны обладать хотя бы минимальной информацией из специальной предметной области [13]. Минимум информации в данном случае должен определяться уровнем «критичности» проекта программного обеспечения, т. е. тем, какое воздействие реализация данного проекта может оказать на своё окружение (в первую очередь исполнителей проекта и пользователей, а впоследствии и на общество в целом или его части). Именно здесь решающее значение имеет человеческий фактор. Значительный объём информации по этому вопросу можно найти в работе [13].

Всё это говорит в пользу подхода к профессиональной ***разработке в определённой предметной области***. Здесь применяется моделирование предметной области [1], появляются специальные термины и образуются некоторые закономерности, которые необходимо учитывать, а в определённых случаях требуется повышенная внимательность и способность длительной концентрации внимания [13]. В этих случаях даже оправдана повышенная предусмотрительность и скрупулёзность исполнителей, ответственных за критические аспекты проекта.

На основании [13] профессиональная разработка включает в себя аспекты ***качества***, ***управления*** и ***экономики***. Вопросы качества и управления имеют существенное значение для проекта, а вопросу экономики, особенно в случае больших проектов, уделяется особое внимание, в частности, согласно [12], формируется новая научная дисциплина — «экономика жизненного цикла программных средств». В рамках аспекта качества должны рассматриваться такие вопросы, имеющие отношение к нефункциональным требованиям программного обеспечения, как удобство, надёжность, производительность и поддерживаемость (сопровождаемость) [19]. Как видно, нефункциональные требования должны распространяться не только на конечный продукт, но и на процесс его производства.

2. Инструменты разработчика программных продуктов

В наборе сегодняшнего разработчика можно встретить такие инструменты, как модели [1, 23, 34], шаблоны [2, 16], предметно-ориентированные языки

[19, 21, 22, 32], компоненты [26, 28, 29, 33, 35], рекомендации и скрипты [19, 25], для автоматизации определённых частей процесса разработки программного обеспечения.

Все эти инструменты стали плодом развития дисциплины разработки компьютерных приложений. С развитием появлялись новые методы, новые инструменты, какие-то отбрасывались, а какие-то укоренялись и совершенствовались. Нельзя сказать, что до появления объектно-ориентированной парадигмы не было представленных выше инструментов, но массово о них заговорили именно с той поры, начиная с конца 1980-х гг. К этому времени уже была определённая теоретическая база, а развитие технологий позволило опробовать это всё на практике, внести коррективы и сделать новые предложения.

Одной из фундаментальных работ считается [26], представленная Малькольмом Дугласом Макилроем (Malcolm Douglas McIlroy) в конце 1960-х гг. на конференции, впервые посвящённой инженерии программного обеспечения. В этой работе делаются предположения о том, как и в какой форме могли бы быть полезны ***повторно используемые активы разработки*** программного обеспечения. В то время под активами понимались библиотеки с набором необходимого функционала. Конкретно предлагались методы каталогизации библиотек компонентов, сегодняшний же подход предполагает взамен метод генерации или автоматического подгона из общего представления [21]. Немалое развитие в этом направлении произвёл Дэвид Парнас (David Parnas), являющийся основоположником множества фундаментальных принципов разработки программного обеспечения. Его главные работы были посвящены процессу проектирования компонентов в рамках семейства родственных программных систем.

Что касается моделей, то большое увлечение ими началось с работ по созданию универсального языка моделирования, с помощью которого, используя единую систему обозначений для выражения своих намерений, могли бы взаимодействовать и профессионально общаться разработчики программных систем разных специализаций, от аналитика до программиста, реализующего отдельный модуль системы. Это вылилось в создание языка UML, и здесь не обойтись без упоминания Гради Буча (Grady Booch), Джеймса Рамбо (James Rumbaugh) и Айвара Джекобсона (Ivar Jacobson), объединивших и развивших всевозможные, в том числе и собственноручно разработанные, подходы к моделированию предметной области программного обеспечения [1].

Фундаментальный труд по шаблонам — работа [16], появившаяся в середине 1990-х гг. В рамках этого проекта по каталогизации шаблонов проектирования принимали участие Эрих Гамма (Erich Gamma), Ричард Хелм (Richard Helm), Ральф Джонсон (Ralph Johnson), Джон Влиссидес (John Vlissides). Однако вдохновителем идеи является Кристофер Александер (Christopher Alexander), не имеющий прямого отношения к компьютерным наукам, один из первых, кто заговорил о языке шаблонов, правда, применительно к архитектурным сооружениям. По некоторым сведениям, например [21], он не был удовлетворён своей теорией, обосновывая это тем, что архитектурные сооружения, построенные с использованием шаблонов, получаются слишком однообразными, что, к сча-

стью, не является проблемой для внутреннего строения проекта программного обеспечения, а, наоборот, идёт ему на пользу.

Предметно-ориентированные языки не являются чем-то очень новым, они были раньше и широко используются сегодня [32], когда платформенные технологии развиты настолько, что дают возможность создавать языки уже на имеющейся технологической базе. Это позволяет обойтись без создания компилятора и не требует от разработчиков детальных знаний процесса их создания. Имеются труды, из которых можно узнать об этом явлении (в частности, [19,22]) и даже ощутить его на практике (например, благодаря [22,32]). В работе [32] всё внимание уделяется разработке текстовых языков, а в работе [22] — графических, где описание сущностей происходит с помощью графических пиктограмм и линий с нанесением поясняющей информации. Первые хороши для решения крупных проблем. Вторые подходят для решения средних и небольших по масштабу задач, но требующих от разработчика определённой внимательности.

В рамках конкретной парадигмы разработки могут использоваться различные ***модели***, статические и динамические, описывающие структуру системы или отдельной её части и взаимодействие элементов внутри неё или с внешним миром. Для эффективного повторного использования основной проблемой, препятствующей повторному использованию модели, являются её границы. Границы мешают использовать её в другом контексте без добавления к ней определённых изменений [34]. Эта проблема частично решается с применением шаблонов проектирования.

Цель ***шаблонов*** — сохранить обобщённые знания экспертов с тем, чтобы впоследствии их можно было использовать в определённом контексте. В основном знания обобщаются в определённой ***горизонтальной предметной области*** (выделенной на основе классов частей систем, исходя из функциональных возможностей) с тем, чтобы их можно было использовать в контексте ***вертикальной предметной области*** (выделенной на основе классов систем согласно области применения). Чтобы эффективно применять шаблоны, в частности проектирования, нужно, во-первых, иметь о них представление, а во-вторых, желательно иметь опыт их применения, которого может и не быть или он утратился со временем. Одно из решений проблемы — встроить эти знания в каркас.

Каркас облегчает разработку системы в целом или отдельной её части, благодаря уже имеющейся в наличии общей структуры будущей программной единицы. От разработчика требуется лишь произвести детализацию каркаса. Возможная проблема состоит в том, что прежде чем использовать каркас, нужно знать, как производить его настройку: если каркас большой, ситуация усложняется. Для облегчения работы с каркасом можно использовать специально созданные для него предметно-ориентированные языки.

Главное преимущество ***предметно-ориентированных языков*** — облегчение процесса детализации каркаса и возможность с их стороны, благодаря встроенным программным рекомендациям и скриптам, объединять компоненты или составляющие в рамках компонента без рутинной работы со стороны разработчика. Одни языки могут применяться для решения мелких проблем в рамках

отдельного компонента, другие — для решения крупных проблем в рамках целой системы. Видно, что перечисленные инструменты взаимосвязаны, и, чтобы избавить разработчиков от путаницы в их использовании и неудобств, связанных с совместимостью, целесообразно объединить их в рамках единого пакета. Подробнее этот вопрос будет раскрыт в разделе, посвящённом фабрике разработки программ.

Стоит сказать ещё несколько слов о подходах, которые оказали непосредственное влияние на появление этих инструментов. В [19, 21, 22] представлены следующие подходы, имеющие отношение к разработке программного обеспечения, учитывая особенности определённой предметной области.

Разработка с использованием моделей (Model-Driven Development). В основе подхода лежит идея, что из модели можно сгенерировать пригодный для применения код. Одно из перспективных направлений подхода — ***архитектура, управляемая моделью*** (Model-Driven Architecture) [27], с которым связаны такие понятия, как «машинно-независимая» и «платформо-независимая модель». ***Языкоориентированное программирование*** (Language-Oriented Programming). Предполагает создание специального языка для решения определённых задач программирования. ***Предметно-ориентированное моделирование*** (Domain-Specific Modeling). Объединяет ряд направлений, связанных идеей создания предметно-ориентированных языков для решения задач определённой предметной области. ***Родовое программирование*** (Generic Programming). Направлено на решение проблемы структурирования компонентов реализации, из которых строятся конечные программные системы. Связано с поиском наиболее абстрактных представлений эффективных решений для оптимального повторного использования компонентов. ***Ментальное программирование*** (Intentional Programming). Главная идея, являющаяся трамплином для двух далее перечисленных подходов, такова — необходима расширяемая среда программирования и метапрограммирования. Предполагает автоматизированное редактирование и рефакторинг кода. Предусматривает решение проблем спутывания кода, включая тем самым идеи ***аспектно-ориентированного программирования*** (Aspect-Oriented Programming). ***Порождающее программирование*** (Generative Programming). Под идеей автоматической (в идеале) генерации приложений объединяет такие темы, как инженерия предметной области, моделирование характеристик и различные техники генерации программ. ***Фабрики разработки программ*** (Software Factories). Подход включает в общем виде идею, представленную в данной статье, а именно: объединение статического содержимого (шаблонов, моделей, предметно-ориентированных языков, компонентов и инструкций) с динамическим содержимым (поддающимися настройке процессами, мастерами, тестами) в рамках пакета, расширяющего интегрированную среду разработки. Несмотря на то, что два последних подхода предлагают генерацию приложений или его частей из моделей, они, тем не менее, предусматривают работу со стороны разработчика, связанную с добавлением программного кода. Это — основное отличие от ***CASE-технологий*** и ***архитектуры, управляемой моделью***, которые предлагают генерацию предложения «нажатием одной кнопки».

3. Инженерия предметной области

Существует множество работ, излагающих идеи инженерии предметной области. Здесь можно выделить два класса направлений. Первый класс составляют работы, описывающие идеи для создания методов разработки приложений, второй класс составляют работы, предлагающие непосредственно методы разработки, на основе идей первого класса и их практическое применение. В качестве примера первого класса можно привести [34], в качестве примера второго класса — [10]. В работе [34] представлена так называемая теория предметной области (Domain theory) для разработки программного обеспечения, а в работе [10] — применение приёмов различных парадигм разработки, в том числе связанных с теорией предметной области, на практике. Используется язык программирования C++.

В работе [21] проведён обзор различных методов анализа и инженерии предметной области. Наибольшее внимание в ней уделяется методу ***характеристико-ориентированного анализа предметной области*** (Feature-Oriented Domain Analysis) и ***методу моделирования предметной области организации*** (Organization Domain Modeling).

Инженерия предметной области (Domain Engineering) имеет прямое отношение к вопросу повторного использования. Согласно определению, взятому из [21], она имеет дело со сбором, систематизацией и сохранением накопленного опыта создания программных систем в «форме средств многократного применения в рамках определённой предметной области». Кроме того, она обеспечивает методы, способствующие повторному использованию этих средств, в процессе разработки новых приложений, облегчает их поиск, классификацию, распространение, адаптацию и сборку.

Главное отличие от традиционных методов разработки состоит в следующем. Во-первых, на стадии анализа, вместо определения требований к отдельной системе, проводится определение требований к семейству систем. Во-вторых, на стадии проектирования вместо разработки проекта отдельной системы проводится разработка проекта семейства систем, ну а на этапе реализации, помимо реализации самой системы, производятся многократно используемые компоненты, которые можно будет использовать в будущем.

Тема повторного использования применительно к разработке программных продуктов весьма полно представлена в работе [33]. Примечательно, что в работе, наряду с организационными аспектами внедрения, представлены ***метрики повторного использования***. К ним относятся атрибутные, структурные, функциональные метрики, метрики, характерные для всего приложения в целом, метрики инженерии предметной области и метрики организационного уровня. Практика введения систематического повторного использования главным образом предполагает создание проблемно и предметно ориентированных абстракций, а также каркасов, в рамках которых эти повторно-используемые абстракции могут быть разработаны путём детализации. Современная тенденция предполагает наличие предметно-ориентированного языка для детализации каркаса, тем самым решается множество проблем, связанных с человеческим

фактором, таких как уровень знаний и объём единовременного восприятия информации в процессе разработки компонента.

Систематическое повторное использование указывает на необходимость разработки не отдельного приложения, а семейства программных систем. А это в свою очередь решается введением в производственный процесс такой технологии, как линейки программных продуктов. С разработкой семейства связаны такие вопросы, как общность и изменчивость характеристик членов семейства. Существуют различные подходы для выявления этих особенностей.

Характерной особенностью ***семейства программных систем*** является наличие обобщённой архитектуры, описывающей всё множество членов конкретного семейства, указывая при этом так называемые точки изменчивости, т. е. те места, в которых имеют место различия между членами семейства.

Линейки программных продуктов реализуют семейство программных систем, применяя имеющиеся производственные активы в рамках организационного процесса. К производственным активам относятся архитектурные образцы, шаблоны проектирования, языки, каркасы, компоненты и т. д. Организационный процесс накладывает определённые ограничения на продукт и его производство. К ограничениям на продукт относятся различные требования к стандартам, интерфейсам, пакетированию, локализации, лицензированию. К ограничениям на производство относятся требования к выбору технологий, платформ, критериям приёмки и отчётности, сюда также относятся требования к бюджету и планированию.

Основополагающей работой в области разработки семейства программных систем является [30]. Заслуживающие внимания работы, касающиеся линеек программных продуктов, — [24, 31].

4. Фабрики разработки программ

Фундаментальной работой по фабрикам разработки программ является работа [19] авторского коллектива во главе с Джеком Гринфилдом (Jack Greenfield). Практическому аспекту фабрик программного обеспечения посвящена работа [25] Гунтера Ленца (Gunther Lenz).

Фабрика разработки программного обеспечения представляет собой линейку программных продуктов, которая благодаря шаблону, основанному на схеме, называемой схемой фабрики программного обеспечения, позволяет конфигурировать инструменты и процессы для автоматизации разработки. Фабрика предполагает наличие специальных компонентов, способных к адаптации, сборке и конфигурированию на основе каркаса.

Основные идеи фабрик программного обеспечения — повышение уровня индустриализации отрасли разработки программного обеспечения, повышение экономической эффективности последующих приложений из определённого класса, экономия за счёт области применения (благодаря уже решённым подпроблемам предметной области), систематическое повторное использование.

Построение семейства приложений происходит в несколько этапов.

Первый этап — ***разработка линейки продуктов***. Включает в себя три подэтапа, состоящих из отдельных фаз.

Первый подэтап — ***анализ линейки продуктов***. Здесь решается вопрос относительно того, какие продукты будут производиться фабрикой. Первая фаза — ***определение линейки продуктов***. Цель — достичь понимания того, как архитектура, учитывающая изменчивость, будет способствовать построению множества схожих, но в то же время уникальных программных продуктов. Также необходимо идентифицировать и зафиксировать проблемы, в решении которых заинтересованы пользователи, и выявить продукты, которые будут способны решить эти проблемы. Немаловажное значение имеет описание контекста, в котором будут использоваться продукты. Вторая фаза — ***определение границ предметной области***, третья фаза — ***определение границ решений***. На второй фазе рассматривается множество проблем, на третьей — множество решений, т. е. то, как именно будут решаться проблемы, описанные во второй фазе.

Второй подэтап — ***дизайн линейки продуктов***. Здесь решается главный вопрос — как именно фабрика будет производить продукты. Первая фаза — ***разработка архитектуры***. Производится поиск необходимых архитектурных шаблонов. Здесь же рассматриваются вопросы о возможности создания специальных инструментов, таких как каркас и предметно-ориентированный язык для использования абстракций, представленных каркасом. Вторая фаза — ***отображение требований***. На этой фазе производится отображение изменчивости требований на изменчивость производственных активов, анализируется зависимость между изменчивыми элементами. Третья фаза — ***определение и автоматизация процесса***.

Третий подэтап — ***реализация линейки продуктов***. Первая фаза — ***обеспечение активами***. На данной фазе строятся или приобретаются необходимые активы. Вторая фаза — ***пакетирование активов***. Здесь все производственные активы упаковываются в шаблон фабрики программного обеспечения.

Итак, на этапе разработки линейки продуктов получен ***шаблон фабрики программного обеспечения***. Будучи установленным в интегрированную среду разработки, данный шаблон становится ***фабрикой разработки программ***.

Следующий этап — ***разработка продуктов***. В роли разработчиков может выступать компания, разработавшая фабрику программного обеспечения. С тем же успехом она может быть использована и сторонними разработчиками, которые приобрели данную фабрику. Разработка продукта включает ***спецификацию продукта***, когда требования выражаются в терминах линейки продуктов с отображением их на производственные активы. То есть внимание уделяется не всему приложению, как при традиционном способе разработки, а отдельным частям уже готового приложения.

5. Фабрика разработки обучающих игровых программных систем

В этом разделе приводятся ответы на главные вопросы, которые стоят в последнем проекте автора, связанном с разработкой фабрики программного обеспечения.

Вопрос № 1. Кто будет использовать фабрику, а кто потреблять продукты, выработанные ею?

Ответ. Фабрику разработки обучающих игровых программных систем лингвистической направленности будут использовать небольшие компании-разработчики программного обеспечения, нацеленные на рынок электронного обучения, решившие занять на этом рынке определённое положение. Это также могут быть специалисты в сфере образования, обладающие знаниями разработки компьютерных приложений и желающие создать себе в помощь продукт, способный решать определённые педагогические задачи, встающие в процессе обучения ими своих подопечных. Продукты, выработанные фабрикой, будут использоваться в учебных заведениях и местах, где есть люди, желающие пройти обучение. Основная задача продукта — способность поддерживать процесс обучения таким образом, чтобы обучающийся не ощущал разницы между обучением, будь он в специальном учебном заведении или дома. То есть необходима эффективная система контроля и помощи со стороны программной системы.

Целесообразность использования фабрики заключается в том, что она позволяет строить приложения, отвечающие требованиям множества заказчиков. Нет необходимости строить излишне перегруженную функциями программную систему обучения, которая может лишь отпугнуть потенциальных заказчиков. А вот лёгкое в освоении приложение с минимальной функциональностью, но выполняющее именно то, что необходимо конкретному заказчику, как раз то, что нужно. Экономия средств в данном случае достигается за счёт применения повторно-используемых элементов для разработки приложений разным заказчикам со своим набором требований, большая часть которых совпадает, а вариации невелики, но обязательны для учёта.

Вопрос № 2. Какие необходимы знания разработчикам, использующим фабрику?

Ответ. Во-первых, это знание процессов, протекающих в ходе учебного процесса, а также — в какой степени продукты фабрики могут их поддерживать. Это нужно для того, чтобы была возможность оценить объём работы по созданию элементов поддержки процессов, не предусмотренных фабрикой. Во-вторых, необходимы знания того, как можно реализовать требования заказчиков, понимать, какие инструменты разработки для этого доступны.

Определить возможности продукта можно, ознакомившись со схемой фабрики программного обеспечения. Схема — это прежде всего способ упорядочивания элементов, использующихся в процессе разработки, иначе говоря, артефактов разработки, таких как модели, файлы исходного кода, манифесты развёртывания и т. д. Говоря образно, их упорядочивание происходит в сет-

ке, где каждый элемент, или ячейка, представляет собой определённую точку зрения, или аспект программного обеспечения. Вначале определяются инструменты для построения продуктов, т. е. языки, каркасы, шаблоны, а уже затем каждой ячейке сопоставляются реальные, пригодные для использования артефакты разработки. Один и тот же инструмент разработки может быть сопоставлен с несколькими ячейками. То есть, например, для работы с несколькими аспектами можно использовать один предметно-ориентированный язык. В конечном счёте сетка преобразуется в схему, где между ячейками уже существуют определённые связи и наложены ограничения. Схема — это также шаблон для описания продуктов фабрики. В [19] приводится удачная аналогия, схема — рецепт, точки зрения — ингредиенты (а также инструменты и процессы приготовления).

Вопрос № 3. Каковы существенные составляющие схемы фабрики в общих чертах?

Ответ. ***Первая категория*** составляющих — требования. Список функциональных требований весьма широк. Делится на категории систем, из которых состоит продукт обучения, получаемый с помощью фабрики. Итак, есть требования к функциональности следующих систем: управления пользователями, обучения, обслуживания, взаимодействия человек-машина, безопасности, коммуникации, учебного материала. В рамках перечисленных систем происходят определённые процессы, которые можно конфигурировать. Нефункциональные требования представлены следующими требованиями. Первая группа — требования к удобству, решают вопросы того, насколько быстро обучаемые могут освоить систему, эффективно ли организована система взаимодействия с системой, не препятствуют ли определённые элементы организации интерфейса продуктивной работе и т. д. Вторая группа — требования к надёжности систем. Решают, какие возможные дефекты системы являются терпимыми, а какие крайне не желательными, препятствующими продуктивному процессу обучения. Содержит требования к нормальному функционированию, защите от сбоев и безопасности. Третья группа — требования к производительности, учитывающие время выполнения определённых операций, пропускную способность и эффективность потребления ресурсов вычислительной машины. Четвёртая группа — требования к сопровождаемости: они включают в себя требования к лёгкости исправления дефектов, модификации, замены, переносимости на другие платформы. ***Вторая категория*** — общая архитектура продуктов и инфраструктура развёртывания. Отвечает на следующие вопросы. Из каких элементов состоит, какой стиль выбран, какие шаблоны предполагает. Как производится настройка продукта? Последний вопрос связан с таким аспектом, как особенность конкретной платформы, которая накладывает определённые ограничения на развёртывание продуктов. ***Третья категория*** — исполняемые артефакты разработки. Сюда входят компоненты, каркасы, конфигурационные файлы, отдельные классы и т. д., отсортированные по принадлежности к конкретной подсистеме (обучения, обслуживания и т. д.). Подробнее о составляющих подсистем можно узнать из работы [8].

Между составляющими схемы могут существовать различные отношения, которые необходимо учитывать и поддерживать их согласованность, чтобы избежать проблем, связанных с перекрытием информации между артефактами и их переименованием. Для разработки конкретного продукта необходимо сконфигурировать схему, а для этого схема должна содержать, помимо фиксированных частей, части изменяемые. Схема фабрики — это простое описание активов, используемых для построения. Реализацией схемы является шаблон фабрики программного обеспечения.

Вопрос № 4. Из каких инструментов состоит шаблон фабрики?

Ответ. Далее перечисляются инструменты и их описание.

Предметно-ориентированные языки. В рамках разрабатываемой фабрики их несколько для каждой из семи подсистем, решающих определённые аспекты. К примеру, язык для установления отношений между составляющими подсистем. Предполагается создание графических и текстовых языков.

Каркасы. Для каждой системы в фабрике представлен свой каркас, содержащий высокоуровневые абстрактные сущности, поддающиеся детализации и переопределению.

Шаблоны проектирования. В большинстве своём это поддержка классических шаблонов, представленных в [16].

Упакованные вместе, эти активы могут использоваться для разработки продуктов в интегрированной среде разработки. Шаблон, так же как и схему, необходимо сконфигурировать перед использованием. О побуждающих мотивах построения фабрики обучающих игровых программных систем лингвистической направленности и предшествующих этому проектах можно узнать из работы [7].

Заключение

В статье представлена информация о процессе разработки программного обеспечения, сфокусированном на систематическом повторном использовании производственных средств и инструментов для их адаптации под конкретный продукт в рамках семейства программных систем.

К сожалению, в рамках статьи невозможно полностью и в деталях описать все аспекты процесса создания пакета повторно используемых элементов и инструментов. Тем не менее автор полагает, что информации в данной статье достаточно для того, чтобы приступить к ознакомлению с подходами, способствующими поднятию зрелости индустрии разработки программного обеспечения. А представленный список литературы и указание на источники в соответствующих разделах поможет получить интересующимся необходимые знания по определённой теме. Для тех, кто желает подробнее ознакомиться именно с инженерной составляющей процесса разработки программного обеспечения, рекомендуется начать с классических работ Дэвида Парнаса (David Parnas) [28, 30] и Барри Боэма (Barry Boehm) [3, 20].

Литература

1. Арлоу Д., Нейштадт А. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование. 2-е изд. СПб.: Символ-Плюс, 2007. 624 с.
2. Басс Л., Клементс П., Кацман Р. Архитектура программного обеспечения на практике. 2-е изд. СПб.: Питер, 2006. 575 с.
3. Боэм Б. У. Инженерное проектирование программного обеспечения. М.: Радио и связь, 1985. 512 с.
4. Брауде Э. Технология разработки программного обеспечения. СПб.: Питер, 2004. 655 с.
5. Брукс Ф. Мифический человеко-месяц или как создаются программные системы. СПб.: Символ-Плюс, 2001. 304 с.
6. Гусс С. В. Модель каркаса программных компонентов поддержки занятий лингвистической направленности в игровой форме // Прикладная информатика. 2010. Вып. 3 (27). С. 62–77.
7. Гусс С. В. Фабрика разработки игровых лингвистических программных систем учебного назначения // Технологии Microsoft в теории и практике программирования: материалы конференции / под ред. проф. В. П. Гергеля. Нижний Новгород: Изд-во Нижегород. гос. ун-та, 2010. С. 110–112.
8. Гусс С. В. Элементы повторного использования для программных систем учебного назначения. Концептуальное проектирование и анализ // Математические структуры и моделирование. 2009. Вып. 20. С. 78–92.
9. Керделлан К., Грезийон Г. Дети процессора: Как интернет и видеоигры формируют завтрашних взрослых. Екатеринбург: У-Фактория, 2006. 272 с.
10. Коплиен Дж. Мультипарадигменное проектирование для C++. СПб.: Питер, 2005. 235 с.
11. Липаев В. В. Программная инженерия. Методологические основы. М.: ТЕИС, 2006. 608 с.
12. Липаев В. В. Технико-экономическое обоснование проектов сложных программных средств. М.: СИНТЕГ, 2004. 284 с.
13. Липаев В. В. Человеческие факторы в программной инженерии: рекомендации и требования к профессиональной квалификации специалистов. М.: СИНТЕГ, 2009. 328 с.
14. Липков А. И. Ящик Пандоры: Феномен компьютерных игр в мире и в России. М.: Изд-во ЛКИ, 2008. 192 с.
15. Макконнелл С. Профессиональная разработка программного обеспечения. СПб.: Символ-Плюс, 2006. 240 с.
16. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма [и др.]. СПб.: Питер, 2008. 366 с.
17. Рекомендации по преподаванию программной инженерии и информатики в университетах = Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering; Computing Curricula 2001: Computer Science: пер. с англ. М.: ИНТУИТ.РУ, 2007. 462 с.
18. Роллингз Э., Моррис Д. Проектирование и архитектура игр: пер. с англ. М.: Вильямс, 2006. 1040 с.

19. Фабрики разработки программ: потоковая сборка типовых приложений, моделирование, структуры и инструменты / Д. Гринфилд [и др.]. М.: Вильямс, 2007. 592 с.
20. Характеристики качества программного обеспечения / Б. Боэм [и др.]. М.: Мир, 1981. 208 с.
21. Чарнецки К., Айзенекер У. Порождающее программирование: методы, инструменты, применение. Для профессионалов. СПб.: Питер, 2005. 731 с.
22. Domain-Specific Development with Visual Studio DSL Tools / S. Cook [etc.]. USA: Addison Wesley Professional, 2007. 563 p.
23. Duffy D. Domain Architectures: Models and Architectures for UML Applications. USA: Wiley, 2004. 406 p.
24. Kang K., Sugumaran V., Park S. Applied Software Product Line Engineering. USA: CRC Press, 2010. 563 p.
25. Lenz G., Wienands C. Practical software factories in .NET. USA: Apress, 2006. 240 p.
26. McIlroy M. Mass Produced Software Components // Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968, Scientific Affairs Division, NATO, Brussels, 1969. URL: http://cs.dartmouth.edu/~doug/components.txt (дата обращения: 28.09.2010).
27. MDA Distilled: Principles of Model-Driven Architecture / S. Mellor [etc.]. USA: Addison Wesley, 2004. 176 p.
28. Parnas D. L. Active Design Reviews: Principles and Practices // Proceedings of the 8th International Conference on Software Engineering, London, August 1985. URL: http://gala.cs.iastate.edu/coms309/references/ActiveDesignReviews.pdf (дата обращения: 28.09.2010).
29. Parnas D. L. On the Criteria to be Used in Decomposing Systems into Modules // Communications of the ACM, 15, 12, December 1972. URL: http://www.cs.umd.edu/class/spring2003/cmsc838p/Design/criteria.pdf (дата обращения: 28.09.2010).
30. Parnas D. L. On the Design and Development of Program Families // IEEE Transactions on Software Engineering. 1976. Vol. SE-2, No. 1. URL: http://web.cecs.pdx.edu/~omse532/Parnas_Families.pdf (дата обращения: 28.09.2010).
31. Parnas D. L. Software Product Lines: What to do when Enumeration won't Work // Proceedings of the First International Workshop on Variability Modelling of Software-Intensive Systems, January 16–18, 2007. URL: http://ulir.ul.ie/bitstream/10344/160/2/09SQRL1045.pdf (дата обращения: 28.09.2010).
32. Rahien A. DSLs in Boo. Domain-Specific Languages in .NET. USA: Manning Publications, 2010. 352 p.
33. Reuse-Based Software Engineering. Techniques, Organization, and Controls / H. Mili [etc.]. USA: John Wiley and Sons, 2002. 682 p.
34. Sutcliffe A. The Domain Theory: Patterns for Knowledge and Software Reuse. USA: CRC Press, 2002. 424 p.
35. Wang A., Qian K. Component-Oriented Programming. USA: Wiley-Interscience, 2005. 336 p.

Математические структуры и моделирование
2010, вып. 22, с. 69–73

УДК 517.925.15+004.92

КОМПЬЮТЕРНЫЙ ГРАФИЧЕСКИЙ ПАКЕТ ДЛЯ ПОСТРОЕНИЯ ПОТЕНЦИАЛЬНОЙ ФУНКЦИИ В СЛУЧАЕ КАТАСТРОФЫ «ЗВЕЗДА»

А. К. Гуц, Е. О. Хлызов

Приводится краткое описание возможностей созданного компьютерного графического пакета CatVis, позволяющего изучать потенциальную функцию ряда элементарных катастроф, в том числе и катастрофы «звезда», управляющее пространство которой шестимерно. Данная катастрофа интересна тем, что с её помощью можно прогнозировать состояния 6-ярусных лесных экосистем.

Введение

Элементарная теория катастроф возникла в конце 1960-х гг. благодаря усилиям французского математика Тома Рене.

Сегодня математическая теория катастроф [2] находит применение в самых различных исследованиях, связанных с изучением изменений состояний равновесий исследуемой системы под воздействием внешних управляющих факторов, образующих управляющее пространство.

Системе в теории катастроф отвечает потенциальная функция V, математическое выражение для которой в случае числа внешних факторов ≤ 4 описывается теоремой Тома. Для исследователя важным является получение полной информации о форме и числовых значениях потенциальной функции в ходе изменений внешних факторов.

Полезно иметь графический пакет программ для компьютера, с помощью которого такая информация о потенциальной функции представляется в полной мере. Авторам известна только одна доступная в Интернете программа, решающая указанную задачу [3].

Наиболее часто для моделирования применяется элементарная катастрофа «сборка» (управляющее пространство двумерно, бифуркационное множество представляется одномерной кривой). Когда управляющее пространство имеет размерность более двух, появляются сложности с визуализацией бифуркационной поверхности. Для того чтобы сохранить наглядность, можно зафиксировать

Омский государственный университет им. Ф. М. Достоевского.
E-mail: hlyzov@gmail.com

точку в управляющем пространстве и рассмотреть сечения поверхности плоскостями, параллельными координатным плоскостям. Но это становится тем сложнее, чем больше размерность управляющего пространства. На практике анализ катастрофы типа «ласточкин хвост» уже требует большой вычислительной работы.

Для облегчения задачи визуализации при анализе катастроф с большим числом управляющих факторов авторами статьи разработан программный продукт CatVIS.

В статье приводится краткое описание возможностей CatVis и рассматривается работа с катастрофой «звезда», управляющее пространство которой шестимерно. Данная катастрофа интересна тем, что с её помощью можно прогнозировать состояния 6-ярусных лесных экосистем.

1. Описание программного пакета

В качестве основного языка программирования был выбран язык Java[1]. Основным аргументом в пользу такого выбора были возможность выполнения программы непосредственно в вёб-браузере (используя технологию java-апплета) и наличие огромного числа готовых библиотек.

В качестве графической подсистемы была выбрана библиотека Processing[2], распространяющаяся по лицензии LGPL и имеющая широкие возможности по визуализации данных[3].

Реализация некоторых вычислительных алгоритмов была заимствована из пакета Java Scientific Library, разработанного M. T. Фланаганом[4].

Программный пакет CatVIS изначально задумывался как вспомогательный инструмент, облегчающий исследователю понимание процессов, возникающих в изучаемой системе при движении в управляющем пространстве.

Возможности пакета CatVIS:

1. Визуализация двумерных сечений по всем плоскостям в заданной точке.

2. Визуализация потенциальной функции $V(x)$.

3. Показ всех критических точек катастрофы при заданных управляющих факторах (решения уравнения $dV/dx = 0$).

4. Визуализация поверхности бифуркационного множества в пространстве $\{x, p_1, p_2\}$, где p_1, p_2 — управляющие факторы.

5. Интерактивность — изменение управляющих факторов сразу же влечёт за собой изменение всех графиков.

[1] См.: URL: http://java.com.
[2] См.: URL: http://processing.org.
[3] Большое количество примеров можно найти на сайте: URL: http://openprocessing.org.
[4] См.: URL: http://www.ee.ucl.ac.uk/~mflanaga.

Подобные программы создавались и за рубежом, но у CatVIS есть как минимум одно существенное отличие — для фиксированной точки управляющего пространства можно получить уравнения сечений в параметрической форме (для ознакомления с используемым алгоритмом см. [1]). Это позволяет гораздо точнее реагировать на изменение управляющих факторов пользователем. Кроме того, CatVIS может быть встроен в вёб-страницу.

2. Демонстрация возможностей

Для примера рассмотрим работу программы с катастрофой типа «звезда», потенциал которой задаётся уравнением [1]:

$$V(x) = \frac{x^8}{8} + \frac{A}{6}x^6 + \frac{B}{5}x^5 + \frac{C}{4}x^4 + \frac{D}{3}x^3 + \frac{E}{2}x^2 + Fx. \tag{1}$$

Управляющее пространство с факторами A, B, C, D, E, F шестимерно. Следовательно, для полного его представления понадобится 15 уникальных сечений — по 5 сечений на каждый управляющий фактор с учётом повторений.

Для определённости положим $A = 0, B = 0, C = 50, E = -24$.

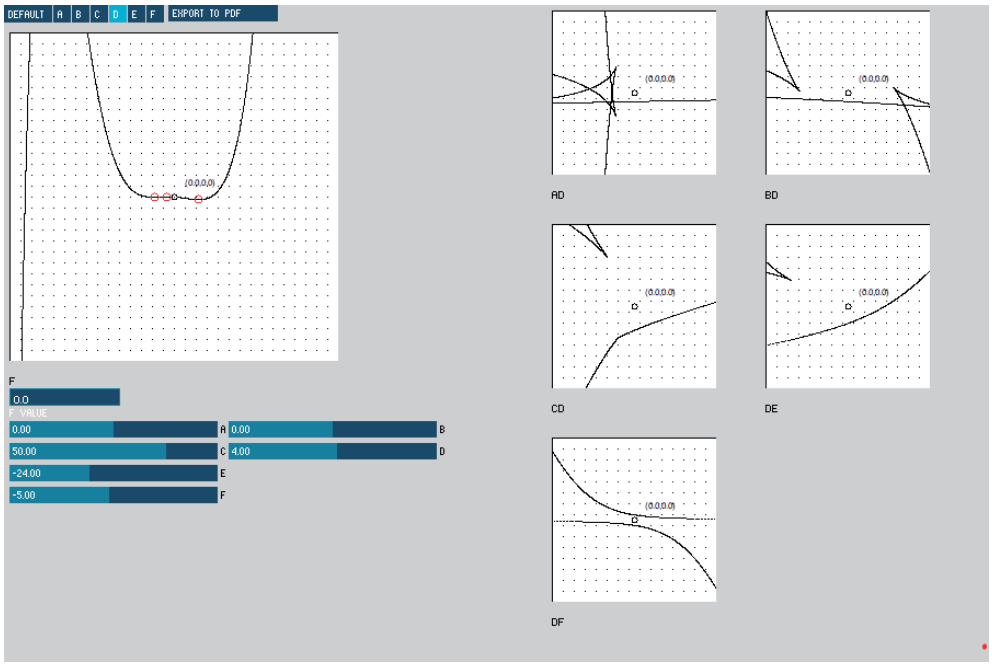

Рис. 1. Главное окно программы

На рис. 1 представлено окно программы CatVIS. В верхнем левом углу расположены кнопки навигации по сечениям, кнопка доступа к окну настроек и кнопка экспорта всех сечений и потенциальной функции в PDF. В нижнем левом углу находятся контроли для точного задания управляющих факторов. В правой части — интерактивные графики сечений. График потенциальной функции $V(x)$ с отмеченными точками экстремума — в левой части основного окна.

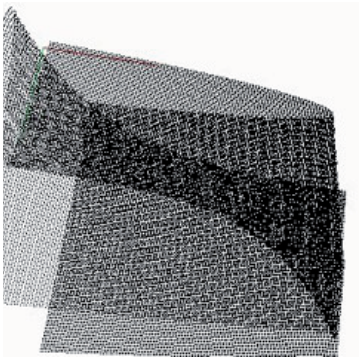

Рис. 2. Взгляд со стороны оси x

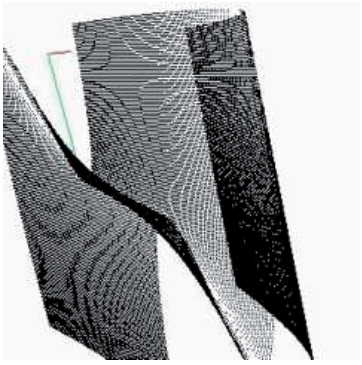

Рис. 3. Взгляд со стороны оси D

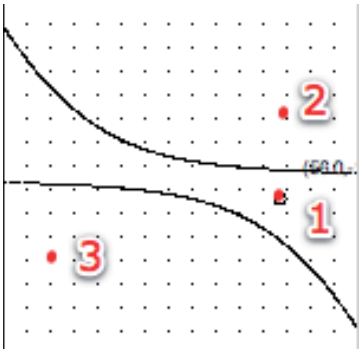

Рис. 4. Сечение бифуркационного множества

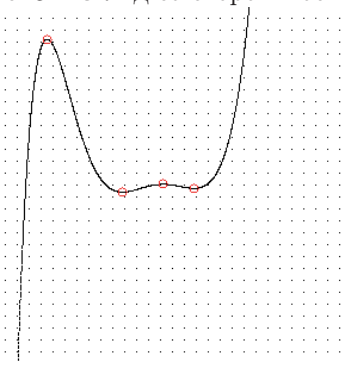

Рис. 5. Потенциал в ситуации (1)

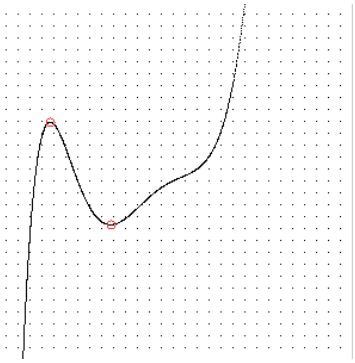

Рис. 6. Потенциал в ситуации (2)

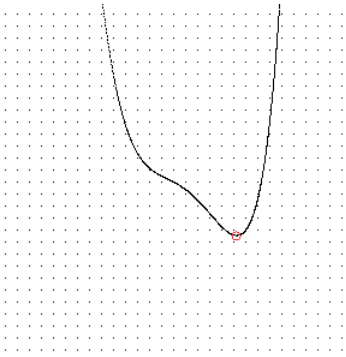

Рис. 7. Потенциал в ситуации (3)

Рассмотрим плоскость DF, где $D \in [-100..100], F \in [-100, 100]$. На рис. 2 и 3 показано изображение бифуркационного множества в пространстве $\{x, D, F\}$.

На полупрозрачных графиках затемнённость области определяется количеством критических точек. Для упрощения можно сказать, что граница области принадлежит бифуркационному множеству в случае, если соседняя область отличается по цвету. На рис. 2 хорошо видно, что центральная область более тёмная — если повернуть поверхность относительно оси OF на $\pi/4$, то легко увидеть, как именно расположены складки.

Рассмотрим сечение бифуркационного множества плоскостью, параллельной DF, в указанной выше точке (рис. 4). На рис. 5, 6, 7 изображены соответствующие виды потенциальной функции.

Литература

1. Гуц А. К., Хлызов Е. О. Компьютерная визуализация сечений бифуркационных множеств в теории катастроф Тома // Вестник Омского университета. 2010. Вып. 2. С. 26–28.
2. Постон Т., Стюарт И. Теория катастроф и её приложения. М.: Мир, 1980.
3. Catastrophe teacher. URL: http://pagesperso-orange.fr/l.d.v.durjardin.
4. Woodcock A., Poston T. A geometrical study of the elementary catastrophes // Lectures Notes in Math. Berlin: Springer, 1974. Vol. 373.

Математические структуры и моделирование
2010, вып. 22, с. 74–85

УДК 004.588

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СОСТАВЛЕНИЯ ОБУЧАЮЩИХ И ПРОВЕРОЧНЫХ ЗАДАНИЙ ПО ЕСТЕСТВЕННЫМ ЯЗЫКАМ

П. С. Долматова

In this article is presented special programming language for making natural language practice exercises and examinations. Special software for creating exercises and exams using this language has been developed. Examples of tasks using the developed language and software are considered.

Введение

В настоящее время при изучении языков широко используется компьютерное оборудование. Существуют различные программные комплексы для изучения языков и контроля знаний по языкам. Однако все подобные подходы были основаны на методиках изучения языка с помощью языка-посредника (чаще всего родного для человека языка), и не существовало средств и методов, основанных на независимом представлении понятий изучаемого языка и не использующих для обучения язык-посредник. Авторы [1, с. 20] предложили изучать языки посредством различных «действий» с предметами на экране без какого-либо другого языка-посредника. Это осуществляется следующим образом: слово и соответствующие действия демонстрируются пользователю; представляется то же самое слово и соответствующая начальная ситуация; пользователю нужно выполнить действие при помощи курсора; компьютер проверяет правильность действия, выполненного пользователем.

Если набор ситуаций фиксирован, то у пользователя может сложиться связь определённого понятия не только с изучаемым словом, но и с другими словами, присутствующими в соответствующих командах. Поэтому ситуации (задания) должны формироваться случайным образом, как предложено в [2, с. 217] и [4], т. е. при повторном запуске программы должны появляться другие, но аналогичные ситуации.

Таким путём могут быть продемонстрированы понятия, представляющие не только реальные объекты, но и воображаемые. В [6] демонстрируется естественное диалоговое представление абстрактных пространств.

Американский университет в Центральной Азии (г. Бишкек, Кыргызстан).
E-mail: dolmatova_p@mail.auca.kg

В [5] предложена общая схема программного обеспечения, которое включает в себя три составляющих: «Формализованное подмножество слов естественного языка», «Конструктор заданий» и «Интерфейс для обучения и тестирования» — необходимых для поставленных целей. Однако ранее это не было реализовано.

В [7] и [3] описана уже предложенная методика интерактивного компьютерного представления понятий естественных языков и её усовершенствованная реализация, позволяющая пользователю осуществлять более гибкое управление объектами на дисплее. И на этой основе построено эскизное программное средство для изучения киргизского языка без использования языков-посредников.

Далее может потребоваться изучение более обширных языковых понятий, которые более не могут быть выражены графически, поэтому возникает необходимость в ином способе представления обучающего материала. Материал можно представлять в виде текста на изучаемом языке. Если составлять набор заданий на каком-либо языке, то он будет конечным — обучаемый не сможет быть уверен, что получил правильное представление о том, что означают те или иные понятия. Следовательно, нужно каким-то образом составить задания так, чтобы их количество было бесконечным или хотя бы достаточно большим. Причём следует учесть тот факт, что в распоряжении составителя учебника зачастую бывают достаточно ограниченные ресурсы, например, время или ёмкость носителя информации, на котором будет распространяться обучающий материал. Выходом из такой ситуации может послужить случайное составление заданий с использованием генератора случайных чисел того компьютера, на котором обучаемый будет выполнять курс обучающих программ. Также нужно будет задать некий алгоритм генерации заданий, который позволит множеству генерируемых заданий оставаться большей частью осмысленным.

Все вышеизложенные требования были учтены при разработке программного обеспечения (ПО) для создания обучающих и контролирующих заданий по естественным языкам. Это программное обеспечение состоит из редактора заданий и основной программы — тестирующей или обучающей. Приложения имеют модульную структуру. Каждый модуль является обособленной частью набора типов заданий, которые требуются для того, чтобы полностью написать учебник или экзамен по языку. Данное ПО предназначено для специалистов-филологов, которые могут составлять с помощью неё вопросы для проверки знания того или иного языка человеком, отвечающим на вопросы. Поскольку далеко не все филологи владеют языками программирования, то для того, чтобы самостоятельно писать подобные программы, был разработан специальный язык, которому составитель экзамена может довольно быстро обучиться и, используя его, формировать задания. Разработанный язык называется TaskLang. Рассмотрим синтаксис этого языка.

1. Описание языка TaskLang

Программирование на языке TaskLang состоит из:

- определения фиксированных объектов;
- формирования случайных объектов;

- составления случайных заданий;
- формирования учебника или экзамена из заданий.

Всё это можно сделать в интерпретаторе языка TaskLang, который мы назвали Language Education System (LES). LES разработан на объектно-ориентированном языке C#. Для создания физических объектов использовались готовые библиотеки обработки столкновений Box2D, Catto, E. (2006–2007) и Box2DX, Kalasouski, I. (2008).

1.1. Объекты в TaskLang

Объекты бывают двух типов: фиксированные и случайные (множество однородных фиксированных объектов).

Фиксированный объект — это строка. Она может идентифицировать различные типы информации: текстовую, графическую и звуковую.

Если использован случайный объект, то вместо него из соответствующего множества подставляется (случайным образом) фиксированный объект. Если случайный объект используется в задании несколько раз, то он принимает одно и то же значение. Можно запросить второе значение этого же объекта, и тогда он принимает отличное от первого значение, но оно остаётся постоянным. Если задание выполнить второй раз, то значения случайных объектов изменятся в соответствии с этим правилом. Определение фиксированного объекта включает:

- имя объекта (object-name);
- класс, написанный на языке программирования C#, способный отвечать на внешние события и рисовать экземпляры данного класса (object-class). Такие классы могут быть взяты из стандартной библиотеки классов либо написаны пользователем на языке программирования C#;
- текстовое слово или список его грамматических форм (object-word);
- значения атрибутов. Стандартным атрибутом является object-word.

Синтаксис объявления фиксированных объектов:

[Declare*<object-name>#<object-class>|Word=
<object-word>|<attribute-name2>=<attribute-meaning2>|...]

Пример 1. Объявим два объекта:

[Declare*Rose#RoseDrawer|Word=роза|Color=розовый]

[Declare*Poppy#PoppyDrawer|Word=мак|Color=красный]

Определение случайного объекта включает:

- имя объекта;
- список фиксированных объектов.

Случайные объекты бывают двух типов:

- случайная последовательность фиксированных объектов;
- заданная последовательность фиксированных объектов.

Синтаксис объявления случайных объектов:

[<extended-object-name>~<primary-object-name1>|
<primary-object-name2>|...]

[<extended-object-name>=<primary-object-name1>|
<primary-object-name2>|...]

Ссылка на случайный объект:

[<extended-object-name>#<number>]

Она возвращает фиксированный объект по его номеру в определении случайного объекта.

Пример 2. Два случайных объекта:

[RandFlower~FlRose|FlPoppy]

[FixFlower=FlRose|FlPoppy]

[RandFlower#1] и [RandFlower#2] могут меняться, но всегда будут различными;

[FixFlower#2] всегда будет FlPoppy.

1.2. Составление случайного текстового задания

Формирование текста задания как последовательности

- строковых констант (включая слова, пробелы и знаки препинания);

  Синтаксис:

  <plain text>

- имён фиксированных объектов;

  Синтаксис:

  <plain text>

- имён случайных объектов (рассматриваемых в качестве ссылок на object-words) с определениями необходимых грамматических форм of object-words.

  Синтаксис:

  [GetProperty*[<extended-object-name1>#<number1>]|
  <attribute-name1>]

  или

  [GetProperty*<primary-object-name1>|<attribute-name1>]

Вследствие того, что в некоторых языках (например, киргизском, казахском) есть специальные алгоритмы словообразования, эти определения могут быть записаны как функции.

Синтаксис: [<language-code>*<word>|<primary-form-of-affix>].

Для других языков такие определения могут быть записаны как массивы: [Gram=word|grammar-form-1|grammar-form-2|grammar-form-3|...]

Пример 3. В этом примере мы хотим, чтобы на экране был один цветок. Этот цветок выбирается случайно. Им может оказаться роза или мак.

Для объявленного случайного объекта RandFlower, состоящего из FlRose со свойствами Word = роза, Color = розовый, и FlPoppy со свойствами Word = мак, Color = красный, мы вводим следующий текст задания:

Какого цвета [GetProperty*[RandFlower#1]|Word]?

И пользователю будет показан один из следующих вариантов:

Какого цвета мак?

Какого цвета роза?

1.3. Составление графического задания

- расположение случайных объектов в графическом окне;
- задание числа слоев для случайных объектов;
- задание направления медленного движения для некоторых случайных объектов.

Пример 4. (Продолжение Примера 3)

[Draw*[RandFlower#1]]

(число слоёв равно 1 и координаты нулевые (центр графической области) по умолчанию).

1.4. Составление словесного ответа в случайных заданиях

Это специальная функция, выполнение которой отображает приглашение для ввода текстового ответа во время выполнения задания. Приглашение для ввода отображается только тогда, когда в очереди событий все события, находящиеся ранее, уже выполнены.

Пример 5. (Продолжение Примеров 3 и 4)

[Answer*[GetProperty*[Flower#1]|Color]]

1.5. Составление ответа-действия к заданию

Ответ составляется как последовательность условий, применимых к действиям, совершаемым над изображением (во временном порядке). Одним из таких условий является утверждение о существовании или не существовании общих точек двух изображений объектов, рассматриваемых во время перемещения объекта

мышью. Также имеется возможность использовать отдельные события grasp и ungrasp для поднятия и отпускания объектов. Это последовательность логических выражений с текстовыми словами объектов и основными отношениями:

$$Intersect(object{-}name1, object{-}name2) \equiv (object{-}image1 \cap object{-}image2 \neq \emptyset);$$
$$Subset(object - name1, object - name2) \equiv (object - image1 \subset object - image2)$$

и операциями AND, OR, XOR, NOT. Эти последовательности определяют требуемую последовательность действий пользователя.

Если все условия выполнены, то на экран выводится ответ программы о том, что задание выполнено верно. Если не выполняется хотя бы одно из условий данной последовательности, то пользователю выдаётся сообщение о неверном ответе.

1.6. БНФ

Язык TaskLang определяется следующим образом:

ParseLanguage = TextCharacter, Brackets, TextCharacter

TextCharacter = (* Any UTF character except [ ] = | # * ~ / *)

Brackets = '[', Comment | FastChoise | KGZALG | MATH | DECLARE | GETPROPERTY | SETPROPERTY | DRAW | ANSWER | PLAYSOUND | DELAY | EVENT | INCORRECTEVENT | GRASP | UNGRASP | Extended Object | Assignment | RandomAssignment, ']'

Комментарий:

Comment = ParseLanguage

Быстрый выбор:

FastChoise = ParseLanguage, '|', ParseLanguage, '|', ParseLanguage

Алгоритм словообразования в киргизском языке:

KGZALG = 'KGZALG*', ParseLanguage, '|', ParseLanguage

Математические преобразования:

MATH = 'MATH*', NumericExpression

Объявление объектов:

DECLARE = 'DECLARE*', ParseLanguage, '#', ParseLanguage, '|', Parse Language, '=', ParseLanguage

Работа со свойствами объектов:

GETPROPERTY = 'GETPROPERTY*', ParseLanguage, '|', ParseLanguage

SETPROPERTY = 'SETPROPERTY*', ParseLanguage, '|', ParseLanguage, '=', ParseLanguage

Рисование:

DRAW = 'DRAW*', ParseLanguage

Ответ:

ANSWER = 'ANSWER*', ParseLanguage

Работа со звуком:

PLAYSOUND = 'PLAYSOUND*', ParseLanguage

Способ организации задержки:

DELAY = 'EVENT*', IntegerLiteral

Событие неверного ответа:
INCORRECTEVENT = 'INCORRECTEVENT*', BooleanExpression
Манипуляции с помощью курсора:
GRASP = 'GRASP*', ParseLanguage
UNGRASP = 'UNGRASP*', ParseLanguage
Случайный объект:
ExtendedObject = ParseLanguage, '#', IntegerLiteral
Фиксированное задание:
Assignment = ParseLanguage, '=', ParseLanguage
Случайное задание:
RandomAssignment = ParseLanguage, '~', ParseLanguage
Числовые выражения языка TaskLang:
NumericExpression = IntegerLiteral | ('-', NumericExpression) | (Numeric Expression, ('+' | '-' | '*' | '/'), NumericExpression)
IntegerLiteral = ('1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'), '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
Булевы функции:
BooleanExpression = ('NOT' BooleanExpression) | (BooleanExpression, ('or' | 'and'), BooleanExpression) | ('(', BooleanExpression, ')') | IntersectExpression | SubsetExpression
Пересечение:
IntersectExpression = 'INTERSECT(', ParseLanguage, ')'
Подмножество:
SubsetExpression = 'SUBSET(, ParseLanguage, ')'

2. Редактор модулей

Редактор модулей — это часть программы, предназначенная непосредственно для создания заданий на языке TaskLang.

Интерфейс редактора модулей (рис. 1) включает в себя два основных элемента для редактирования списка заданий и формирования готового набора заданий в виде дерева. После нажатия на кнопку «Добавить задание» создаётся задание выбранного типа и добавляется в список заданий. По списку можно перемещаться, нажимая кнопки «влево» и «вправо». Для того чтобы задание было показано пользователю, нужно, чтобы составитель включил это задание в дерево заданий. Сначала необходимо добавить раздел в дерево заданий, потом включить отображаемое задание как подпункт раздела с помощью контекстного меню. Одно и то же задание может быть добавлено в дерево заданий несколько раз. Так как задания включают в себя случайный элемент, то одно и то же задание будет воспринято пользователем как набор однотипных заданий.

В редакторе есть шаблоны типов заданий, с помощью которых составителю легче разрабатывать задания. Разработаны шаблоны для таких типов заданий, как диктант, текст, действия. Каждый шаблон реализован в виде отдельного модуля, поэтому в дальнейшем несложно будет расширить набор шаблонов,

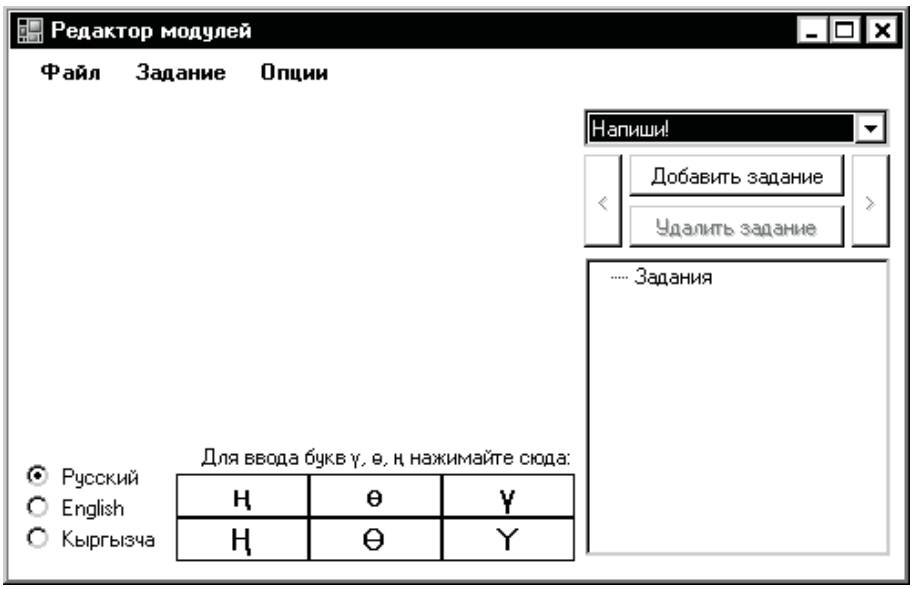

Рис. 1. Редактор модулей

просто добавив соответствующие модули. Рассмотрим интерфейс каждого из шаблонов в отдельности.

Задание типа диктант имеет своей особенностью то, что в нем обязательно присутствует звуковая составляющая. К этому типу можно отнести задания, где требуется написать произнесённое программой слово, ответить на аудиовопрос и др. В задание типа диктант можно включать звуковые файлы, текст, графические файлы. На рис. 2 представлен интерфейс редактора модулей с примером задания типа диктант. Здесь показаны следующие области:

1) область списка звуковых файлов. Если задание содержит один звуковой файл, то при каждом запуске этого задания будет воспроизведён звук из этого файла; если же задание содержит более одного звукового файла, то при запуске задания программа случайным образом выбирает и воспроизводит только один из звуковых файлов;

2) область графического изображения. Сюда при необходимости можно вставить графические файлы, соответствующие добавленным звуковым файлам;

3) область текста. В эту область при необходимости вставляется текст, обычно поясняющий основное задание, которое даётся при помощи соответствующего звука;

4) область кнопок используется для редактирования вариантов одного задания.

Задание типа текст в отличие от диктанта не содержат звуковых файлов, задание даётся с помощью текста. К такому типу относятся, например, задания: вставить пропущенные буквы, записать слово в требуемой форме (падеж,

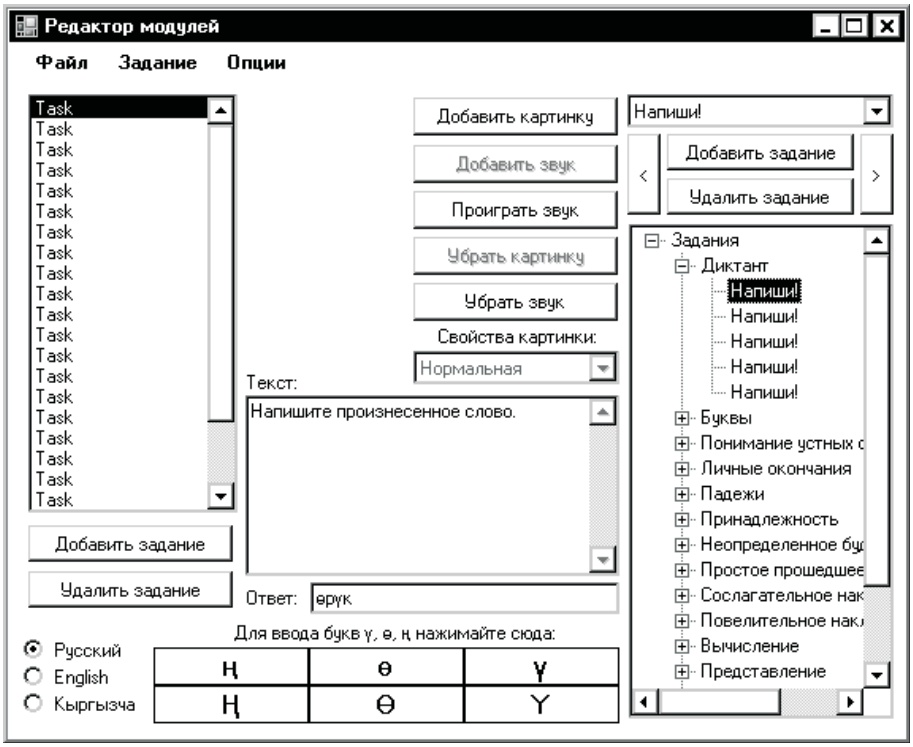

Рис. 2. Диктант

число, время и т. д.), понять текст и ответить на поставленные вопросы и др. Интерфейс редактора модулей с примером задания типа текст дан на рис. 3. Здесь показаны области:

1) область редактирования задания. В этой области на языке TaskLang составитель пишет задание. При составлении задания можно предусмотреть случайную генерацию задания;

2) область предварительного просмотра задания. В этой области отображается лишь тот текст задания, который будет выведен на экран в случае, если задание написано без ошибок, или будет выведено сообщение об ошибке, если таковая имеется. Эта область необходима для проверки правильности задания и его корректировки;

3) область предварительного просмотра начального значения строки ответа;

4) область отображения ответов, которые будут засчитаны программой как верные.

Задания типа действия пишутся на языке TaskLang. Эти задания требуют от пользователя выполнения некоторых действий с объектами на сцене с помощью курсора мыши.

Во вкладке Task definition (рис. 4) на языке TaskLang описывается сцена; тут же находится область предварительного просмотра задания. Во вкладке Scene

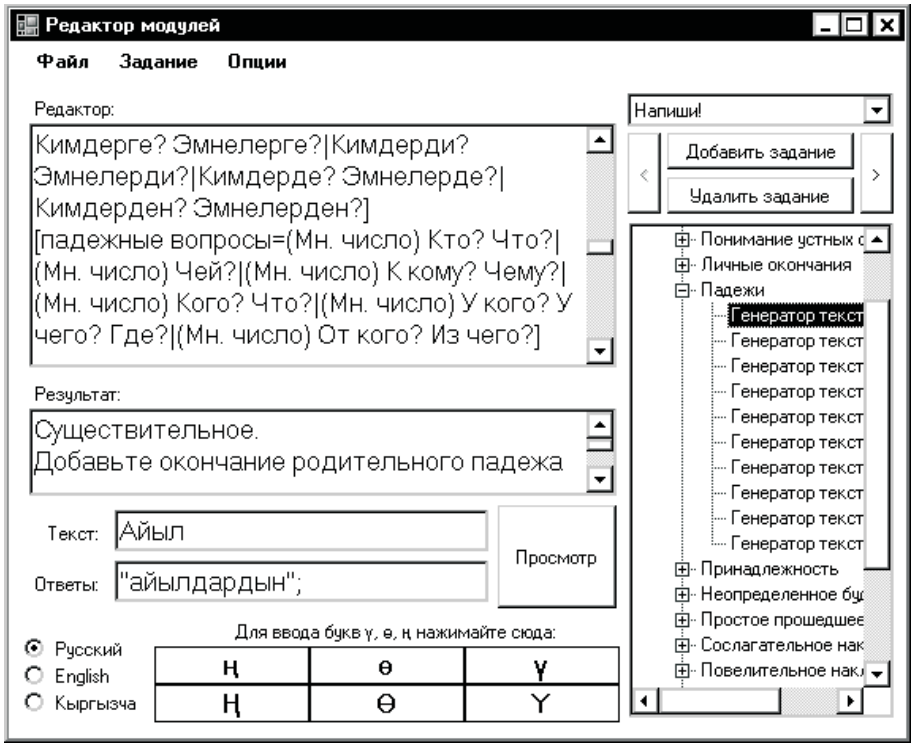

Рис. 3. Текст

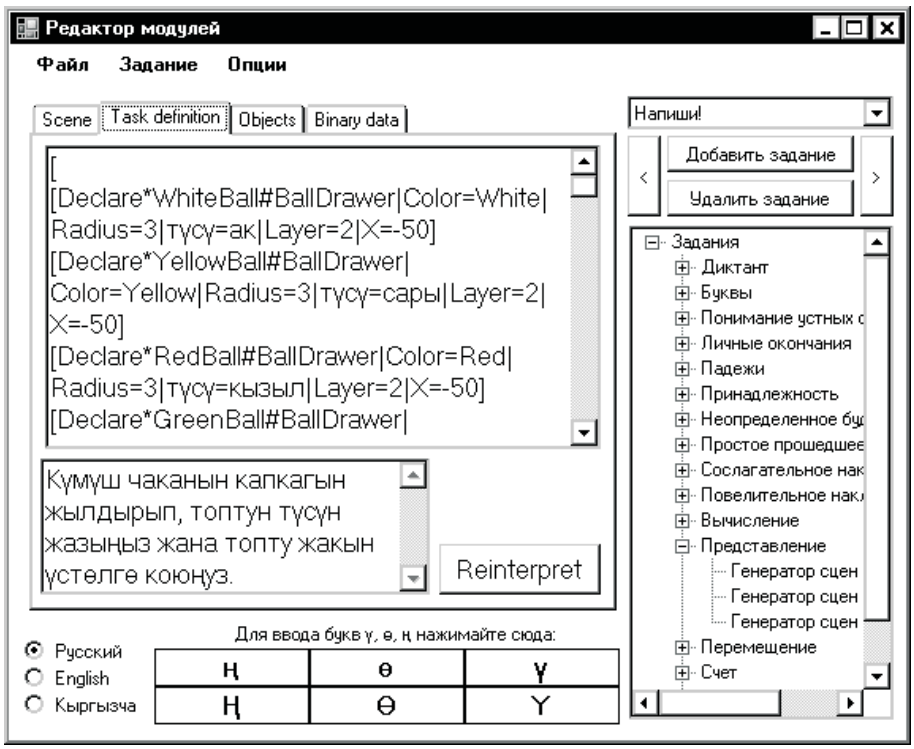

Рис. 4. Описание сцены на языке TaskLang

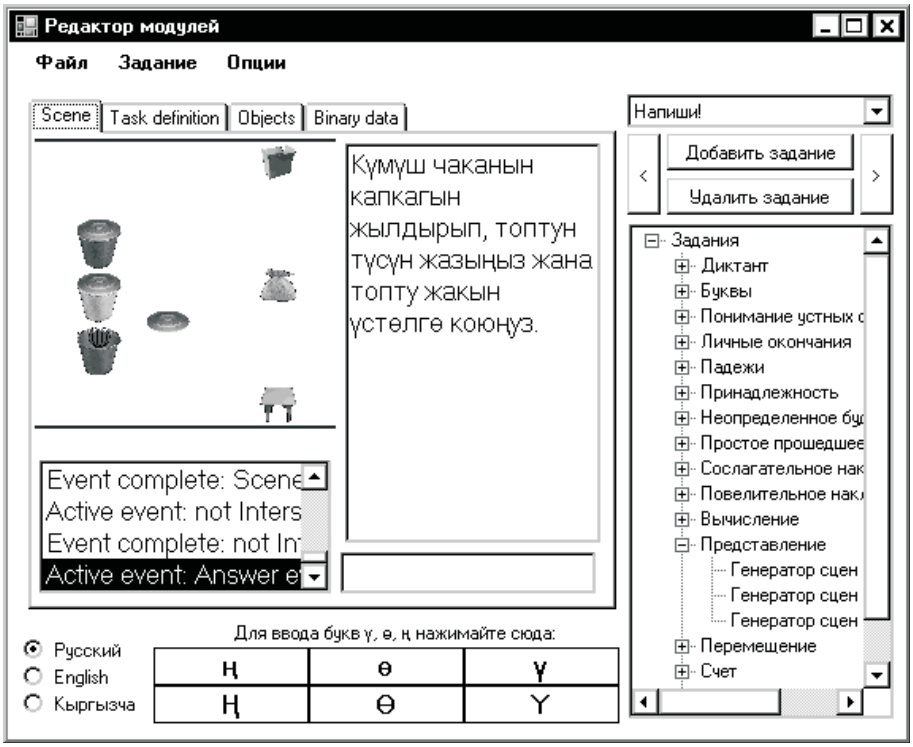

Рис. 5. Сцена

(рис. 5) изображается сцена. Рисунки можно вставлять из графических файлов и можно рисовать с помощью программы на языке TaskLang, в которой есть классы простейших геометрических фигур. Здесь же находится окно просмотра событий, в котором по очереди отображаются активные в данный момент события, а также сообщения об их успешном или не успешном окончании этих событий. Это окно предназначено для составителя заданий; здесь он может проверить правильность составленного им задания, отладить задание.

Вкладка Objects предназначена для создания новых объектов на языке C#. Во вкладке Binary data в задание можно добавлять звуковые и графические файлы.

Заключение

В заключение хотелось бы добавить, что с помощью программного обеспечения LES был создан первый вариант комплексного экзамена по киргизскому языку, некоторые задания из которого отображены на рис. 1–5. Этот экзамен был использован как демонстрационный в ряде вузов (Киргизско-Российский славянский университет, МУК, КГУСТА и др.), школ и в Институте теоретической и прикладной математики НАН КР. Экзамен выполняли как студенты и учащиеся, так и преподаватели, в том числе преподаватели киргизского языка. Участники экзамена дали положительную оценку ПО и разработанным с помощью него заданиям, проявили заинтересованность в использовании данного ПО в учебном процессе.

Литература

1. Панков П. С. Обучающая и контролирующая программа по словоизменению в кыргызском языке на ПЭВМ. Бишкек: Мектеп, 1992.
2. Панков П. С., Джаналиева Ж. Р. Опыт и перспективы использования комплекса UNIQTEST уникальных тестовых заданий в учебном процессе // Образование и наука в новом геополитическом пространстве: тез. докл. науч.-практ. конф. Бишкек: МУК, 1995.
3. Панков П. С., Долматова П. С. Построение дифференциальных уравнений для гибкого управления объектами с помощью компьютерной мыши // Исследования по интегро-дифференциальным уравнениям. Бишкек: Илим, 2007. Вып. 37. С. 18–23.
4. Pankov P. S. Independent learning for Open society // Collection of papers as results of seminars conducted within the frames of the program «High Education Support» / Foundation «Soros-Kyrgyzstan». Bishkek, 1996. Iss. 3. P. 27–38.
5. Pankov P. S., Alaeva S. A., Kutsenko V. A. Algorithmic Interactive Presentation of Notions // «Speech and Computer» SPECOM'2006: Proceedings of the 11th International Conference. Saint-Petersburg: Anatolya Publishers, 2006. P. 478–480.
6. Pankov P. S., Bayachorova B. J. Using computers to perform non-Euclidean topological spaces // The 6th conference and exhibition on computer graphics and visualization «Graphicon–96». Saint-Petersburg, 1996. Vol. 2. P. 73–84.
7. Pankov P. S., Dolmatova P. S. Algorithmical Language for Computer-Based Presentation of Notions // IKECCO'2007: Proceedings of the 4th International Conference on Computers and Techniques. Almaty, 2007. P. 274–279.

Математические структуры и моделирование
2010, вып. 22, с. 86–94

УДК 004.655.3

ОБРАБОТКА РЕЛЯЦИОННЫХ БАЗ ДАННЫХ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ

Д. А. Лыфарь

Review of existing technologies which can be used to process relational databases on graphical processor units (GPU). Short description of GPU primitives like prefixsum, Map-Reduce, parallel sort and solutions which are built on them are provided.

1. Введение

Поиск и обнаружение информации в базах данных является одной из важнейших задач в информационных технологиях сегодня. Академические центры таких компаний, как Oracle, Microsoft, SAP, заинтересованы поиском масштабируемых решений на базе графических процессоров (GPU). Вычисления на графических процессорах получили серьёзный толчок в развитии лишь недавно и по возможностям, которые предоставляют текущие API для GPU (OpenCL/CUDA) приближаются к центральным процессорам, что способствует появлению направления GPGPU (Generic Programming on GPU). Эта область относительно молода, однако ряд исследований показывает, что с помощью GPU возможно эффективно решать задачи общего назначения, в частности, такие как: data mining и обработка запросов к большим объёмам данных.

Основным препятствием в обработке баз данных на графическом процессоре является ограниченный объем памяти GPU. Традиционно производительность баз данных упирается в подсистему ввода–вывода, т. е. центральный процессор не является узким местом. Хотя диски существенно дешевле оперативной памяти и имеют высокую ёмкость, они гораздо медленнее оперативной памяти. С распространённой скоростью вращения шпинделя, равной 7200 об/мин, оборот требует примерно 8,33 мс, что почти на 5 порядков превышает время обращения к оперативной памяти (которое составляет 100 нс). Однако сейчас с ростом объёма оперативной памяти серверов, которые обычно обслуживают СУБД, скорость обработки становится немаловажным фактором, так как зачастую вся таблица(ы) помещается в оперативной памяти. На рынке существует такое понятие, как *commodity hardware*. В русской интерпретации это означает недорогое аппаратное обеспечение, в противовес дорогим серверным системам,

Омский государственный университет им. Ф. М. Достоевского.
E-mail: dlyfar@gmail.com

из которых можно построить вычислительный комплекс под нужды бизнеса. В данный момент commodity hardware можно назвать компьютер с 4–8 ядрами, 8–16 Гб оперативной памяти и дисками в 15K об/мин. Так как подобное оборудование имеет большое превосходство по объёму памяти над графическим процессором, то область, где мы можем получить прирост производительности, — это непосредственно сама обработка данных (полагаясь на возможность параллельного исполнения задач на векторной архитектуре GPU). В этой статье мы сделаем обзор существующих решений, предназначенных для обработки больших объёмов данных на GPU, рассмотрим детали реализации и компоненты системы управления баз данных на GPU.

Прежде всего стоит рассмотреть архитектуру существующих баз данных. Практически в основе любой базы данных в качестве структуры данных для хранения лежит B-дерево. Оно представляет собой сбалансированное дерево поиска, созданное специально для быстрой работы с дисковой памятью. B-деревья отличаются от красно-чёрных деревьев тем, что узлы B-дерева могут иметь много дочерних узлов — до тысяч, так, что степень ветвления B-дерева может быть очень большой (хотя она обычно определяется характеристиками используемых дисков. B-деревья схожи с красно-чёрными деревьями в том, что все B-деревья с n узлами имеют высоту $O(\lg n)$, хотя само значение высоты в этом дереве существенно меньше, чем у красно-чёрного, за счёт более сильного ветвления. Таким образом, B-деревья также могут использоваться для реализации многих операций над динамическими множествами за время $O(\lg n)$.

Мы указывали, что узким местом системы обработки баз данных является производительность дисковой подсистемы, по этой причине в алгоритмах обработки баз данных делают акцент на:

- количество обращений к дисковой подсистеме,
- вычислительное время (время процессора).

Как уже было указано выше, ранее производительность обработки данных упиралась в дисковую подсистему, в настоящий момент можно отметить появление новых исследований на эту тему, в частности обработки на GPU (исследование компании Oracle [10]). Существует множество аргументов в пользу поиска масштабируемых решений на графическом процессоре. С течением времени это становится наиболее экономически эффективным. На момент написания статьи одна из самых быстрых видеокарт от NVIDIA (GTX 285) соразмерна по цене с четырёхядерным процессором (имея мощность около 200 ватт). Скорость математических операций для подобного видеоадаптера порядка 1 триллион в секунду (что в 20 раз быстрее самого быстрого CPU). Так как видеоадаптеры можно объединить в одном сервере, то четыре GPU обеспечивают 80-кратное превосходство над скоростью одного CPU при увеличении мощности лишь в 6 раз, что ведёт к существенной экономии потребляемой мощности и затрат на обслуживание серверов (их становится существенно меньше). Появление кластерных решений для GPU способствует тому, что в ряде задач по обработке данных GPU будут иметь приоритет перед CPU.

2. Обзор существующих решений

2.1. Исполнение табличных SELECT запросов средствами GPU

В области обработки баз данных на GPU уже существует ряд исследований ([6,7] и др.). Рассмотрим одно из последних и наиболее близких к исследуемому вопросу в работе [1]. Под обработкой баз данных в этой статье будем понимать выполнение операций CRUD (это сокращённое наименование четырёх базовых функций управления данными: создание (C), чтение (R), редактирование (U) и удаление (D)) над множеством данных с помощью языка запросов. Цель указанных исследований — показать, насколько GPU превосходят обычные процессоры на определённом множестве операций.

Среди исследований, которые также можно отнести к обработке баз данных, можно выделить: поиск подстроки, выявление зависимостей, быструю сортировку ([5, 8, 9]), являющиеся базовыми блоками при реализации CRUD.

Так же как и в этом исследовании, был выбран язык SQL для описания запросов. SQL — это признанный декларативный язык в индустрии для манипулирования данными. Он является промежуточным слоем между логикой программы и набором данных. Реализация SQL на графическом процессоре позволит ускорить программу без изменений в исходном коде. Несмотря на очевидное преимущество этого подхода, пока не существует законченных реализаций SQL на GPU.

Архитектура СУБД включает в себя множество компонентов, поэтому, чтобы исключить ненужную работу по реализации парсера, виртуальной машины, тестов, была выбрана существующая СУБД SQLite. Она широко используется в ряде проектов и поддерживается большим числом известных компаний. Виртуальная машина SQLite отвечает за преобразование декларативного SQL в набор команд, реализующих логику запроса (всего около 128 команд). Каждая из команд виртуальной машины может принимать до пяти аргументов (команды машины достаточно высокоуровневые: открытие таблицы, загрузка значения, математические операции, операции перехода и т. п.). Самый простой SELECT состоит из инициализации таблицы, цикла по всем строкам и очистки ресурсов. Решение использует практически все виды памяти, предоставляемые программной моделью CUDA. Регистровая память используется для хранения смещений в блоке данных и результатов, разделяемая (shared) память — для хранения результатов каждого потока. В константной памяти хранится программа виртуальной машины, которую исполняет каждый из потоков, а также служебная информация, например, размерность обрабатываемых типов. Глобальная память хранит обрабатываемые данные. В этом исследовании реализована поддержка операторов выборки вида:

```
SELECT id, uniformi, normali5 FROM test WHERE
       uniformi > 60 AND normali5 < 0
SELECT id, uniformf, normalf5, normalf20 FROM test
       WHERE NOT uniformf OR NOT normalf5
       OR NOT normalf20
```

Существует ряд препятствий для использования графических процессоров при реализации запросов на основе архитектуры SQLite:

- Внутренняя структура хранения данных (B-tree).
- Ограничения архитектуры GPU.
- Вопросы хранения данных (исходных и результатов).

Множество обрабатываемых данных было ограничено столбцами с типами целое (int) и число одинарной точности с плавающей точкой (float). Тестовые таблицы представляют собой набор кортежей, в каждом из которых — пять элементов. В дополнение к реализации функции SELECT авторы реализовали набор примитивных операций, таких как: SUM, MIN, MAX, COUNT, AVG. Данные одной таблицы постоянно хранятся в памяти GPU для того, чтобы снизить размер передаваемых данных через шину для каждого запроса. Для запуска выборки над множеством данных необходимо преобразование из B-дерева в структуру, аналогичную табличному представлению. Поэтому необходимо преобразование из B-дерева к табличной структуре. Тесты проводились на Tesla C1060 GPU, имеющем 4 Гб оперативной памяти, что и является ограничением размера таблицы (для того, чтобы избежать подкачки данных). Множество результатов тоже требует определённого количества памяти (заранее неизвестное). Этот размер выбирается эмпирически, в будущем возможен поиск эвристики, позволяющей свести размер множества результатов к минимуму. Для метаданных, таких как размер блока, число колонок в блоке, размер каждой колонки, предусмотрена отдельная область памяти.

Существует ряд факторов, которые необходимо учесть при сравнении результатов, полученных на CPU и GPU:

- Мы загружаем данные на GPU полностью, в то время как SQLite на CPU запрашивает их с диска по необходимости. Для того чтобы устранить очевидное преимущество GPU, в этом случае в SQLite использовалась таблица, которая хранит содержимое таблицы полностью в памяти.
- Чтобы сделать традиционное исполнение SQLite быстрее, результаты, возвращаемые ей, игнорировались, а также она была скомпилирована с опцией хранения всех временных файлов в памяти.
- Не использовалась страничная память (pinned или paged-memory), которая позволяет примерно в два раза увеличить скорость обмена данными между host и device. Это означает, что результаты, в которых учитывается время передачи данных на GPU, могут быть лучше, если провести ряд оптимизаций, основанных на этом типе памяти.

На скорость исполнения влияет такое явление, как расхождение потоков (thread divergence) из-за присутствия условных переходов. Оно заключается в том, что GPU исполняет код условного перехода по разу для каждого условия последовательно. То есть, имея обычную ветку if… then… else, этот код будет выполнен сначала для тех потоков, которые в первой части имеют истинное значение, и затем для тех, которые имеют ложное. Полученные результаты

в этом исследовании можно разделить на те, в которых учитывалось время передачи с хоста на устройство и не учитывалось. Без учёта времени загрузки данных на GPU запросы выполнялись в среднем в 50 раз быстрее, чем на обычном процессоре, если учитывать, что время передачи в среднем GPU быстрее в 36 раз (табл. 1).

Таблица 1

Ускорение исполнения запросов на GPU по отношению к CPU

Запросы (тип)	Ускорение на GPU	Ускорение на GPU (со временем передачи)
int	42,11	28,89
float	59,16	43,68
aggregation($AND, OR, etc.$)	36,22	36,19
average	50,85	36,20

2.2. Map-Reduce framework Mars на GPU

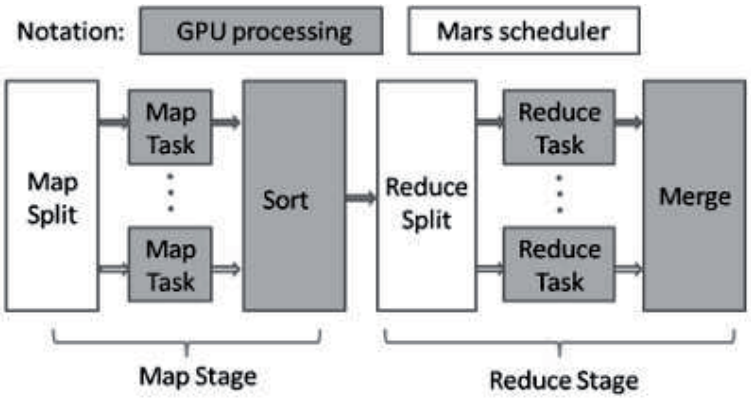

Рис. 1. Архитектура Mars

Компания Microsoft провела исследование, которое показало эффективность Map-Reduce алгоритмов на GPU [3]. В ходе исследований был реализован фреймворк, облегчающий реализацию задач, которые могут быть эффективно решены при переносе их на Map-Reduce фазы. Изначально подобный подход использовался в программировании кластеров с большим количеством компьютеров, чтобы получить вычислительную мощность, сравнимую с суперкомпьютерами на commodity hardware. MapReduce — это фреймворк для вычисления некоторых наборов распределённых задач с использованием большого количества компьютеров (называемых нодами), образующих кластер (ядер в нашем случае). Работа MapReduce состоит из двух шагов: Map и Reduce. На Map-шаге происходит предварительная обработка входных данных. Для этого один из компьютеров (называемый главным узлом — master node) получает входные данные задачи, разделяет их на части и передаёт другим компьютерам (рабочим узлам — worker node) для предварительной обработки. Название данный

шаг получил от одноимённой функции высшего порядка. На Reduce-шаге происходит свёртка предварительно обработанных данных, главный узел получает ответы от рабочих узлов и на их основе формирует результат — решение задачи, которая изначально формулировалась.

Преимущество MapReduce заключается в том, что он позволяет распределённо производить операции предварительной обработки и свёртки. Операции предварительной обработки работают независимо друг от друга и могут производиться параллельно (хотя на практике это ограничено источником входных данных и/или количеством используемых процессоров). Аналогично множество рабочих узлов могут осуществлять свёртку — для этого необходимо только, чтобы все результаты предварительной обработки с одним конкретным значением ключа обрабатывались одним рабочим узлом в один момент времени. Хотя этот процесс может быть менее эффективным по сравнению с более последовательными алгоритмами, MapReduce может быть применён к большим объёмам данных, которые могут обрабатываться большим количеством серверов. Так, MapReduce может быть использован для сортировки петабайта данных, что займёт всего лишь несколько часов. Классический пример использования MapReduce — это подсчёт слов в документе:

```
// Функция, используемая рабочими нодами на Map-шаге
// для обработки пар ключ-значение из входного потока
map(String name, String document):
    // Входные данные:
    //   ключ --- название документа
    //   значение --- содержимое документа
    // Результат:
    //   ключ --- слово
    //   значение --- всегда 1
    for each word w in document:
        EmitIntermediate(w, "1");

// Функция, используемая рабочими нодами на Reduce-шаге
// для обрабоки пар ключ-значение, полученных на Map-шаге
reduce(String word, Iterator partialCounts):
    // Входные данные:
    //   ключ --- слово
    //   значения --- всегда 1. Количество записей в partialCounts и
    //     требуемое значение
    // Результат:
    //   общее количество вхождений слова word во все
    //     обработанные на Map-шаге документы
    int result = 0;
    for each v in partialCounts:
        result += parseInt(v);
    Emit(AsString(result));
```

Поскольку GPU пока ещё не поддерживает динамическое выделение памяти на устройстве во время исполнения кода на нём, были использованы заранее выделенные массивы для работы фреймворка. Входящие данные, промежуточные и конечные результаты хранятся в трёх различных типах массивов: массив ключей, массив значений и индекс. Под индексом понимается запись вида: <key offset, key size, value offset, value size> для каждой пары ключ-значение. Имея индексную запись для каждой пары ключ–значение, мы запрашиваем ключ или значение по соответствующему смещению в массиве ключей или значений.

Рис. 1 иллюстрирует принцип работы MapReduce фреймворка. Так же как MapReduce, фреймворк, исполняемый на обычном процессоре, имеет две фазы.

На стадии map оператор MapSplit разделяет пары входных значений на равные наборы таким образом, число наборов было равно числу потоков. Таким образом достигается балансировка на стадии map — каждый поток ответственен за один набор. После окончания фазы map промежуточные пары ключ–значение сортируются таким образом, чтобы пары с одинаковым ключом хранились последовательно.

На стадии reduce оператор ReduceSplit делит отсортированные промежуточные пары ключ–значение на наборы одинакового размера. Пары с одинаковыми ключами принадлежат одному набору. Число наборов равно числу потоков. Поток с наибольшим threadID отвечает за наборы с большими ключами. Это даёт гарантию, что вывод фазы reduce будет также отсортирован. На последнем шаге вывод со всех потоков записывается в один общий буфер.

Существует планировщик, который отвечает за некоторые стадии алгоритма и исполняется на CPU. В задачи планировщика входит: подготовить входные пары ключ–значение, вычислить необходимое число потоков, скопировать входные массивы на GPU и забрать результаты исполнения.

На GPU выделяется место как для входных данных, так и для результата до исполнения программы на GPU. Но размеры выходных данных из фаз map и reduce неизвестны заранее, более того, существуют конфликты по записи, когда много потоков записывают результаты в общий массив, так как в модели CUDA нет механизмов, обеспечивающих синхронизацию между разными блоками исполнения на аппаратном уровне. Это потребовало реализации собственного механизма записи результатов (который одинаков для обоих фаз).

Каждая из задач map имеет результат в виде трех чисел: число промежуточных результатов, размер ключей (в байтах) и размер значений (в байтах). Основываясь на суммарном объеме ключей (либо значений), планировщик вычисляет префиксную сумму [2] (которая может быть вычислена параллельно) и создаёт массив, необходимый для хранения промежуточных результатов. Далее так же вычисляется место для индекса. Таким образом планировщик выделяет нужное количество памяти на устройстве для хранения промежуточных результатов.

Далее каждая map задача выводит промежуточный набор ключ–значение в выходной массив и обновляет индекс. Так как каждая из задач map знает позиции, в которые она должна писать, то конфликты по записи исключены.

Эти две стадии существуют на GPU, так как он не поддерживает динамического выделения памяти с кода, исполняющегося на устройстве, как уже говорилось. Это стало причиной существования двух стадий в каждой из фаз (map или reduce). Первая стадия отвечает за подсчёт необходимого места для своих результатов, вторая делает всю работу и выводит результат в заранее выделенный массив. Для программиста это выглядит так (фаза map для подсчёта слов в документе):

```
MAP\_COUNT(key, val, keySize, valSize){
1: for each word w in key
2:    EMIT\_INTERMIDIATE\_COUNT(w.length, sizeof (int));
}

MAP (key, val, keySize, valSize) {
1: for each word w in key
2:      EMIT\_INTERMIDIATE (w, 1);
}
```

3. Заключение

Мы рассмотрели наиболее значимые на данный момент исследования в области применения GPU для обработки данных. Несмотря на существующие ограничения архитектуры GPU, результаты говорят о существенном преимуществе в области обработки больших объёмов данных. Мощность графических процессоров растёт быстрее, чем мощность центральных процессоров по закону Мура (который для CPU уже перестал выполняться), поэтому существенными ограничениями являются скорость шины и объем памяти. Частично вопрос нехватки памяти решается использованием pinned memory, т. е. проецированием памяти RAM и передачей порциями на GPU по запросу. Так как шина PCI-E является полнодуплексной, то вместе с запросами на чтение из pinned memory возможны одновременные запросы на запись. Усложнённая модель программирования в сравнении с CPU накладывает свои ограничения: отсутствие возможности динамического выделения памяти на GPU, невозможность синхронизации между исполняемыми блоками приводят к появлению конструкций, которые добавляют накладные расходы там, где на CPU их нет. Такие особенности архитектуры, как расхождение потоков, конфликты банков разделяемой памяти, особенности доступа к глобальной памяти, обусловливают необходимость адаптации существующих примитивов для параллельного программирования (сортировка, префиксная сумма) для GPU.

Литература

1. Bakkum P., Skadron K. Accelerating SQL Database Operations on a GPU with CUDA / Department of Computer Science University of Virginia.
2. Harris M. Parallel Prefix Sum (Scan) with CUDA / NVIDIA corp.
3. Mars: A MapReduce Framework on Graphics Processors / Bingsheng He, Wenbin Fang, Qiong Luo and other; Microsoft research.
4. NVIDIA CUDA Compute Unified Device Architecture / NVIDIA corp.
5. Parallel Data Mining on Graphics Processors / Wenbin Fang, Ka Keung Lau, Mian Lu and others; Microsoft Research Asia, Microsoft China Co.
6. Relational Joins on Graphics Processors / Bingsheng He, Ke Yang, Rui Fang, Mian Lu, Naga K. Govindaraju, Qiong Luo, and Pedro V. Sander. ACM SIGMOD, 2008.
7. Relational Query Co-Processing on Graphics Processors / Bingsheng He, Mian Lu, Ke Yang, Rui Fang, Naga K. Govindaraju, Qiong Luo, and Pedro V. Sander. TODS, 2009.
8. Schatz M. C., Trapnell C. Cmatch: Fast Exact String Matching on the GPU. URL: http://www.cbcb.umd.edu/software/cmatch/ (дата обращения: 20.10.2010).
9. Sintorn E., Assarsson U. Fast Parallel GPU-Sorting Using a Hybrid Algorithm. URL: http://www.cse.chalmers.se/~uffe/hybridsort.pdf (дата обращения: 20.10.2010).
10. Why graphics processors will transform database processing / IEEE Spectrum. URL: http://www.spectrum.ieee.org/ computing/ software/ data-monster/ 0 (дата обращения: 20.10.2010).

Математические структуры и моделирование
2010, вып. 22, с. 95–100

УДК 004.932.2

ИСПОЛЬЗОВАНИЕ МЕТОДА SURF ДЛЯ ОБНАРУЖЕНИЯ УСТОЙЧИВЫХ ПРИЗНАКОВ ИЗОБРАЖЕНИЯ ПРИ СОЗДАНИИ СФЕРИЧЕСКИХ ПАНОРАМНЫХ СНИМКОВ

Г. М. Джгаркава, Д. Н. Лавров

In this article the search algorithm and a description of specific image points Speeded Up Robust Features (SURF) is considered. The method can be used to compare images, finding objects in images when creating spherical panoramic images from multiple images.

Введение

В статье рассматривается алгоритм поиска и описания особых точек изображения Speeded Up Robust Features (SURF). Метод может применяться для сравнения изображений, поиска объектов на изображениях при создании сферических панорамных снимков из нескольких изображений.

1. Проблематика

Когда мы смотрим на окружающих нас людей, предметы, природу, наш мозг проделывает огромный объем работы, чтобы обработать весь поток визуальной информации. Нам не составит труда найти знакомого нам человека на фотографии или отличить здание от памятника. Основные моменты, которые делают задачу распознавания образов не тривиальной для решения с использованием вычислительной техники, заключаются в следующем:

- *Масштаб.* Изображения имеют разный масштаб. Предметы, которые мы воспринимаем как одинаковые, на самом деле занимают разную площадь на разных изображениях.

- *Место.* Интересующий нас объект может находиться в разных местах изображения.

Омский государственный университет им. Ф. М. Достоевского.
E-mail: `george@dzhgarkava.com`

- *Фон и помехи.* Предмет, который мы воспринимаем как что-то отдельное, на изображении никак не выделен и находится на фоне других предметов. Кроме того, изображение не идеально и может быть подвержено всякого рода искажениям и помехам.

- *Проекция, вращение и угол обзора.* Изображение является лишь двумерной проекцией нашего трёхмерного мира. Поэтому поворот объекта и изменение угла обзора кардинальным образом влияют на его двумерную проекцию — изображение.

Итак, даны два изображения, одно из них будем считать образцом, другое — сценой. Задача сводится к определению факта наличия образца на сцене и к его локализации. При этом образец на сцене может: иметь другой масштаб; быть повернут в плоскости изображения; быть в произвольном месте сцены; может быть зашумлен, виден не полностью, частично заслонён другими предметами; может иметь отличную от образца яркость и контраст; его может не быть совсем.

Самое простое и очевидное решение заключается в следующем: возьмём образец в разных масштабах, повернём его на всевозможные углы, переберём всевозможные места на сцене и сравним все эти шаблоны попиксельно со сценой. Однако для реализации этого решения потребуется огромное количество ресурсоёмких операций сравнения. Кроме того, непосредственное сравнение образца со сценой может дать плохой результат из-за шумов, искажений, заслонения, объектов фона.

Для того чтобы уменьшить количество сравнений, нам необходимо упростить задачу. Выделим на образце некие ключевые точки и небольшие участки вокруг них. Ключевой будем считать такую точку, которая имеет некие признаки, существенно отличающие её от основной массы точек. Например, это могут быть края линий, небольшие круги, резкие перепады освещённости, углы и т. д. Предполагая, что ключевые точки присутствуют на образце всегда, можно поиск образца свести к поиску на сцене ключевых точек образца. А поскольку ключевые точки сильно отличаются от основной массы точек, то их число будет существенно меньше, чем общее число точек образца.

2. Обзор метода SURF

SURF решает две задачи — поиск особых точек изображения и создание их дескрипторов, инвариантных к масштабу и вращению. Это значит, что описание ключевой точки будет одинаково, даже если образец изменит размер и будет повернут (здесь и далее мы будем говорить только о вращении в плоскости изображения). Кроме того, сам поиск ключевых точек тоже должен обладать инвариантностью, так, чтобы повёрнутый объект сцены имел тот же набор ключевых точек, что и образец.

Метод ищет особые точки с помощью матрицы Гессе. Детерминант матрицы Гессе достигает экстремума в точках максимального изменения градиента яркости. Он хорошо детектирует пятна, углы и края линий.

Рис. 1. Показаны особые точки изображения в образце (слева) и на сцене (справа). Несмотря на то, что сцена имеет другой масштаб, угол обзора и частично заслонена другим объектом, ключевые точки достаточно точно идентифицируются

Гессиан инвариантен относительно вращения. Но не инвариантен масштабу. Поэтому SURF использует разномасштабные фильтры для нахождения гессианов.

Для каждой ключевой точки считается направление максимального изменения яркости (градиент) и масштаб, взятый из scale-коэффициента матрицы Гессе. Градиент в точке вычисляется с помощью фильтров Хаара.

После нахождения ключевых точек SURF формирует их дескрипторы. Дескриптор представляет собой набор из 64 (либо 128) чисел для каждой ключевой точки. Эти числа отображают направление градиента вокруг ключевой точки. Поскольку ключевая точка представляет собой максимум гессиана, то это гарантирует, что в окрестности точки должны быть участки с разными градиентами. Таким образом, обеспечивается дисперсия (различие) дескрипторов для разных ключевых точек.

Направление градиента окрестностей ключевой точки считается относительно направления градиента вокруг точки в целом (по всей окрестности ключевой точки). Таким образом, достигается инвариантность дескриптора относительно вращения. Размер же области, на которой считается дескриптор, определяется масштабом матрицы Гессе, что обеспечивает инвариантность относительно масштаба. Направление градиента также считается с помощью фильтра Хаара.

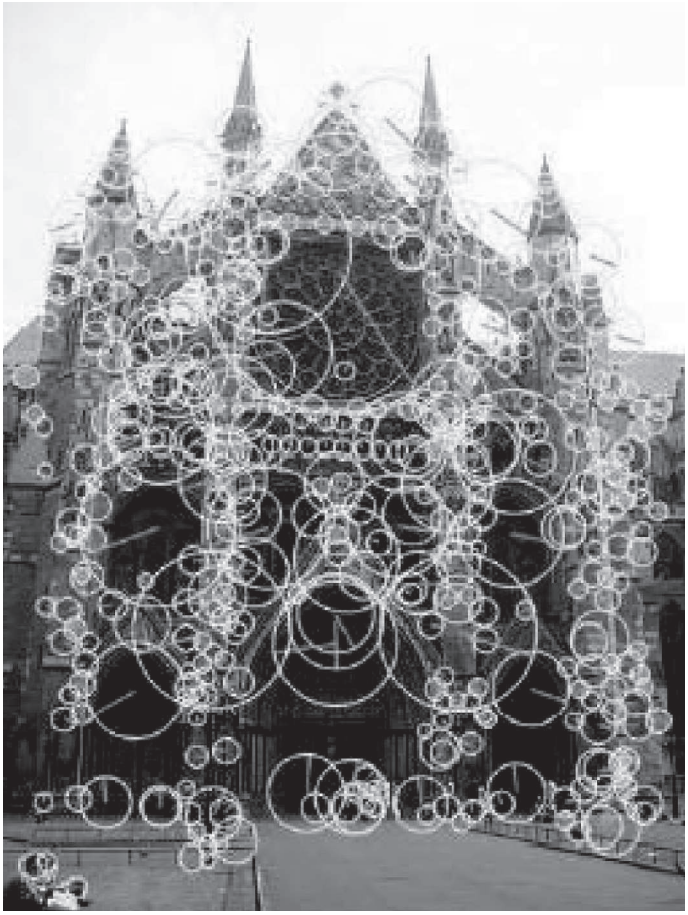

Рис. 2. Выделены особые точки изображения здания, найденные с помощью матрицы Гессе. Диаметр круга показывает масштаб особой точки, линия по радиусу — направление градиента яркости

3. Применение на практике

Съемка панорам в рамках данной статьи производилась с помощью установки, которая состоит из четырех синхронизированных между собой для одновременной съемки зеркальных фотокамер, оснащенных fish-eye объективами.

Объектив «рыбий глаз», «фишай» (англ. *fish-eye*) — сверхширокоугольный фотографический объектив, который имеет угол изображения, близкий к 180° или больший. Принято считать, что все разновидности «фишаев» имеют угол зрения в 180°, но это не так. Изображение пейзажа с углом 180° даёт круговую картинку, но кадр представляет собой прямоугольник. Есть два способа устранения этого несоответствия и три типа «фишаев»:

- Циркулярный — на полученном кадре изображение занимает не всю его площадь, а лишь вписанный круг. Такой объектив имеет угол зрения 180° в любом направлении (справа налево, сверху вниз и т. п.). С помощью такого объектива можно сделать снимок, на котором будет изображён, например, весь небосвод.

- Диагональный (или «полнокадровый») — полученный кадр целиком занят изображением, однако угол зрения в 180° соответствует лишь диагоналям кадра. Другими словами, полный круг, который даёт циркулярный тип, этот объектив не изображает в кадре. В данном случае наоборот: кадр вписывается в круговое изображение.

- С кругом изображения более 180° — обычно имеют также круглое изображение и угол зрения может составлять 220°.

При очень широком угле зрения возникают сильные искажения перспективы: задний план кажется дальше, чем на самом деле. Кроме того, увеличение во многих съёмочных ситуациях может быть недостаточным, и освещённость сильно падает по краю.

Чтобы компенсировать эти недостатки, в объектив при его разработке намеренно вводят отрицательную дисторсию («бочка»). Тогда увеличение по центру становится больше, и в этой области объектив работает как менее широкоугольный. Только дисторсия позволяет довести угол зрения до 180° и больше. Тем не менее такая компенсация вносит свои искажения перспективы — выпячивание центра, а также ведёт к искажениям формы предметов: прямые линии изображаются кривыми.

В ходе работы был использован циркулярный объектив Sigma AF 8 mm f/3.5 EX DG Circular Fisheye Nikon F. При портретной установке камеры данный объектив позволяет получить снимок с углом обзора 180° по вертикали и 120° по горизонтали.

Четыре снимка, частично накладываясь друг на друга, полностью покрывают окружающее пространство. Для точного наложения снимков друг на друга и получения кругового панорамного изображения на исходных снимках выделяются ключевые точки с помощью метода SURF.

4. Тесты

Тестирование метода проводилось на 4000 изображений. Процент правильно идентифицированных ключевых точек составил 91,3. Основные ошибки при выделении ключевых точек были отмечены на слишком тёмных и не контрастных снимках.

Нужно отметить, несмотря на то, что SURF используется для поиска объектов на изображении, он сам работает не с объектами. SURF никак не выделяет объект из фона. Он рассматривает изображение как единое целое и ищет особенности этого изображения. При этом особенности могут быть как внутри объекта, так и на фоне, а также на точках границы объекта и фона. В связи с этим метод плохо работает для объектов простой формы и без ярко выраженной текстуры. Внутри таких объектов метод скорее всего не найдёт особых точек. Точки будут найдены либо на границе объекта с фоном, либо вообще только на фоне. А это приведёт к тому, что объект не сможет быть распознан в другом изображении, на другом фоне.

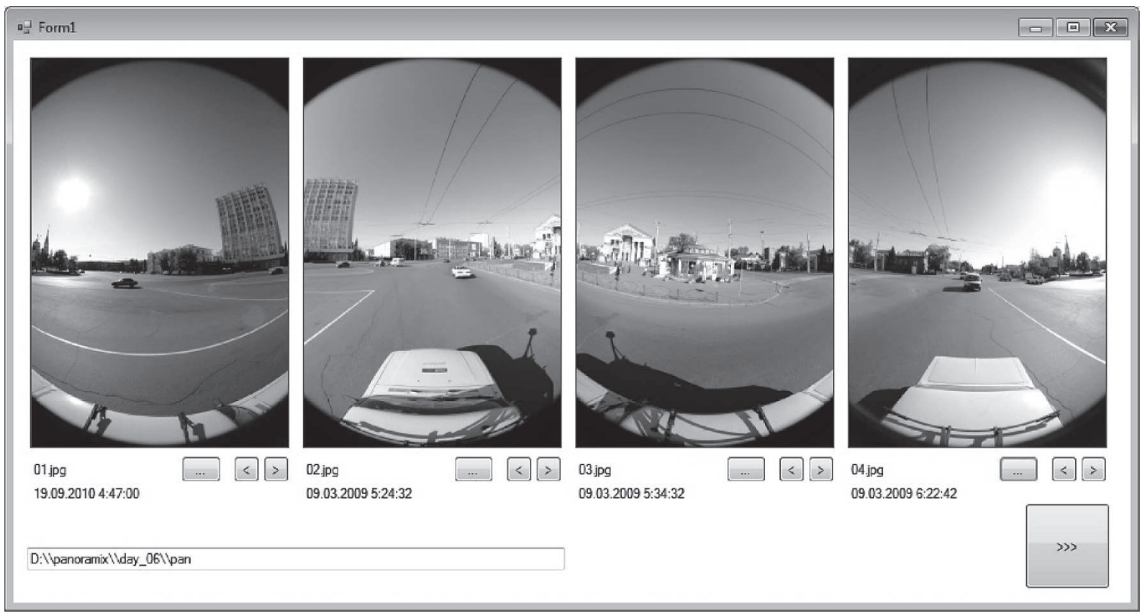

Рис. 3. Исходные изображения с камеры

Рис. 4. Готовый панорамный снимок

В рамках задачи идентификации точек на изображениях при создании сферических панорамных снимков проблема с фоном не возникает, поэтому метод SURF применим для решения этой задачи.

Литература

1. Русинов М. М. Композиция оптических систем. Л.: Машиностроение, 1989.
2. Bay H., Ess A., Tuytelaars T., Gool L. V. Speeded-Up Robust Features (SURF) // Proceedings of the 9th European Conference on Computer Vision. Springer LNCS. 2006. Vol. 3951. Pt. 1. P. 404–417.

Математические структуры и моделирование
2010, вып. 22, с. 101–113

УДК 004.056:004.057.4

СКРЫТЫЕ КАНАЛЫ ПЕРЕДАЧИ ИНФОРМАЦИИ В АЛГОРИТМЕ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ ГОСТ Р 34.10-2001

М. И. Атмашкин, С. В. Белим

Приведены различные способы организации скрытых каналов в алгоритме цифровой подписи ГОСТ Р 34.10-2001. Рассмотрены примеры использования скрытых каналов для обмена сообщениями, для тайной передачи ключа подписи, а также для создания криптографических протоколов. Представлен способ ликвидации скрытых каналов.

Введение

Многие схемы цифровой подписи имеют «особенность», позволяющую подписывающему спрятать в подписи некоторую информацию, которая может быть затем извлечена при знании дополнительного секрета. Причём получаемые таким образом подписи неотличимы от «обычных». Это свойство было обнаружено Симмонсом и названо [7] им *скрытым каналом* (англ. *subliminal channel*).

Данным свойством обладают схемы, в которых получаемая подпись зависит не только от подписываемого сообщения, но и от некоторого «случайного» числа. Перебирая различных кандидатов на место этого числа, подписывающий может управлять видом подписи и закодировать в ней некоторую информацию.

Такими, например, являются подписи: Ong-Schnorr-Shamir, ElGamal, DSA, ESIGN и другие [3]. Не является исключением [1] и действующий российский стандарт цифровой подписи ГОСТ Р 34.10-2001.

Симмонс приводит такую [5] классификацию скрытых каналов:

- *широкополосные* (англ. *broadband*) — количество бит в скрытом сообщении сравнимо с числом бит в случайном числе. Обычно скрываемое сообщение и является этим числом либо легко из него получается;

- *узкополосные* (англ. *narrowband*) — количество бит в скрытом сообщении значительно меньше количества бит в случайном числе. Как правило, в этом случае размер сообщения зависит не от размеров случайного числа, а от вычислительной мощности подписывающего или проверяющего.

Омский государственный университет им. Ф. М. Достоевского.
E-mail: AtmashkinMI@gmail.com, sbelim@mail.ru

Не стоит, однако, считать, что широкополосные каналы лучше. В большинстве случаев они обладают недостатком, в результате которого проверяющему становится известен закрытый ключ подписи. Узкополосные каналы этого недостатка лишены.

В данной работе предложены различные типы широкополосных и узкополосных каналов для алгоритма цифровой подписи ГОСТ Р 34.10-2001. Рассмотрены примеры организации криптографических протоколов на основе широкополосных скрытых каналов, а также представлена модификация алгоритма подписи, позволяющая ликвидировать скрытые каналы.

1. Алгоритм ЭЦП ГОСТ Р 34.10-2001

Российский стандарт [2] ГОСТ Р 34.10-2001 является одним из самых молодых алгоритмов цифровой подписи. Стандарт принят и введён в действие Постановлением Госстандарта России от 12 сентября 2001 года взамен ГОСТ Р 34.10-94 и основан на эллиптических кривых. Его стойкость основывается на сложности вычисления дискретного логарифма в группе точек эллиптической кривой, а также на стойкости функции хэширования ГОСТ Р 34.11-94.

1.1. Параметры схемы цифровой подписи

- простое число p — модуль эллиптической кривой такой, что $p > 2^{255}$;
- эллиптическая кривая E, задаваемая своим инвариантом $J(E)$ или коэффициентами $a, b \in F_p$, где F_p — конечное поле из p элементов;
- целое число m — порядок группы точек эллиптической кривой, m должно быть отлично от p;
- простое число q — порядок некоторой циклической подгруппы группы точек эллиптической кривой, т. е. $q \mid m$. Кроме того, $2^{254} < q < 2^{256}$;
- точка P эллиптической кривой E, являющаяся генератором подгруппы порядка q, т. е. $qP = O$ и $kP \neq O$ для всех $k = 1, 2, \ldots, q-1$, где точка O является нейтральным элементом группы;
- $h(M)$ — функция хэширования (ГОСТ Р 34.11-94), которая отображает сообщения M в двоичные вектора длины 256 бит.

1.2. Используемые ключи

- ключ подписи — целое число d, удовлетворяющее неравенству $0 < d < q$;
- ключ проверки — точка эллиптической кривой Q, удовлетворяющая равенству $dP = Q$.

1.3. Дополнительные требования

- должно быть выполнено условие $p^t \neq 1 \ (mod\ q)$, для всех целых чисел $t = 1, 2, \ldots, B$, где B удовлетворяет неравенству $B \geq 31$;
- инвариант кривой должен удовлетворять условию $J(E) \neq 0$ или 1728.

1.4. Формирование цифровой подписи

1. Вычисление хэш-функции от сообщения M: $z = h(M)$.
2. Вычисление $e = z \ (mod\ q)$, и если $e = 0$, положить $e = 1$.
3. Генерация случайного числа k такого, что $0 < k < q$.
4. Вычисление точки эллиптической кривой $C = kP$ и по ней нахождение $r = x_C \ (mod\ q)$, где x_C — это координата x точки C. Если $r = 0$, то возвращаемся к предыдущему шагу.
5. Нахождение:
$$s = rd + ke \ (mod\ q), \tag{1}$$
если $s = 0$, то возвращаемся к шагу (3).
6. Формирование цифровой подписи $\zeta = (r \mid\mid s)$.

1.5. Проверка цифровой подписи

1. Вычисление по цифровой подписи ζ чисел r и s. Если хотя бы одно из неравенств $0 < r < q$ и $0 < s < q$ неверно, то подпись неправильная.
2. Вычисление хэш-функции от сообщения M: $z = h(M)$.
3. Вычисление $e = z \ (mod\ q)$, и если $e = 0$, положить $e = 1$.
4. Вычисление $\nu = e^{-1} \ (mod\ q)$.
5. Вычисление $z_1 = s\nu \ (mod\ q)$ и $z_2 = -r\nu \ (mod\ q)$.
6. Вычисление точки эллиптической кривой $C = z_1P + z_2Q$. И определение $R = x_C \ (mod\ q)$, где x_C — координата x точки C.
7. В случае равенства $R = r$ подпись правильная, иначе — неправильная.

2. Скрытые каналы

Поскольку в алгоритме формирования цифровой подписи присутствует случайное число k, то подписывающий может надлежащим выбором этого числа придать подписи ζ некоторый «особый» вид, закодировав в ней сообщение.

Рассмотрим скрытые каналы согласно классификации, данной Симмонсом.

2.1. Широкополосные каналы

В простейшем случае можно подставить само сообщение вместо числа k. Алгоритмы формирования и проверки подписи принципиально не изменятся, и тот, кто знает ключ проверки, может по-прежнему проверить правильность подписи. Но если проверяющему известен также ключ подписи d, то он может вычислить k по формуле, вытекающий из (1):

$$k = e^{-1}(s - rd) \ (mod\ q),$$

а если ключ подписи неизвестен, то для нахождения k нужно решить проблему дискретного логарифма в группе точек эллиптической кривой.

Несмотря на максимальную пропускную способность, данный канал обладает серьёзными недостатками:

1. Проверяющему необходимо знать ключ подписи. Это является очень серьёзным недостатком, потому что в этом случае для подписывающего существует угроза компрометации его ключа подписи, и он должен полностью довериться своему собеседнику. Кроме того, в ряде случаев заранее невозможно узнать ключ подписи. Стоит также отметить, что в общем случае число k будет некоторой обратимой функцией от скрываемого сообщения, и знание сообщения влечёт за собой знание числа k, из которого легко получается ключ подписи:

$$d = r^{-1}(s - ke) \ (mod\ q), \tag{2}$$

   т. е. обмен «большими» сообщениями влечёт знание ключа подписи.

2. Стойкость алгоритма ГОСТ Р 34.10-2001 сильно зависит от качества используемого ГПСП, потому что при повторе числа k можно легко вычислить ключ подписи d. Так как число r зависит только от k, то оно будет в обоих случаях одинаковым, и получается такая система:

$$s_1 = rd + ke_1 \ (mod\ q), s_2 = rd + ke_2 \ (mod\ q), \tag{3}$$

   из которой следует формула для k:

$$k = (s_2 - s_1)(e_2 - e_1)^{-1} \ (mod\ q),$$

   а зная k, можно уже найти d по формуле (2).

   В случае использования скрытого канала данная ситуация становится очень вероятной — примером может быть обмен регулярными дежурными сообщениями. Чтобы исправить этот недостаток, можно использовать приём «подсаливания» числа k — замены части его бит на случайные. При извлечении скрытого сообщения собеседник будет просто игнорировать эти биты. Данный приём плох тем, что жертвуется пропускная способность скрытого канала, причём число подменяемых бит должно быть довольно велико для защиты от атаки перебором. Если злоумышленник

знает, что два случайных числа в системе (3) отличаются на небольшое число n бит в известных позициях, то он может попытаться решить 2^n систем линейных уравнений. Для усложнения схемы можно постоянно менять позиции заменяемых бит по некоторому заранее известному закону.

Но есть и более безопасный метод, который не поглощает пропускной способности канала. Можно вместо числа k использовать: $k'_i = k + h_i \ (mod\ q)$, где число h_i является действительно случайным. Получатель для восстановления исходного сообщения вычислит: $k = k'_i - h_i \ (mod\ q)$. Понятно, что последовательность h_i должна быть известна обоим. Этого можно достичь, например, используя общие секретные параметры для криптографически стойкого ГПСП. В качестве такого генератора может выступать генератор на основе функций хэширования. Пусть $H(x)$ — это 256-битная хэш-функция, например, это может быть ГОСТ Р 34.11-94 или SHA-256. Тогда ПСП будет выглядеть так:

$$h_0 = H(d), h_i = H(h_{i-1} \ || \ d), i = 1, 2, \ldots \tag{4}$$

Очевидно, последовательность легко порождается по известному ключу подписи, но без его знания вычислительно трудно получить как следующие, так и предыдущие члены последовательности. Кроме знания ключа подписи собеседникам не нужно иметь других секретов. Получающиеся в результате подписи будут неотличимы от «обычных».

3. Не всякое сообщение, которому соответствует некоторое число $0 < k < q$, можно встроить в подпись. Это связано с ограничениями (4) и (5) шагов алгоритма формирования подписи: $r \neq 0 \ (mod\ q)$ и $s \neq 0 \ (mod\ q)$. Подобные подписи отбрасываются на шаге (1) алгоритма проверки подписи. Причём заранее сказать, для каких k не получится создать подпись, тяжело из-за сложной зависимости r и s от k.

Используя некоторое обобщение системы (3), можно реализовать схему генерации чисел k, которая позволит при знании некоторого секрета узнать ключ подписи. Например, можно использовать такую последовательность k_i:

$$k_0, k_i = a_i k_{i-1} + b_i \ (mod\ q), i = 1, 2, \ldots,$$

где k_0 — зерно, полученное из некоторого внешнего источника энтропии, а последовательности a_i и b_i построены аналогично (4) на основе ключей d_a и d_b соответственно. Для тех, кто не знает этих ключей, последовательность k_i является стойкой криптографической ПСП. Но тот, кому известны ключи d_a и d_b, может перехватить две последовательные подписи и, зная номер i, вычислить a_i и b_i, а затем решить систему:

$$s_{i-1} = r_{i-1} d + k_{i-1} e_{i-1} \ (mod\ q), s_i = r_i d + (a_i k_{i-1} + b_i) e_i \ (mod\ q). \tag{5}$$

Она линейная, и в ней только два неизвестных: d и k_{i-1}, которые легко находятся. Для «простых» перехватчиков неизвестных в этой системе слишком много. Имея даже большое количество перехваченных подписей, вычислительно трудно решить систему из-за сложной связи между членами в ПСП a_i и b_i.

3. Узкополосные каналы

Основным преимуществом узкополосных каналов является отсутствие необходимости раскрывать ключ подписи. К недостаткам же можно отнести повышенные требования к вычислительным ресурсам — памяти и процессорному времени. Именно ограниченностью этих ресурсов и обусловлена низкая пропускная способность этих каналов. Узкополосные скрытые каналы можно условно разделить на вероятностные и детерминированные.

3.1. Вероятностные каналы

Ключевой особенностью этих каналов является то, что требуемые скрытые сообщения можно получить только с некоторой вероятностью. Чем больше бит будет в сообщении, тем меньше вероятность его получить. В общем случае алгоритм создания вероятностного канала выглядит так [4]:

1. Подписывающий и проверяющий договариваются о некоторой секретной ключевой функции $h_K()$, которая отображает подпись ζ или её часть (r или s) в некоторую последовательность бит M, а также о некотором способе кодирования сообщения такой последовательностью.

2. Подписывающий генерирует случайные числа k, формирует для каждого из них цифровую подпись по стандартному алгоритму и вычисляет значения функции $h_K()$ от полученных подписей до тех пор, пока не получит последовательность бит M, соответствующую встраиваемому сообщению, либо пока не кончится отведённое на подпись время — в этом случае встроить сообщение не удастся.

3. После проверки подписи получатель, зная секретный ключ K и алгоритм декодирования последовательности M, извлекает секретное сообщение.

В качестве примера можно привести канал, предложенный Симмонсом, для DSA [6]. Его адаптация для подписи ГОСТ Р 34.10-2001 будет выглядеть так. Секретным ключом K являются n различных больших простых чисел. Функция $h_K()$ ставит в соответствие части r подписи последовательность M, состоящую из n бит, где i-й бит равен «1», если r является квадратичным вычетом по модулю i-го простого числа, и «0» — в противном случае. Последовательность M без дополнительных преобразований и будет являться секретным сообщением. Вероятность того, что r будет квадратичным вычетом по модулю некоторого простого числа, составляет $\frac{1}{2}$, для n чисел соответственно — $\frac{1}{2^n}$. Чем больше подписей в единицу времени может генерировать подписывающий, тем большую длину скрытого канала он может себе позволить.

На шаге (1) алгоритма создания вероятностного канала действительно удобнее использовать часть r подписи, потому что она вычисляется первой. В качестве функции $h_K()$ можно, например, использовать следующие функции:

- хэш-функцию ГОСТ Р 34.11-94 со стартовым вектором хэширования, равным K, либо другую ключевую хэш-функцию;

- возведение числа r в секретную степень, равную K, по модулю числа q. Этот способ основан на проблеме дискретного логарифмирования в кольце вычетов по модулю простого числа.

- умножение точки эллиптической кривой C, получаемой на шаге (4) алгоритма формирования подписи, на секретное число K, в качестве значения функции будет выступать координата x полученной точки. Этот способ основан на проблеме дискретного логарифмирования в группе точек эллиптической кривой.

В качестве способа кодирования значения предложенных функций $h_K()$ в скрытое сообщение можно использовать остаток от деления на степень 2, который равен младшим битам числа. Если допустить, что все остатки будут равновероятны, то вероятность получения конкретного сообщения будет такая же, как в примере Симмонса.

На шаге (2) алгоритма создания вероятностного канала можно использовать следующую оптимизацию. Вместо генерации нескольких случайных чисел k можно сгенерировать одно, а затем прибавлять к полученной точке $C = kP$ точку P, пока не получим нужного значения функции $h_K()$. Этот способ не уменьшает безопасность подписи, но позволяет заменить многократное умножение точек на более простое сложение.

3.2. Детерминированные каналы

Данный тип скрытых каналов позволяет гарантированно создавать требуемые скрытые сообщения, но требует от лица, извлекающего сообщения, большой процессорной мощности и дополнительной памяти. Алгоритм выглядит так:

1. Подписывающий и проверяющий договариваются о размере скрываемых сообщений. Подписывающий генерирует ключ для ПСП, построенной по принципу (4), и сохраняет его в своей постоянной памяти. Затем он генерирует большое число n членов этой ПСП, где n — число сообщений, которые он сможет отправить по этому каналу. Для каждого из этих чисел подписывающий вычисляет точку C из шага (4) алгоритма формирования подписи. Все n полученных точек подписывающий отправляет проверяющему.

2. Подписывающий начинает формировать подписи по стандартному алгоритму, но случайные числа он берет из ПСП, восстановленной по сохранённому ключу. Вместо точки C он использует точку $C' = C + mP$, где $0 \leq m < 2^n$ является скрываемым сообщением, длиной n бит.

3. После проверки подписи проверяющий начинает прибавлять к очередной точке C из точек, переданных ему на шаге (1), точку P до тех пор, пока не получит точку C'. По числу операций сложения он восстановит скрытое сообщение m.

В данном алгоритме проверяющий так и не узнает ни одного случайного числа k из секретной ПСП. Следовательно, он не сможет вычислить ключ подписи по формуле (2). Проверяющему сложно установить зависимость между переданными ему точками, а также узнать по конкретной точке использованное случайное число. Недостатком данного алгоритма является необходимость передавать большой объем информации на шаге (1), а также неравномерность во времени, затрачиваемом на декодирование получателем различных сообщений. Эту неравномерность можно сгладить, если начинать проверку не с точки C, а с точки, сдвинутой от неё на некоторое случайное число точек P, и в дальнейшем осуществлять сложение с точками P циклически с учётом границ сообщения m.

4. Ликвидация скрытых каналов

Все представленные скрытые каналы основаны на том, что подписывающий может самостоятельно выбирать случайное число k. С другой стороны, нельзя доверить кому-то генерацию этого числа, так как это неминуемо раскроет ключ подписи. Очевидным и наиболее подходящим решением будем совместная генерация числа k таким образом, что подписывающий не сможет контролировать ни один бит числа k, а контроллер не сможет узнать ни один бит числа k. В конце протокола контроллер должен убедиться, что подписывающий использовал именно то число, которое они сгенерировали вместе. Адаптация протокола, предложенного Симмонсом для DSA [5], для подписи ГОСТ Р 34.10-2001 будет выглядеть так:

1. Подписывающий генерирует число k' и отправляет контроллеру точку $U = k'P$.

2. Контроллер выбирает случайное число k'' и отправляет подписывающему.

3. Подписывающий использует для создания подписи число $k = k'k'' \ (mod\ q)$ и отправляет полученную подпись контроллеру.

4. Контроллер проверяет подпись, а также то, что координата x точки $k''U$ сравнима с r по модулю q.

Протокол Симмонса имеет серьёзный недостаток — на шаге (2) контроллер может сам управлять частью r подписи и встроить в неё собственное скрытое сообщение. Для предотвращения этого можно сделать следующую модификацию протокола. На шаге (1) подписывающий вместо точки U должен отправить значение хэш-функции от этой точки. Контроллер умножает точку C, получаемую на шаге (6) алгоритма проверки цифровой подписи, на $k''^{-1} \ (mod\ q)$, считает от неё значение хэш-функции и сверяет с полученным от подписывающего. В таком варианте протокола ни подписывающий, ни контроллер не смогут заранее узнать, каким будет число r. Но применение контроллера для создания подписей возможно далеко не всегда, и от скрытых каналов полностью избавиться нельзя.

5. Криптографические протоколы

В то время как узкополосные каналы удобнее для организации утечки ключа подписи в непроверенных реализациях цифровой подписи, широкополосные каналы можно использовать в криптографических протоколах для легальной передачи сеансовых ключей, причём сам факт передачи ключей может быть скрыт. Скрытый канал может выполнять функции шифра, при этом сохранятся все «обычные» свойства цифровой подписи. Таким образом в подписи можно зашифровать сеансовый ключ, а проверяющий сможет по-прежнему проверить целостность сообщения и его авторство. В широкополосном канале можно передать сеансовый ключ размером 256 бит, что достаточно для таких симметричных шифров, как ГОСТ 28147-89 или AES-256.

Будем использовать стандартные [3] имена для участников протоколов: Алиса — сторона, инициирующая сеанс, Боб — сторона, с которой устанавливается сеанс, Трент — доверенная промежуточная сторона.

Подпись будем обозначать так: $S_d(k, M)$, где d — ключ подписи; k — случайное число, используемое для формирования подписи; M — подписываемое сообщение. Вместе с подписью в протоколах подразумевается передача самого сообщения M. Представленные протоколы предполагают синхронизацию часов всех участников. Погрешность синхронизации компенсируется запоминанием истории передач в течение некоторого времени. Передача меток времени вместе с временем жизни ключей защищает от повторной передачи сообщений.

5.1. Протокол № 1

Рассмотрим первый протокол, который использует простейший широкополосный канал с общим ключом подписи. Он является модифицированным вариантом базовой версии протокола *Kerberos* [3]. Вместо шифрования здесь используется цифровая подпись со скрытым каналом.

Пусть Трент имеет общий ключ подписи с Алисой — d_A и с Бобом — d_B. Данный протокол позволяет обменяться сеансовым ключом, который сгенерировал Трент, а также провести взаимную аутентификацию.

1. Алиса отправляет Тренту сообщение с идентификаторами: A, B.

2. Трент создаёт два сообщения, состоящих из метки времени T_T, времени жизни ключа L и идентификаторов. В сообщении для Алисы используется идентификатор Боба, а для Боба наоборот. Затем он подписывает эти сообщения ключами d_A и d_B. В качестве случайного числа в обеих подписях используется сгенерированный им сеансовый ключ K, зашифрованный соответственно ключами d_A и d_B. Полученные подписи Трент отправляет Алисе: $S_{d_A}(E_{d_A}(K), B \mid\mid T_T \mid\mid L), S_{d_B}(E_{d_B}(K), A \mid\mid T_T \mid\mid L)$.

3. Алиса проверяет первую подпись Трента, убеждается, что сообщение текущее, извлекает и расшифровывает сеансовый ключ. Затем она создаёт сообщение, состоящее из её идентификатора и метки времени, шифрует

его сеансовым ключом и отправляет Бобу. Алиса также посылает Бобу вторую подпись Трента: $E_K(A \,||\, T_T), S_{d_B}(E_{d_B}(K), A \,||\, T_T \,||\, L)$.

4. Боб проделывает с подписью Трента те же операции, что и Алиса, затем он расшифровывает сообщение Алисы и убеждается в том, что оно текущее. Создаёт сообщение, состоящее из метки времени плюс единица, шифрует его сеансовым ключом и отправляет Алисе: $E_K(T_T + 1)$.

5. Алиса расшифровывает сообщение Боба и проверяет метку времени.

6. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Шифрование сеансового ключа необходимо для того, чтобы Алиса не смогла вычислить ключ d_B по формуле (2). Кроме того, это делает части r передаваемых подписей различными. Злоумышленник, перехватывающий сообщения, может даже не узнать, что в этом протоколе, кроме аутентификации, происходит обмен сеансовым ключом. Главным преимуществом по сравнению с *Kerberos* является аутентификация Трента, обеспечиваемая цифровыми подписями.

5.2. Протокол № 2

Рассмотрим теперь протокол, основанный на «недостатке» широкополосных каналов, связанном с повтором случайного числа и последующим вычислением ключа подписи. Цифровая подпись может использоваться здесь для разделения секрета, в роли которого будет выступать сеансовый ключ.

Пусть Тренту известны открытые ключи Алисы — A_{open} и Боба — B_{open}, используемые в некоторой схеме асимметричного шифрования. Алисе и Бобу известен ключ проверки подписи $S_{main}()$ Трента. Эта подпись будет использоваться для аутентификации Трента и может вычисляться по другому алгоритму. Скрытый канал в этой подписи, если он есть, использоваться не будет.

1. Алиса отправляет Тренту сообщение с идентификаторами: A, B.

2. Трент генерирует сеансовый ключ K, а также ключи подписи и проверки для алгоритма со скрытым каналом — T_{close} и T_{open}. Трент составляет сообщение из подписи $S_{main}()$ ключа T_{open}, а также подписи сообщения, состоящего из идентификатора Боба, открытого ключа Боба, метки времени и времени жизни сеансового ключа, сделанной на ключе T_{close}. В качестве случайного числа в подписи используется ключ K. Полученное сообщение Трент шифрует открытым ключом Алисы и отправляет ей: $E_{A_{open}}(S_{main}(T_{open}) \,||\, S_{T_{close}}(K, B \,||\, B_{open} \,||\, T_T \,||\, L))$.

3. Трент проделывает те же действия для Боба и отправляет ему аналогичное сообщение: $E_{B_{open}}(S_{main}(T_{open}) \,||\, S_{T_{close}}(K, A \,||\, A_{open} \,||\, T_T \,||\, L))$.

4. Алиса расшифровывает сообщение Трента, проверяет подписи, убеждается в том, что сообщение текущее. Затем она зашифровывает открытым ключом Боба подпись Трента со скрытым каналом и отправляет Бобу: $E_{B_{open}}(S_{T_{close}}(K, B \,||\, B_{open} \,||\, T_T \,||\, L))$.

5. Аналогично Боб отправляет Алисе: $E_{A_{open}}(S_{T_{close}}(K, A \mid\mid A_{open} \mid\mid T_T \mid\mid L))$.

6. Алиса и Боб расшифровывают и проверяют принятые подписи. Зная две подписи Трента, каждый из них может восстановить сеансовый ключ K.

7. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Ключ подписи со скрытым каналом является одноразовым, чтобы участники не могли узнать сеансового ключа после шагов (2) и (3). По той же причине используется шифрование с открытым ключом вместо симметричного шифрования. Без подписи $S_{main}()$ Алиса или Боб могли бы выдавать себя в дальнейшем за Трента. Ключ для этой подписи может быть долговременным.

5.3. Протокол № 3

Не всегда приемлем тот факт, что сеансовый ключ генерирует третья сторона, в дальнейшем ей будет доступна вся переписка. Более привлекательной является совместная генерация ключа Алисой и Бобом. Алиса генерирует ключ K_A, Боб соответственно — K_B, сеансовый ключ получается в результате сложения по модулю 2 этих ключей: $K = K_A \oplus K_B$. Совместную генерацию ключа можно организовать, например, используя формулу (5). Обменявшись одной подписью, участники протокола не смогут в дальнейшем отказаться от выбранного ключа. После обмена второй подписью можно будет вычислить сеансовый ключ собеседника и убедиться в его подлинности.

Пусть Алиса знает ключи, используемые Бобом для генерации последовательностей a_i^B и b_i^B, а Боб знает ключи Алисы для a_i^A и b_i^A. Также они имеют общий ключ шифрования K_{AB}, используемый для взаимной аутентификации.

1. Алиса генерирует свой ключ K_A, вычисляет ключ подписи $d_A = E_{K_{AB}}(K_A)$ и соответствующий ему ключ проверки Q_A. Затем она формирует сообщение, состоящее из ключа проверки, текущего номера в ПСП i_A и подписи своего идентификатора, метки времени T_A и времени жизни L_A ключа K_A. В качестве случайного числа в этой подписи используется ключ K_A. Алиса отправляет Бобу: $Q_A, i_A, S_{d_A}(K_A, A \mid\mid T_A \mid\mid L_A)$.

2. Боб проверяет подпись Алисы и проделывает те же действия. Он отправляет Алисе: $Q_B, i_B, S_{d_B}(K_B, B \mid\mid T_B \mid\mid L_B)$.

3. Алиса проверяет подпись Боба и создаёт новую подпись сообщения, состоящего из ключа проверки Боба и его номера в ПСП. В качестве случайного числа она использует $K'_A = a^A_{i_A} K_A + b^A_{i_A}$ и отправляет полученную подпись Бобу: $S_{d_A}(K'_A, Q_B \mid\mid i_B)$.

4. Боб вычисляет $K'_B = a^B_{i_B} K_B + b^B_{i_B}$ и отправляет Алисе аналогичную подпись: $S_{d_B}(K'_B, Q_A \mid\mid i_A)$.

5. Алиса и Боб проверяют полученные подписи. Зная номера текущих членов ПСП друг друга, они могут решить систему (5) и вычислить по ней ключ

подписи и использованное случайное число. Для взаимной аутентификации они проверяют, что ключ подписи — это зашифрованное случайное число. Затем они вычисляют сеансовый ключ: $K = K_A \oplus K_B$.

6. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Злоумышленник не сможет решить систему (5), он также не сможет подменить подписи или номера в ПСП, потому что ему неизвестен ключ K_{AB}. Ни Алиса, ни Боб не смогут вычислить ключ собеседника до получения второй подписи и не смогут отказаться от выбранного сеансового ключа, потому что в случае подмены ключа не произойдёт аутентификации.

5.4. Протокол № 4

Можно провести совместную генерацию сеансового ключа и взаимную аутентификацию по аналогии с алгоритмом ликвидации скрытого канала.

Пусть Алиса и Боб имеют общий ключ подписи K_{AB}.

1. Алиса генерирует свой ключ K_A, вычисляет хэш-код ключа и отправляет его Бобу вместе со своим идентификатором: $A, H(K_A)$.

2. Боб генерирует свой ключ K_B и использует его в качестве случайного числа для создания подписи сообщения, состоящего из своего идентификатора, метки времени T_B и времени жизни L_B ключа K_B. Ключом подписи является общий ключ K_{AB}. Полученную подпись Боб отправляет Алисе: $S_{K_{AB}}(K_B, B \mid\mid T_B \mid\mid L_B)$.

3. Алиса проверяет подпись Боба, извлекает из подписи его ключ K_B, вычисляет сеансовый ключ $K = K_A \oplus K_B$ и отправляет Бобу аналогичную подпись: $S_{K_{AB}}(K, A \mid\mid T_A \mid\mid L_A)$.

4. Боб проверяет подпись Алисы, извлекает из подписи сеансовый ключ K, вычисляет по нему ключ Алисы K_A и убеждается, проверяя его хэш-код, что Алиса не отказалась от первоначального ключа.

5. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Цифровая подпись обеспечивает взаимную аутентификацию Алисы и Боба. Злоумышленник не сможет подменить подписи, потому что он не знает ключа K_{AB}. Алисе вычислительно трудно найти другой ключ, дающий тот же хэш-код, а Бобу вычислительно трудно узнать ключ Алисы до того, как он отправит ей свой. Для этого ему нужно обратить хэш-функцию.

6. Заключение

Представленные в статье скрытые каналы предоставляют множество возможностей для тайной и безопасной передачи информации в стандартных цифровых подписях ГОСТ Р 34.10-2001. В случае передачи сообщений широкополосные

каналы стоит применять, когда источник и приёмник сообщений — это одно и то же лицо либо подразделения одной организации. В этих случаях не произойдёт компрометации ключа подписи. Узкополосные каналы предпочтительнее, когда приёмником является лицо с более низкой степенью доверия либо группа лиц. Нахождение ключа подписи для получателей будет в этом случае вычислительно трудной задачей. Оба типа скрытого канала при правильном применении доступны лишь авторизованным получателям и необнаружимы для посторонних лиц, не обладающих дополнительным секретом.

На широкополосных каналах можно строить криптографические протоколы. Цифровая подпись с таким каналом будет выполнять и функции подписи, и функции шифра. Также с помощью таких цифровых подписей можно проводить совместную генерацию ключа одновременно с взаимной аутентификацией. Размер скрытого канала достаточен для большинства симметричных шифров.

Узкополосные каналы можно использовать для создания реализаций подписи, организующих утечку конфиденциальной информации. Следует особенно тщательно проверять используемые программные и аппаратные средства для создания цифровых подписей и генерации случайных чисел, полученные от третьих лиц, возможно, даже имеющих лицензию на их изготовление. Предпочтительнее исследование исходных кодов используемых программ и их самостоятельная компиляция. В случае, когда исходные коды недоступны либо необходимы дополнительные гарантии проверяющему, можно провести совместную генерацию подписи с контроллером по предложенному протоколу.

Литература

1. Белим С. В., Федосеев А. М. Исследование скрытых каналов передачи информации в алгоритме цифровой подписи ГОСТ Р 34.10-2001 // Известия Челябинского научного центра. 2007. Вып. 2. С. 55–57.
2. ГОСТ Р 34.10-2001. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. М.: Изд-во стандартов, 2001. 16 с.
3. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2002. 816 с.
4. Kobara K., Imai H. On the Channel Capacity of Narrow-band Subliminal Channels // In Proc. of ICICS '99. 1999. Vol. 1726. P. 309–323.
5. Simmons G. J. Subliminal Channels: Past and Present // European Transactions on Telecommunications. 1994. Vol. 4. No. 4. P. 459–473.
6. Simmons G. J. Subliminal Communication is Easy Using the DSA // Advances in Cryptology — EUROCRYPT '93 Proceedings. Springer-Verlag. 1994. P. 218–232.
7. Simmons G. J. The Prisoner's Problem and the Subliminal Channel // Advances in Cryptology: Proceedings of CRYPTO '83. Plenum Press. 1984. P. 51–67.

Математические структуры и моделирование
2010, вып. 22, с. 114–120

УДК 004.056

ЭЛЕМЕНТАРНЫЕ ОПЕРАТОРЫ ПОСТРОЕНИЯ РОЛЕВОЙ ПОЛИТИКИ БЕЗОПАСНОСТИ

С. В. Белим, Н. Ф. Богаченко

Строится набор элементарных операторов, позволяющий модифицировать дерево ролей и отслеживать передачу прав доступа.

Введение

Ролевое разграничение доступа к информации получило широкое распространение не только в системах управления базами данных, но и в операционных системах. Основное отличие ролевой политики безопасности от мандатной и дискреционной состоит в возможности управления привилегиями. Под привилегией понимается возможность осуществления некоторых действий в системе в целом. Безусловно, для получения привилегии необходимо разрешение на доступ к некоторым служебным объектам системы, которое может регламентироваться любой политикой безопасности. Однако ролевое разграничение доступа позволяет наиболее естественно организовать сопоставление привилегий отдельным пользователям или группам пользователей, а также осуществить наследование привилегий между ролями.

Ролевую политику безопасности принято анализировать исходя из иерархии ролей, наиболее удобным представлением которой являются ориентированные графы, называемые далее ролевыми графами (если исходя из контекста понятно, что речь идёт об ориентированном графе, то орграф будем называть просто графом). Однако на сегодняшний день отсутствует формальный подход, позволяющий на основе некоторого набора формальных операций отслеживать любые преобразования ролевого графа. Целью данной статьи является создание модели ролевого разграничения доступа, исходя из набора примитивных операторов по аналогии с моделью HRU [4, с. 48].

1. Безопасность системы

Будем считать, что в системе существует некоторое множество ролей R, наделённых полномочиями из множества P. Далее всюду будем считать, что множества R и P конечны. Между ролями задана иерархия, определяемая ориентированным графом G. Вершины данного графа соответствуют ролям, а дуги —

Омский государственный университет им. Ф. М. Достоевского.
E-mail: sbelim@omsu.ru

наследованиям. То есть, если в графе присутствует дуга от вершины r к вершине r', значит, роль r авторизована на роль r'. Авторизация одной роли на другую подразумевает полное наследование её привилегий. Очевидно, что роль r может унаследовать только привилегии ролей, связанных с ней ориентированными путями. Более строго, если существует ориентированный путь $p(r, r')$ в графе G от роли-вершины r до роли-вершины r', то роль r наследует привилегии r'. Будем обозначать набор привилегий роли r через $r.p$. Тогда предыдущее утверждение может быть записано следующим образом:

$$\text{if } \exists\, p(r, r') \Rightarrow r.p \supseteq r'.p.$$

Введём обозначение для множества ролей, до которых существует путь от вершины r в ролевом графе G:

$$PR(r) = \left\{ r' \middle| \exists\, p(r, r') \right\}.$$

Как легко понять, это множество тех ролей, от которых привилегии передаются роли r.

Перейдём теперь к вопросам безопасности системы с ролевым разграничением доступа.

Определение 1. Будем считать, что происходит ***утечка привилегии*** p, если роль r получает её несанкционированно.

Определение 2. Система с ролевым разграничением доступа ***безопасна***, если в ней не происходит утечка привилегий.

Следует отметить, что данное выше определение безопасности является алгоритмическим. По сути, система будет безопасной, если возможна реализация алгоритма, следящего за передачей привилегий в системе. Задача проверки безопасности системы сводится к исследованию возможности передачи привилегий по дугам графа. Как было сказано выше, привилегии роли r могут передаваться только от ролей из множества $PR(r)$. То есть роли из множества $PR(r)$ влияют на r.

Определение 3. Будем говорить, что роль r' ***не влияет*** на роль r (обозначение $r' : |r$), если добавление любой привилегии p в множество привилегий $r'.p$ не приводит к изменению множества привилегий $r.p$.

Легко понять, что $(R \setminus PR(r)) : |r$.

Теорема 1. *Если в системе ролевое дерево неизменно, то система является безопасной.*

Доказательство. Для доказательства данной теоремы достаточно показать, что для любой роли r алгоритм поиска множества $PR(r)$ будет конечным, а также будет конечным само множество $PR(r)$. Конечность множества $PR(r)$

вытекает из того, что $PR(r) \subseteq R$, а множество R конечно. Поиск же множества $PR(r)$ может быть осуществлён с помощью построения матрицы достижимости [6, с. 176] для ориентированного графа G и имеет полиномиальную сложность [5, с. 48]. Таким образом строится множество $PR(r)$, которое является конечным и неизменным, а затем осуществляется контроль за тем, чтобы нелегитимная привилегия не была внесена в список привилегий как самой роли r, так и всех ролей из множества $PR(r)$, что может быть осуществлено прямым перебором за конечное число шагов. ■

Определение 4. Подграф $GR(r)$ ролевого графа G, вершинами которого является множество $PR(r)$, а дугами — соответствующие дуги между этими вершинами из графа G, будем называть ***графом влияния*** на роль r.

Предложение 1. *Если ролевой граф G является деревом, граф влияния $GR(r)$ на произвольную роль r будет также деревом.*

Доказательство. По свойствам ориентированного дерева [5, с. 239] подграф, определяемый множеством узлов, достижимых из вершины r, является ордеревом с корнем r. ■

Предложение 2. *Если ролевой граф G является решёточным, граф влияния $GR(r)$ на произвольную роль r будет также решёточным.*

Доказательство. Согласно [3], решёточный граф — это ориентированный граф, вершины которого образуют решётку, при этом отношение порядка определяется ориентированными путями графа.

В графе влияния $GR(r)$ для любой вершины r_i существует ориентированный путь $p(r, r_i)$ по определению. Следовательно, $\sup\{r, r_i\} = r$, $\inf\{r, r_i\} = r_i$.

Пусть теперь r_i и r_j — две различные вершины графа влияния $GR(r)$, отличные от r. Очевидно, что $\inf\{r_i, r_j\} \in GR(r)$, так как в этом графе содержатся все вершины, достижимые из r, а значит, и достижимые из r_i и r_j. Пусть $\sup\{r_i, r_j\} = r_s$. Докажем, что $r_s \in GR(r)$. Допустим это не так. По определению графа влияния: $r \geqslant r_i$ и $r \geqslant r_j$, следовательно, r является верхней гранью этих вершин. Так как r_s — наименьшая из всех верхних граней, то $r \geqslant r_s$, следовательно, существует ориентированный путь $p(r, r_s)$ — противоречие с тем, что $r_s \notin GR(r)$.

Таким образом, для любых двух вершин графа влияния $GR(r)$ наибольшая нижняя и наименьшая верхняя грани также принадлежат этому графу, следовательно $GR(r)$, — решёточный. ■

Для анализа путей передачи привилегий между ролями не обязательно рассматривать граф влияния полностью, можно удалить некоторые дуги, дублирующие друг друга при передаче привилегий роли r. При таком преобразовании могут «потеряться» некоторые привилегии ролей из множества $PR(r)$, однако перед нами стоит задача обеспечения безопасности роли r. Такую процедуру преобразования будем называть ***оптимизацией графа влияния***. Очевидно, что в результате оптимизации можно получить разные графы. Оптимизированный граф влияния с минимальным количеством дуг будем называть ***минимальным***.

Теорема 2. *Минимальный граф влияния является деревом.*

Доказательство. По определению, любая вершина r' графа влияния $GR(r)$ достижима из вершины r (существует ориентированный путь $p(r, r')$).

Как было показано в работе [2], в ролевом графе G отсутствуют ориентированные циклы. Отсюда следует, что полустепень захода (число входящих дуг) вершины r в графе влияния $GR(r)$ равна нулю ($d^+(r) = 0$), так как иначе в исходном графе G существовал бы ориентированный цикл $(r', r) \cup p(r, r')$ для некоторой вершины r'.

Очевидно, что полустепень захода всех остальных вершин графа влияния $GR(r)$ больше или равна единице: $\forall r' \in GR(r) : (r' \neq r) \Rightarrow (d^+(r') \geqslant 1)$. Покажем, что в минимальном графе влияния $\forall r' \in GR(r) : (r' \neq r) \Rightarrow (d^+(r') = 1)$. Пусть это не так, тогда $\exists r'' \in GR(r) : (r'' \neq r) \wedge (d^+(r'') > 1)$. Следовательно, в графе $GR(r)$ существует как минимум две различные дуги (r_1, r'') и (r_2, r''). А так как в графе $GR(r)$ любая вершина достижима из вершины r, то в этом графе существуют ориентированные пути $p(r, r_1)$ и $p(r, r_2)$, отличные друг от друга по крайней мере конечными вершинами. В соответствии с принципом наследования привилегий: $PR(r'') \subseteq PR(r_1) \subseteq PR(r)$ и $PR(r'') \subseteq PR(r_2) \subseteq PR(r)$. Но тогда к графу $GR(r)$ можно применить процедуру оптимизации графа влияния, удалив одну из дуг (r_1, r'') или (r_2, r''), что противоречит с его минимальностью.

В итоге, опираясь на определение ориентированного дерева [5, с. 238], теорема доказана. ■

Замечание 1. Если ролевой граф — дерево, то на основании предложения 1 и теоремы 2 для произвольной роли r граф влияния единственен (так как он является деревом и не подлежит оптимизации).

Теорема 3. *Трудоёмкость поиска графа влияния на произвольную вершину ролевого дерева не превосходит* $O(n^2)$.

Доказательство. Пусть ролевой граф G на n вершинах задан списками смежности L [5, с. 202]. Воспользуемся алгоритмом поиска в ширину (в глубину), представленным, например, в работе [5, с. 203]. Для построения множества $PR(r_i)$ достаточно в этом алгоритме начать обход с вершины r_i. Очевидно, результатом работы алгоритма на ориентированном графе будет множество вершин, достижимых из начальной, т. е. множество $PR(r_i)$. Трудоёмкость подобного алгоритма равна $O(n^2)$ [1].

Для задания графа влияния $GR(r_i)$ достаточно из исходных списков смежности L выбрать те, которые соответствуют вершинам множества $PR(r_i)$. Трудоёмкость этого шага зависит от реализации списков, но в любом случае не превышает $O(n^2)$. ■

Замечание 2. Вообще говоря, трудоёмкость алгоритма поиска в ширину равна $O(m)$, где m — число дуг графа. Для достаточно «плотных» графов $m = O(n^2)$. Если же граф, в котором ведётся поиск, является деревом, то $m = n - 1$. Таким образом, если ролевой граф — дерево, то трудоёмкость поиска графа влияния

на произвольную вершину может быть понижена до $O(n)$ при соответствующей реализации списков смежности.

Замечание 3. Пусть теперь ролевой граф G задан матрицей смежности M размерности $n \times n$. Очевидно, переход от такого представления к спискам смежности требует не более $O(n^2)$ операций, что оставляет справедливым утверждение теоремы 3.

Если же в нашем распоряжении имеется матрица достижимости M^* исходного ориентированного графа, то вновь трудоёмкость поиска графа влияния остаётся $O(n^2)$. Действительно, напомним, что элемент m^*_{ij} матрицы достижимости равен 1, если в орграфе существует ориентированный путь из вершины r_i в вершину r_j, и 0 — в противном случае. Тогда вершины, принадлежащие графу влияния $GR(r_i)$, — это те вершины r_j, для которых элемент $m^*_{ij} = 1$, а также сама вершина r_i. Таким образом трудоёмкость поиска множества $PR(r_i)$ равна $O(n)$. Осталось построить матрицу смежности $MR(r_i)$ графа влияния $GR(r_i)$. Для этого достаточно выбрать из матрицы смежности M ролевого графа G строки и столбцы, соответствующие множеству $PR(r_i)$. Трудоёмкость этого шага равна $O(n^2)$. Но здесь стоит заметить, что сама процедура построения матрицы достижимости имеет трудоёмкость $O(n^3)$ [5, с. 48].

2. Элементарные операторы

Для анализа преобразований введём набор элементарных операторов, преобразующих граф G.

1. $Auth(r_1, r_2)$ — авторизация роли r_1 на роль r_2, добавляет дугу от r_1 к r_2. Данная операция приводит к тому, что множество привилегий роли $r_1.p$ изменяется и становится равным $r_1.p \cup r_2.p$. Привилегии же роли r_2 остаются неизменными.

2. $DeleteA(r_1, r_2)$ — отменяет авторизацию роли r_1 на роль r_2, удаляет дугу от r_1 к r_2. Новое множество привилегий роли r_1 имеет вид

$$(r_1.p \setminus r_2.p) \cup \left(\bigcup_{r' \in (PR(r_1) \setminus r_2)} r'.p \right).$$

Простая разность множеств привилегий $r_1.p \setminus r_2.p$ может приводить к неверному результату, так как возможна ситуация, когда одна и та же привилегия наследуется от нескольких ролей.

3. $CreateR(r)$ — создаёт роль r, добавляет в граф вершину, не связанную с другими вершинами. Данный оператор сам по себе никак не влияет на распределение полномочий и не может приводить к нарушению безопасности.

4. $DeleteR(r)$ — удаляет роль r, удаляет в графе вершину. Условием выполнения оператора является изолированность вершины. Для удаления вершины со связями необходимо предварительно удалить все дуги с помощью оператора $DeleteA()$. Как легко понять, данный оператор сам по себе также не приводит к перераспределению полномочий.

5. $EnterP(p, r)$ — добавляет привилегию p в множество привилегий роли r. Соответственно данная привилегия также добавляется всем ролям, доминирующим над данной ролью.

6. $DeleteP(p, r)$ — удаляет привилегию p из множества привилегий роли r. После этого привилегия p удаляется также и у ролей, доминирующих над r, если они не наследуют эту привилегию от других ролей.

Для выполнения сложных преобразований системы из операторов могут быть составлены команды:

$$Command\ C\{$$
$$\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_n;$$
$$\}.$$

Здесь $\alpha_1, \alpha_2, \ldots, \alpha_n$ — элементарные операторы.

Теорема 4. *Для любых двух ролевых графов G и G' существует команда C, преобразующая G в G'.*

Доказательство. Для доказательства теоремы построим одну из возможных команд преобразования ролевого графа G в G'. Применим для каждой из дуг графа G команду $DeleteA()$, в результате чего получим множество изолированных вершин. Применяя нужное количество раз оператор $CreateR()$ или $DeleteR()$, добьёмся того, чтобы количество вершин стало равным количеству вершин в графе G'. Далее с помощью оператора $Auth()$ создадим дуги, соответствующие дугам графа G'. Распределив полномочия с помощью оператора $EnterP()$, получим граф, изоморфный G'. ■

Таким образом, описанный выше набор элементарных операторов является полным и на его основе можно построить любой ролевой граф.

3. Классы безопасных систем

Следует отметить, что команды в системе выполняются атомарно, т. е. подсистема безопасности может производить проверку только после завершения команды. Выявим несколько классов безопасных систем. Для этого рассмотрим ограничения, которые при накладывании на систему могут гарантировать безопасность.

1. *Статический ролевой граф.* К этому классу относятся системы, у которых ролевой граф остаётся неизменным, т. е. отсутствует администрирование системы в процессе её функционирования. Как было показано в теореме 1, такие системы являются безопасными. Для проверки безопасности системы со статическим ролевым деревом могут применяться теоремы поиска ориентированных путей из теории графов.

2. *Фиксированное количество административных ролей.* Новые роли может создавать конечное фиксированное множество ролей (администраторов). Расширение предыдущего класса.

3. *Монооперационные системы.* Каждая команда содержит только один элементарный оператор. Можем после каждой команды проводить проверку системы, причём выполняется она за конечное число шагов.

4. *Системы с выделенным оператором создания административных ролей.* Если команда включает создание административной роли, то не включает других ролей.

Заключение

Таким образом, для ролевой политики безопасности имеется возможность формализации путём определения элементарных операторов преобразования ролевого графа. Их использование, в свою очередь, позволит получать более строгие доказательства безопасности систем с ролевым разграничением доступа к информации в соответствии с определёнными формализованными критериями.

Литература

1. Алексеев В. Е., Таланов В. А. Графы и алгоритмы. URL: http://www.intuit.ru/department/algorithms/gaa/4/ (дата обращения: 16.10.2010).
2. Белим С. В., Белим С. Ю., Богаченко Н. Ф. Теоретико-графовый анализ ролевой политики безопасности // Математические структуры и моделирование. Омск: УниПак. 2009. Вып. 19. С. 85–96.
3. Белим С. В., Богаченко Н. Ф., Ракицкий Ю. С. Совместная реализация мандатного и ролевого разграничения доступа к информации в компьютерных системах // Математические структуры и моделирование. Омск: УниПак. 2009. Вып. 20. С. 141–152.
4. Гайдамакин Н. А. Разграничение доступа к информации в компьютерных системах. Екатеринбург: Изд-во Урал. ун-та, 2003. 328 с.
5. Новиков Ф. А. Дискретная математика для программистов: учебник для вузов. СПб.: Питер, 2001. 304 с.
6. Хаггарти Р. Дискретная математика для программистов: пер. с англ. М.: Техносфера, 2005. 400 с.

Математические структуры и моделирование
2010, вып. 22, с. 121–130

УДК 004.056.52

ИССЛЕДОВАНИЕ БЕЗОПАСНОСТИ ДИСКРЕЦИОННОГО РАЗДЕЛЕНИЯ ДОСТУПА В ОС WINDOWS

С. В. Белим, Д. М. Бречка

На основе модели HRU исследуется дискреционное разделение доступа в операционных системах семейства Windows.

Введение

На сегодняшний день разработано несколько математических моделей подсистем безопасности компьютерных систем, допускающих гарантированную защищённость от утечек информации. Однако при попытке применения полученных результатов к реальным системам приходится сталкиваться с рядом трудностей. В данной статье предпринимается попытка описания механизма дискреционного разделения доступа подсистемы безопасности ОС Windows в рамках модели HRU [2–5, 8].

Рассмотрим основные моменты разграничения доступа в ОС Windows [7, 9, 10]. Модель защиты Windows требует, чтобы поток заранее указывал операции, которые он собирается выполнить с объектом. Система проверяет права доступа, запрошенные потоком, и, в случае наличия разрешений, выдаёт потоку дескриптор, позволяющий осуществлять операции с объектом.

Права доступа к конкретному объекту хранятся в структурах, называемых списками контроля доступа ACL (Access Control List). Ссылки на ACL объекта хранятся в дескрипторе безопасности объекта DS. Для каждого объекта существуют системный ACL (SACL, System ACL) и избирательный ACL (DACL, Discretionary ACL). DACL администрируется владельцем объекта и предназначен для разделения доступа к объекту. SACL объекта администрируется системным администратором и предназначен для аудита действий с объектом. Списки контроля доступов состоят из записей ACE (Access Control Entry), каждая из которых отвечает за разрешение или запрет одного права доступа.

В операционных системах семейства Windows существует три вида прав доступа к объектам:

1) стандартные права доступа — это права доступа, применимые к любому объекту (изменение владельца объекта, изменение списка DACL объекта, удаление объекта и т. д.);

Омский государственный университет им. Ф. М. Достоевского.
E-mail: sbelim@omsu.ru

2) специальные права доступа — это права доступа, применимые только к объектам данного вида (чтение данных из объекта, запись данных в объект, чтение атрибутов объекта, выполнение программного файла и т. д.);

3) общие права доступа — комбинации специальных и стандартных прав доступа. Общие права доступа вводятся для того, чтобы при установке разрешений на доступ абстрагировать владельца от конкретной реализации объекта.

Процесс преобразования общего права доступа к объекту в набор специальных и стандартных прав называется отображением права доступа.

Определены следующие общие права доступа:

1) чтение, включающее в себя чтение DACL объекта, чтение данных из объекта, чтение его атрибутов и расширенных атрибутов, использование объекта для синхронизации;

2) запись, включающая в себя чтение DACL объекта, запись и добавление данных в объект, запись его атрибутов и расширенных атрибутов, использование объекта для синхронизации;

3) выполнение, включающее в себя чтение DACL объекта, чтение его атрибутов, выполнение программного файла и использование объекта для синхронизации;

4) все действия с объектом.

Модель HRU названа по первым буквам фамилий ученых, предложивших её (Харисон, Руззо, Ульман). Основное назначение этой модели состоит в анализе политики безопасности, построенной на основе матрицы доступов. Всю совокупность DACL объектов компьютерной системы можно представить в виде одной таблицы, называемой матрицей доступов M. В модели HRU определены шесть примитивных операторов, преобразующих матрицу доступов:

1. $Enter(r, M[S, O])$ — внести право $r \in R$ в ячейку $M[S, O]$, т. е. объект S получает право доступа r к объекту O.

2. $Delete(r, M[S, O])$ — удалить право r из ячейки $M[S, O]$, т. е. запретить субъекту S доступ r к объекту O.

3. $CreateS(S)$ — создать объект S. При этой операции в матрице доступов появляются дополнительный столбец и дополнительная строка. Вновь создаваемый субъект не имеет никаких прав.

4. $CreateO(O)$ — добавление нового объекта O. В матрице доступов появляется новый столбец, на который ни у кого нет никаких прав доступа.

5. $DeleteS(S)$ — уничтожить субъект S. В таблице удаляется одна строка и один столбец.

6. $DeleteO(O)$ — уничтожить объект O. В таблице удаляется один столбец.

Из примитивных операторов могут составляться команды. Каждая команда состоит из двух частей:

1. Условия, при которых возможно выполнение команды.

2. Последовательность примитивных операторов.

$$\begin{array}{ll} command & C(X_1, ..., X_k) \\ & if\ r_1 \in M[X_1]\ and ... and\ r_k \in M[X_k]\ then \\ & \alpha_1;\ \alpha_2;\ ...\ \alpha_n; \\ end. & \end{array}$$

Здесь через $X_i (i = \overline{1,k})$ обозначены пары $X_i = (S_{im}, O_{ij})$. Условия в теле команды являются необязательными, если известно, что система гарантированно находится в нужном состоянии.

1. Описание прав доступа ОС Windows в рамках модели HRU

Для корректного описания системы в рамках модели HRU необходимо корректно определить множество возможных прав доступа. Как видно из спецификации подсистемы безопасности ОС Windows, элементарные операции задаются стандартными и специальными правами доступа, тогда как общие права доступа являются комбинацией стандартных и специальных прав, т. е. командами.

Для записи конкретной команды, соответствующей некоторому общему праву доступа, будем рассматривать объект операционной системы как совокупность объектов. Такое рассмотрение допустимо в рамках субъектно-объектного подхода, который требует, чтобы конкатенация двух объектов также была объектом.

Для обозначения какой-то части объекта будем указывать общий идентификатор объекта и идентификатор соответствующей части, разделённые точкой. Например, содержимое файла F будет иметь обозначение $F.Data$. При работе с файлом важными структурами являются его заголовок $F.Header$, дескриптор безопасности $F.DS$ и списки контроля доступа $F.DACL$, $F.SACL$.

Выпишем команды, соответствующие общим правам доступа к файлу F.

1. Чтение r файла F неким процессом S:

$$\begin{array}{ll} command & Read\_File((S, F)) \\ & Enter(r, M[S, F.Header]); \\ & Enter(r, M[S, F.Data]); \\ & Enter(r, M[S, F.DS]); \\ & Enter(r, M[S, F.DACL]); \\ & Enter(r, M[S, F.SACL]); \\ end. & \end{array}$$

2. Запись w в файл F неким процессом S:

$$
\begin{aligned}
command \quad & Write\_File((S,F)) \\
& Enter(w, M[S, F.Header]); \\
& Enter(w, M[S, F.Data]); \\
& Enter(w, M[S, F.DS]); \\
& Enter(r, M[S, F.DACL]); \\
& Enter(w, M[S, F.SACL]); \\
end.
\end{aligned}
$$

3. Запуск исполненяемого файла F неким процессом S:

$$
\begin{aligned}
command \quad & Execute\_File((S,F)) \\
& Enter(r, M[S, F.Header]); \\
& Enter(x, M[S, F.Data]); \\
& Enter(r, M[S, F.DS]); \\
& Enter(r, M[S, F.DACL]); \\
& Enter(r, M[S, F.SACL]); \\
end.
\end{aligned}
$$

4. Полный доступ к файлу F неким процессом S:

$$
\begin{aligned}
command \quad & All\_File((S,F)) \\
& Enter(r, M[S, F.Header]); \\
& Enter(w, M[S, F.Header]); \\
& Enter(r, M[S, F.Data]); \\
& Enter(w, M[S, F.Data]); \\
& Enter(x, M[S, F.Data]); \\
& Enter(r, M[S, F.DS]); \\
& Enter(w, M[S, F.DS]); \\
& Enter(r, M[S, F.DACL]); \\
& Enter(r, M[S, F.SACL]); \\
& Enter(w, M[S, F.SACL]); \\
end.
\end{aligned}
$$

При рассмотрении безопасности системы важным является вопрос организации журнала аудита системы. Если аудит доступа к объектам системы фиксирует каждый доступ со стандартными или специальными правами, то данная система монооперационная и является безопасной [2, 4]. Однако, как легко понять в ОС Windows, в журнале аудита могут фиксироваться и общие права доступа, т. е. система перестаёт быть монооперационной и вопрос о её безопасности остаётся открытым, так как для произвольной системы задача проверки безопасности алгоритмически не разрешима [2, 4].

2. Базисный подход

Для исследования безопасности механизма дискреционного разделения доступа в ОС Windows применим базисный подход к модели HRU, развитый в [1,6]. Для этого рассмотрим представление прав доступа на битовом уровне с помощью маски доступов [7,9,10]. Запрос субъекта на конкретный вид доступа к объекту преобразуется в маску доступа, которая сравнивается с масками разрешённых и запрещённых доступов в элементах частного списка контроля доступов (DACL) объекта.

Маска доступа, содержащаяся в элементе DACL, представляет собой значение длиной 32 бита. Первые 16 битов определяют специальные права доступа, биты с 16 до 23 — стандартные права доступа, бит 24 — право ACCESS_ SYSTEM_ SECURITY, бит 25 — право MAXIMUM_ ALLOWED (полный доступ), биты 26 и 27 зарезервированы для дальнейшего использования, биты с 28 по 31 определяют общие права доступа, отображаемые в специальные и стандартные права при попытке доступа к объекту.

Как было показано в [1,6], в модели HRU можно перейти к новому базису, работающему с битовой строкой. Будем использовать следующие обозначения: M — матрица доступов (совокупность DACL всех объектов системы), $\mathbf{S}$ — множество субъектов компьютерной системы, $\mathbf{O}$ — множество объектов компьютерной системы, $M[S,O]$ — элемент матрицы доступов, определяющий права доступа субъекта $S \in \mathbf{S}$ к объекту $O \in \mathbf{O}$.

1. $AddO(O,M)$ — добавить столбец, соответствующий объекту O в матрицу доступов M.

2. $DelO(O,M)$ — удалить столбец, соответствующий объекту O в матрице доступов M.

3. $AddS(S,M)$ — добавить строку и столбец, соответствующие субъекту S в матрицу доступов M.

4. $DelS(S,M)$ — удалить строку и столбец, соответствующие субъекту S в матрице доступов M.

5. $Inv(M[S,O],k)$ — инвертировать в элементе матрицы доступов $M[S,O]$ бит с номером k. По сути эта операция выдает или отменяет право доступа a_k.

Выпишем команды для всех стандартных, специальных и общих прав доступа в данном базисе.

Специальные права доступа

Специальные права и их значения приведены в табл. 1.

Команды в базисе для специальных прав приведены ниже.

1. Чтение данных файла F:

$$
\begin{aligned}
&command \quad && File\_Read\_Special(F)\\
&&& if(b_0 == FALSE);\\
&&& Inv(F,0);\\
&end.
\end{aligned}
$$

Таблица 1

Специальные права и их значения

Название права	*Значение*	*Описание права*
FILE_READ_DATA	0x00000001	Чтение данных
FILE_WRITE_DATA	0x00000002	Запись данных
FILE_APPEND_DATA	0x00000004	Добавление данных
FILE_READ_EA	0x00000008	Чтение расширенных атрибутов
FILE_WRITE_EA	0x00000010	Запись расширенных атрибутов
FILE_EXECUTE	0x00000020	Исполнение
FILE_DELETE	0x00000040	Удаление
FILE_READ_ATTRIBUTES	0x00000080	Чтение атрибутов
FILE_WRITE_ATTRIBUTES	0x00000100	Запись атрибутов

2. Запись данных в файл F:

$$\begin{aligned} command \quad & File\_Write\_Special(F) \\ & if(b_1 == FALSE); \\ & Inv(F,1); \\ end. & \end{aligned}$$

3. Добавление данных в файл F:

$$\begin{aligned} command \quad & File\_Uppend\_Special(F) \\ & if(b_2 == FALSE); \\ & Inv(F,2); \\ end. & \end{aligned}$$

4. Чтение расширенных атрибутов файла F:

$$\begin{aligned} command \quad & File\_Read\_EA\_Special(F) \\ & if(b_3 == FALSE); \\ & Inv(F,3); \\ end. & \end{aligned}$$

5. Запись расширенных атрибутов файла F:

$$\begin{aligned} command \quad & File\_Write\_EA\_Special(F) \\ & if(b_4 == FALSE); \\ & Inv(F,4); \\ end. & \end{aligned}$$

6. Запуск исполненяемого файла F:

$$\begin{aligned} command \quad & File\_Execute\_Special(F) \\ & if(b_5 == FALSE); \\ & Inv(F,5); \\ end. & \end{aligned}$$

7. Удаление файла F:

$$\begin{aligned} command \quad & File\_Delete\_Special(F) \\ & if(b_6 == FALSE); \\ & Inv(F, 6); \\ end. & \end{aligned}$$

8. Чтение атрибутов файла F:

$$\begin{aligned} command \quad & File\_Read\_Attributes\_Special(F) \\ & if(b_7 == FALSE); \\ & Inv(F, 7); \\ end. & \end{aligned}$$

9. Запись атрибутов файла F:

$$\begin{aligned} command \quad & File\_Write\_Attributes\_Special(F) \\ & if(b_8 == FALSE); \\ & Inv(F, 8); \\ end. & \end{aligned}$$

Стандартные права доступа
Стандартные права доступа приведены в табл. 2.

Таблица 2

Стандартные права доступа

Название права	*Значение*	*Описание права*
OBJECT_DELETE	0x00010000	Удаление объекта
OBJECT_READ_CONTROL	0x00020000	Чтение DS объекта
OBJECT_WRITE_DACL	0x00040000	Чтение DACL объекта
OBJECT_CHANGE_OWNERSHIP	0x00080000	Изменение права собственности объекта
OBJECT_SYNCHRONIZE	0x00100000	Использование объекта для синхронизации

Ниже приведены стандартные права, выраженные в базисе.
1. Удаление объекта F:

$$\begin{aligned} command \quad & Object\_Delete\_Standard(F) \\ & if(b_{16} == FALSE); \\ & Inv(F, 16); \\ end. & \end{aligned}$$

2. Чтение DS объекта F:

$$\begin{aligned} command \quad & Object\_Read\_Control\_Standard(F) \\ & if(b_{17} == FALSE); \\ & Inv(F, 17); \\ end. & \end{aligned}$$

3. Запись DACL объекта F:

$$\begin{aligned} command \quad & Object\_Write\_DACL\_Standard(F) \\ & if(b_{18} == FALSE); \\ & Inv(F, 18); \\ end. & \end{aligned}$$

4. Изменение прав владения объекта F:

$$\begin{aligned} command \quad & Object\_Change\_Ownership\_Standard(F) \\ & if(b_{19} == FALSE); \\ & Inv(F, 19); \\ end. & \end{aligned}$$

5. Изменение свойства синхронизации объекта F:

$$\begin{aligned} command \quad & Object\_Synchronize\_Standard(F) \\ & if(b_{20} == FALSE); \\ & Inv(F, 20); \\ end. & \end{aligned}$$

Общие права доступа

В табл. 3 приведены общие права доступа, ниже эти права представлены в базисе.

Таблица 3

Общие права доступа

Название права	*Значение*	*Описание права*
GENERIC_ALL	0x10000000	Общее право на полный доступ
GENERIC_EXECUTE	0x20000000	Общее право исполнения
GENERIC_WRITE	0x40000000	Общее право записи
GENERIC_READ	0x80000000	Общее право чтения

1. Добавление права на полный доступ на объект F:

$$\begin{aligned} command \quad & Generic\_All(F) \\ & if(b_{28} == FALSE); \\ & Inv(F, 28); \\ end. & \end{aligned}$$

2. Добавление права общего исполнения на объект F:

$$\begin{aligned} command \quad & Generic\_Execute(F) \\ & if(b_{29} == FALSE); \\ & Inv(F, 29); \\ end. & \end{aligned}$$

3. Добавление права общей записи на объект F:

$$\begin{aligned} command \quad & Generic\_Write(F) \\ & if(b_{30} == FALSE); \\ & Inv(F, 30); \\ end. & \end{aligned}$$

4. Добавление права общего чтения на объект F:

$$\begin{aligned} command \quad & Generic\_Read(F) \\ & if(b_{31} == FALSE); \\ & Inv(F, 31); \\ end. & \end{aligned}$$

Таким образом, набор прав доступа, описанный в спецификации ОС Windows, можно считать базисом, а сама ОС Windows, как видно из вышеприведённого, будет монооперационной системой в данном базисе.

3. Заключение

В качестве общего заключения можно сказать, что переход к новому базису в рамках модели HRU позволяет рассматривать широкий спектр реальных компьютерных систем. В частности, возможно рассмотрение дискреционного разделения доступа в подсистеме безопасности ОС Windows на основе модели HRU, а также рассмотрение ОС Windows как монооперационной в некотором базисе.

Литература

1. Бречка Д. М., Белим С. В. Исследование безопасности компьютерных систем в модели дискреционного разделения доступа HRU // Математические структуры и моделирование. 2009. Вып. 19. С. 97–103.
2. Гайдамакин Н. А. Разграничение доступа к информации в компьютерных системах. Екатеринбург: Изд-во Урал. ун-та, 2003.
3. Грушо А. А., Тимонина Е. Е. Теоретические основы защиты информации. М.: Яхтсмен, 1996.

4. Девянин П. Н. Модели безопасности компьютерных систем. М.: Академия, 2005.
5. Зегжда Д. П., Ивашко А. М. Основы безопасности информационных систем. М.: Горячая линия – Телеком, 2000.
6. Проблемы обработки и защиты информации: коллективная монография / под общ. ред. д-ра физ.-мат. наук С. В. Белима. Омск: КАН, 2010. Кн. 1: Модели политик безопасности компьютерных систем.
7. Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер-класс: пер. с англ. 4-е изд. М.: Русская редакция, 2005.
8. Harrison M. A., Ruzzo W. L., Ullman J. D. Protection in Operating Systems // Communications of the ACM. 1975. P. 14–25.
9. Microsoft Corporation. Microsoft Windows XP Professional. Учебный курс MCSA/ MCSE: пер. с англ. 2-е изд., испр. М.: Русская редакция, 2003.
10. Microsoft Developer Network. URL: http://msdn.microsoft.com (дата обращения: 10.10.2010).

Математические структуры и моделирование
2010, вып. 22, с. 131–138

УДК 004.056.55

ПРАКТИЧЕСКИЕ АСПЕКТЫ ГЕНЕРАЦИИ КЛЮЧЕЙ ДЛЯ КРИПТОСИСТЕМЫ ЭЛЬ-ГАМАЛЯ

Е. А. Илюшечкин, А. А. Лаптев

Рассматриваются оптимальные методы выработки ключей шифрования для криптосистемы Эль-Гамаля. Приводится результат практической реализации этих методов.

Введение

В наше время большое практическое значение имеет асимметричная криптография (или криптография с открытым ключом). Одной из популярных криптосистем с открытым ключом является криптосистема Эль-Гамаля, реализованная в таких криптографических пакетах, как PGP и GnuPG.

Весьма важной частью реализации любой криптосистемы является генерация ключей. С одной стороны, неправильно подобранный ключ может серьёзно снизить безопасность системы и, следовательно, практическую применимость данной её реализации. С другой стороны, желательно минимизировать время, затрачиваемое как на генерацию самих ключей, так и на операции, производимые с помощью этих ключей в дальнейшем. К сожалению, в литературе зачастую трудно найти единый комплекс требований, обеспечивающий оптимальную стратегию выбора ключей. Цель данной работы — составить подобный комплекс практических рекомендаций, исследовав различные методы теоретически и сравнив их экспериментально.

1. Теоретический подход к выбору ключей

1.1. Модуль p

Наиболее ресурсоёмким этапом генерации ключей исследуемой криптосистемы (её описание, а также обозначения, принятые в данной работе, можно найти в [1]) является нахождение большого простого числа p. Выбор p ограничен условием: число $q = \frac{p-1}{2}$ должно иметь хотя бы один большой простой делитель. На практике обычно используют более сильное условие: q не должно иметь малых

Омский государственный университет им. Ф. М. Достоевского.
E-mail: ilush2005@yandex.ru

простых делителей вовсе [2, 4]. Было выделено два класса простых чисел, удовлетворяющих второму варианту условия. Первый класс известен в литературе как безопасные простые числа. Числа второго класса для удобства нами были названы «оптимальными», так как на текущий момент выглядят сравнимыми с безопасными по защищённости от известных атак, но время их поиска гораздо меньше.

Простое число p называется безопасным (англ. *safe prime*), если число $\frac{p-1}{2}$ также простое. Безопасные числа достаточно редки. В связи с этим были исследованы методы ускорения поиска таких чисел, в том числе идея об эффективном просеивании кандидатов из [8]. На основе этих методов нами был составлен алгоритм генерации простого числа q, для которого $p = 2q + 1$ заведомо будет простым, т. е. безопасным. Алгоритм описан в табл. 1; через P_B обозначено множество всех простых чисел, меньших B.

Таблица 1

Алгоритм генерации q

1. Выбрать случайное число $q \equiv 5 \ (mod\ 6)$ требуемого размера.
2. Если $\exists\ r \in P_B : q \in \{0, \frac{r-1}{2}\} \ (mod\ r)$, перейти к шагу 5.
3. Если $2^{p-1} \not\equiv 1 \ (mod\ p)$, перейти к шагу 5.
4. Если q проходит тест Рабина–Миллера, вернуть q, алгоритм завершён.
5. Увеличить q на 6 и перейти к шагу 2.

Алгоритм завершает работу за конечное число шагов, поскольку, начиная с некоторого числа, последовательно в порядке возрастания перебирает всех возможных кандидатов и гарантированно распознает первое же число q, для которого $2q+1$ является безопасным простым. Справедливость этого алгоритма базируется на следующих утверждениях.

Теорема 1. *Пусть* $p = 2q + 1$, $r \neq 2$ — *простое. Тогда* $p \equiv 0 \ (mod\ r) \Leftrightarrow q \equiv \frac{r-1}{2} \ (mod\ r)$.

Доказательство. Двойка обратима по модулю r, поэтому:

Достаточность. $p = 2q + 1 \equiv 2\frac{r-1}{2} + 1 \equiv r \equiv 0 \ (mod\ r)$.

Необходимость. $q = 2^{-1}(p - 1) \equiv \frac{r+1}{2}(r - 1) \equiv \frac{r-1}{2} \ (mod\ r)$. ■

Теорема 2. *Пусть* $p = 2q+1$, q — *простое. Тогда* p — *простое* $\Leftrightarrow p \not\equiv 0 \ (mod\ 3)$, $2^{p-1} \equiv 1 \ (mod\ p)$.

Доказательство. По теореме Поклингтона: если существует простое $q \mid (p-1)$:

1. $q > \sqrt{p} - 1$

2. $\exists a : (a^{p-1} \equiv 1 \ (mod\ p)) \wedge ((a^{\frac{p-1}{q}} - 1,\ p) = 1)$

$\Rightarrow p$ — простое.

Необходимость. Когда $p = 2q + 1$, первое условие теоремы выполнено для любого положительного q, а $2^{\frac{p-1}{q}} - 1 = 3$. Если $2^{p-1} \equiv 1 \ (mod\ p)$ и $p \not\equiv 0 \ (mod\ 3)$, то второе условие также выполнено для $a = 2 \Rightarrow p$ — простое.

Достаточность. Если же p — простое, то $2^{p-1} \equiv 1 \ (mod\ p)$ в силу малой теоремы Ферма и очевидно, что $p \not\equiv 0 \ (mod\ 3)$. ■

Так как простота q опирается в итоге на результат теста Рабина–Миллера (стандартный подход для современных методов генерации простых чисел), алгоритм является вероятностным и с крайне низкой вероятностью ошибается. Если бы простота p проверялась также с помощью теста Рабина–Миллера, то вероятность ошибки была бы выше. Использование для числа p теста Поклингтона (который является детерминированным и не ошибается) на третьем шаге алгоритма позволяет не только ускорить поиск чисел, но и снизить вероятность ложного результата.

Кроме оптимизации алгоритма, поиск безопасных простых чисел можно улучшить с помощью следующих приёмов.

1. Реализация теста Рабина–Миллера должна учитывать размер числа. Чтобы подобрать минимальное число итераций для заданного размера ключа, можно использовать табл. 4.4 из [5].

2. Желательно подобрать оптимальную границу B. Можно для этого использовать замечание 4.45 об оптимальной границе для пробных делений из [5].

3. Размер таблицы малых простых чисел можно заметно сократить, храня разность между соседними числами, вместо самих чисел.

4. При нахождении остатков от деления на малые простые числа можно использовать остатки, вычисленные для предыдущих кандидатов.

5. Поскольку значение i-го кандидата легко вычислить, последовательность можно разбить на блоки и производить проверку в каждом блоке на отдельном процессоре.

Простое число p называется «оптимальным» (в рамках данной статьи), если $p = 1 + 2 \cdot q_1 \cdot ... \cdot q_n$, где все q_i $(i = 1, ..., n)$ — простые числа, имеющие битовый размер не менее q_{size}. Для оптимальных чисел q_{size} может быть найден по табл. 2. Чтобы ускорить поиск таких чисел, можно использовать идею перебора сочетаний из пула простых чисел, описанную в [4]. Процесс генерации выглядит следующим образом. Сначала вычисляется n — число чисел q_i. Затем генерируется $m > n$ (в нашей программе использовалось $m = 3n$) простых чисел. Теперь последовательно выбираются всевозможные сочетания по n чисел из m. Из каждой такой выборки составляется соответствующее число p, для которого проверяется соответствие заданному размеру и, если размер правильный, p проверяется на простоту.

Если p прошло проверки, то искомое число найдено, иначе делается следующая выборка, либо генерируется новый пул, если все возможные сочетания были исчерпаны. Чтобы ускорить этот процесс, можно не генерировать весь пул сразу: каждый элемент пула нужно генерировать только тогда, когда он впервые понадобится для некоторой выборки.

Таблица 2

Таблица Винера

p_{size}	512	768	1024	1280	1536	1792	2048	2304	2560	2816	3072
q_{size}	119	145	165	183	198	212	225	237	249	259	269

1.2. Порождающий элемент g

Алгоритмы нахождения порождающего элемента Z_p^* (мультипликативной группы поля вычетов) и её подгруппы можно найти в [5] (номера алгоритмов — 4.80 и 4.81). Для порождающего элемента полной группы необходимо знать разложение числа $p-1$ на простые множители: $p-1 = q_1^{a_1} \cdot q_2^{a_2} \cdot ... \cdot q_n^{a_n}$. Это не является помехой для оптимальных или безопасных чисел, поскольку описанные выше методы их поиска гарантируют, что $p-1$ будет всегда построено из известных простых сомножителей. Для безопасных чисел, кроме того, шаг 2 алгоритма можно заменить на однократную проверку символа Лежандра $(\frac{q}{p})$, который вычисляется на порядок быстрее, чем возведение в степень.

Двойка с точки зрения производительности является лучшим значением для порождающего элемента, поскольку при эффективной реализации возводится в степень по модулю быстрее, чем другие числа [6]. В предложенном алгоритме генерации q для безопасных чисел можно легко исключить из рассмотрения кандидатов, для которых двойка не является порождающим элементом требуемой подгруппы. Для этого нужно увеличивать q не на 6, а на 12 и выбрать на первом шаге случайное число $q \equiv 5 \ (mod\ 12)$ или $q \equiv 11 \ (mod\ 12)$, чтобы 2 имела порядок $2q$ или q соответственно. Такая модификация объясняется следующим утверждением.

Теорема 3. *Пусть $p = 2q + 1 > 5$ — безопасное простое. Тогда 2 имеет в Z_p^* порядок $2q$, если $q \equiv 5 \ (mod\ 12)$, и порядок q, если $q \equiv 11 \ (mod\ 12)$.*

Доказательство. $(\frac{2}{p}) = (-1)^{\frac{p^2-1}{8}} = ((-1)^q)^{\frac{q+1}{2}} = (-1)^{\frac{q+1}{2}}$;

$q \equiv 5 \ (mod\ 12) \Rightarrow \frac{q+1}{2} \equiv 1 \ (mod\ 2) \Rightarrow (\frac{2}{p}) = -1$;

$q \equiv 11 \ (mod\ 12) \Rightarrow \frac{q+1}{2} \equiv 0 \ (mod\ 2) \Rightarrow (\frac{2}{p}) = 1$.

$(\frac{2}{p}) = 2^{\frac{p-1}{2}} = 2^q \ (mod\ p)$. Порядок должен делить $p-1$, поэтому для всех вычетов $2, ..., p-2$ он равен q или $2q$. Соответственно,

$q \equiv 5 \ (mod\ 12) \Rightarrow 2^q \equiv -1 \not\equiv 1 \ (mod\ p) \Rightarrow$ порядок равен $2q$;

$q \equiv 11 \ (mod\ 12) \Rightarrow 2^q \equiv 1 \ (mod\ p) \Rightarrow$ порядок равен q. ■

1.3. Закрытый ключ

Скорость операций в криптосистеме пропорциональна размеру секретной части ключа. Сделав размеры x и k минимально допустимыми для требуемого уровня безопасности, можно получить существенный выигрыш производительности. Но заметим, что секретная часть ключа должна быть достаточно большой, чтобы избежать возможности использования лямбда-метода Полларда. Оптимальный размер секретной части ключа — это значение $q_{size} \cdot C$, где q_{size} определяется из табл. 2 (размер p должен быть задан), а $C \geq 1$ — константа, задающая запас прочности.

1.4. Выбор группы

Другой важный этап генерации ключей — выбор рабочей группы. В предположении о неразрешимости проблемы DDH система является семантически бе-

зопасной (по известному шифртексту и открытому ключу невозможно получить значимую информацию об открытом тексте) только при работе в собственной подгруппе Z_p^* простого порядка.

Для безопасных чисел предлагается следующая семантически безопасная схема [7]. Пусть сообщение $M \in \{1, ..., q\}$. Шифруется $M_2 = M^2 \ (mod\ p)$. Восстановление M по M_2: Если $M' = (M_2)^{\frac{q+1}{2}} \ (mod\ p) > q$, то $M = -M' \ (mod\ p)$, иначе $M = M'$.

Отметим, что для расшифрования в данной схеме требуется дополнительное возведение в степень. В результате скорость расшифрования падает примерно вдвое. Кроме того, при уменьшении размера секретного ключа пропорционально повысится лишь скорость шифрования, тогда как скорость расшифрования почти не изменится. Другим очевидным недостатком является уменьшение в два раза пространства разрешённых сообщений, однако этот недостаток является менее существенным, поскольку на практике шифруемые сообщения, как правило, имеют намного меньшую длину, чем q (основная масса сообщений — симметричные ключи длиной 64–256 бит).

Для «оптимальных» чисел представить сообщение как элемент собственной подгруппы Z_p^* простого порядка q значительно сложнее. Манипуляции только лишь с сообщением на порядок увеличивают время шифрования, поэтому необходимо прибегать к модификациям самого процесса шифрования.

Одним из известных подходов, решающих проблему кодирования, является хэшированный вариант криптосистемы, описанный в [3]. Там же авторы предлагают более сложный и оригинальный метод. Необходимо отметить, что оба этих варианта нуждаются в некоторых дополнительных нестандартных предположениях. Такие предположения, в отличие от предположений Диффи–Хеллмана, не являются хорошо изученными и потому выглядят менее убедительными.

2. Практические результаты

Для проверки работоспособности изложенных методов на практике нами была написана программа для генерации ключей и шифрования. Эта программа использовалась в серии тестов, результаты которых можно видеть на представленных ниже графиках.

На рис. 1 сравнивается скорость поиска безопасных простых чисел в криптографическом пакете OpenSSL (функция BN_generate_prime_ex) и в нашей программе.

На рис. 2 сравнивается скорость поиска безопасных и оптимальных чисел. Как видно, оптимальные числа, в отличие от безопасных, могут генерироваться на лету для ключей приемлемого размера.

На рис. 3 сравнивается скорость шифрования и расшифрования для различных вариантов криптосистемы:

1. Классический вариант, работающий в полной группе Z_p^*;
2. Семантически безопасная схема для безопасных чисел;

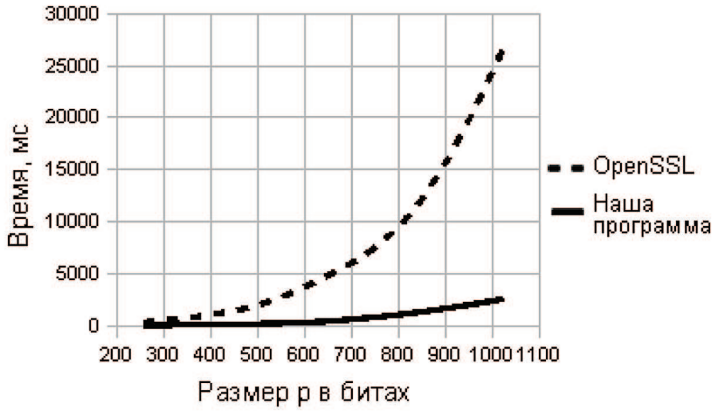

Рис. 1. Сравнение скорости генерации безопасных простых чисел в разных реализациях

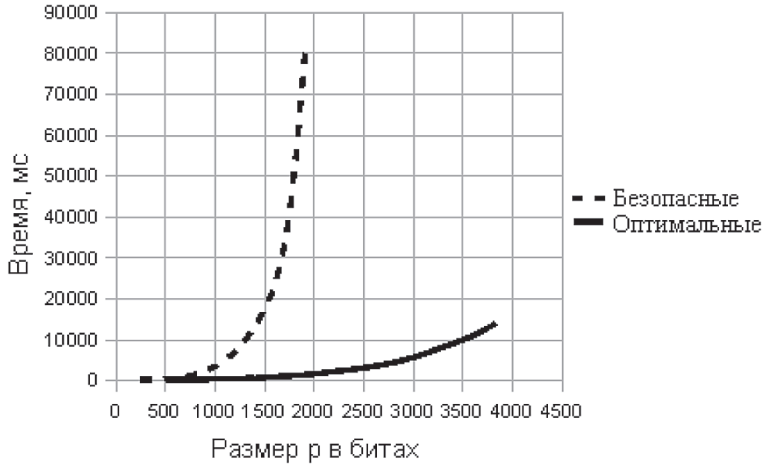

Рис. 2. Сравнение скорости генерации разных типов простых чисел

3. Хэшированный вариант криптосистемы для оптимальных чисел, использующий в качестве хэш-функции SHA.

Представленные на рис. 3 графики показывают зависимость времени шифрования и расшифрования одного сообщения от размера числа p. Поскольку асимметричные криптосистемы, как правило, применяются для шифрования довольно небольших симметричных сеансовых ключей, на практике время шифрования одного сообщения важнее времени шифрования фиксированного объёма информации. Размер подгруппы для оптимальных чисел выбирался по табл. 2. Секретная часть ключа везде выбиралась из всех возможных значений.

Как видно из графиков, скорости шифрования первых двух вариантов практически равны, а скорости расшифрования отличаются примерно вдвое, что сходится с ожидаемым результатом. Скорость шифрования и расшифрования для третьего варианта меньше примерно во столько же раз, что и размер используемой рабочей группы. Таким образом, вычисление хэш-функции по вре-

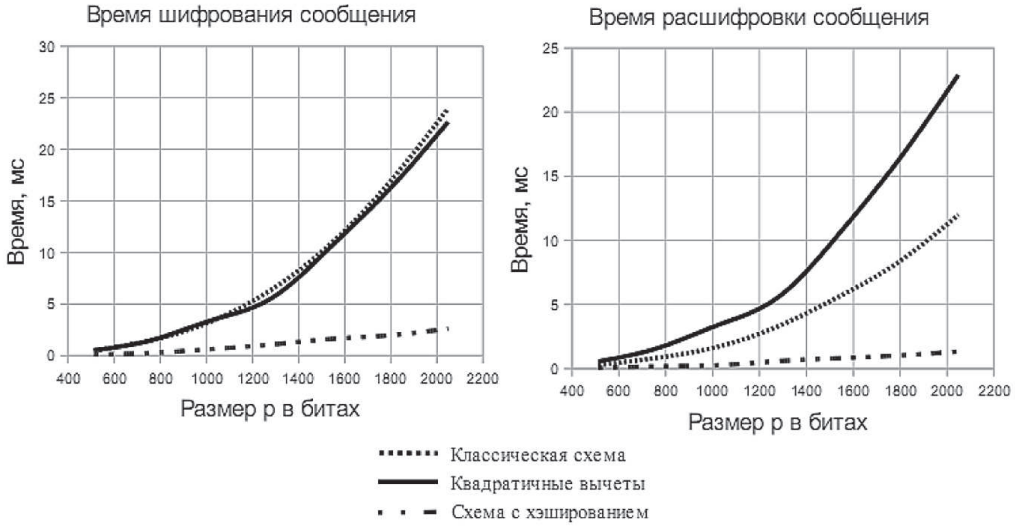

Рис. 3. Сравнение скорости генерации безопасных простых чисел в разных реализациях

мени сравнимо с умножением и заметно не влияет на скорость. Стоит отметить, что заметный отрыв графика для хэшированной схемы связан с тем, что возможный диапазон размера секретного ключа изначально гораздо меньше. Как было сказано ранее, для двух других схем секретный ключ также может выбираться из меньшего диапазона. В этом случае хэшированная схема не будет иметь преимущества по скорости.

Таким образом, при выборе между классическим вариантом криптосистемы и вариантом, основанным на подгруппах, следует решить, насколько системе нужна семантическая безопасность. Скорее всего, она будет нужна, если шифруемые сообщения очень специфичны. Если семантическая безопасность не важна, то лучше использовать классическую схему. Для достижения скорости, сравнимой при работе в подгруппах, нужно будет, разумеется, выбирать секретный ключ из минимального диапазона. Если же семантическая безопасность важна, то необходимо работать в подгруппах. В этом случае использовать описанную выше семантически безопасную схему для безопасных чисел следует в том случае, если приоритетом является долгосрочная безопасность, но не скорость расшифрования, которая будет заметно ниже. При отсутствии требования долгосрочной безопасности стоит работать с «оптимальными» числами, но в этом случае нужно хорошо продумать способ кодирования сообщений.

3. Заключение

В данной работе нами были проанализированы подходы к генерации ключей для базовой криптосистемы Эль-Гамаля и её варианта, работающего в подгруппе простого порядка группы Z_p^*. Основываясь на полученных результатах, мы предлагаем следующие правила для выработки оптимальных ключей с точки зрения безопасности и производительности:

1. В качестве модуля p лучше всего брать безопасные или «оптимальные» числа. Для эффективного поиска безопасных чисел выше предложен соответствующий алгоритм.
2. Генерация безопасных чисел занимает гораздо больше времени, чем генерация «оптимальных», поэтому первые следует генерировать заранее.
3. Для безопасных чисел есть возможность ускорения операций за счёт использования двойки в качестве порождающего элемента.
4. Поскольку размер секретного ключа сильно влияет на скорость операций в системе, мы рекомендуем выбирать секретный ключ минимально необходимого размера.
5. Выбрать рабочую группу в зависимости от требований семантической и долгосрочной безопасности.

Для дальнейшего изучения хотелось бы выделить следующие направления:

1. Вопросы кодирования для варианта, использующего подгруппы;
2. Исследование генерации ключей для варианта криптосистемы на эллиптических кривых.

Литература

1. Allen B. Implementing several attacks on plain ElGamal encryption. Iowa State University, 2008.
2. Boneh D., Joux A., Nguyen P. Q. Why textbook elgamal and rsa encryption are insecure // In ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security. London: Springer, 2000. P. 30–43.
3. Chevallier-Mames B., Paillier P., Pointcheval D. Encoding-Free ElGamal Encryption Without Random Oracles // Public Key Cryptography – 2006. Berlin: Springer, 2006.
4. Lee C. H., Lee P. J. A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup // Advances in Cryptology. CRYPTO, 1997.
5. Menezes A. J., Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. CRC Press, 1997.
6. Oorschot P. C., Wiener M. J. On Diffie-Hellman Key Agreement with Short Exponents // Proc. Eurocrypt'96. 1996.
7. Reyzin L. Diffie-Hellman, ElGamal, and a Bit of History // Lecture Notes for Boston University CAS CS 538. 2004.
8. Wiener M. J. Safe Prime Generation with a Combined Sieve. 2003.

Математические структуры и моделирование

Выпуск 22

Журнал

Редактор *Л. Ф. Платоненко*
Технический редактор *А. Ю. Углиржс*
Художественное оформление *В. В. Коробицын*

Подписано в печать 10.02.2011. Формат 60 × 84/8.
Печ. л. 17,4. Усл. печ. л. 16,2. Уч.-изд. л. 11,5.
Тираж 100 экз. Заказ № 88.

Издательство Омского госуниверситета
644077, Омск-77, пр. Мира, 55а
Отпечатано на полиграфической базе ОмГУ
644077, Омск-77, пр. Мира, 55а