Федеральное государственное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований Российской академии наук»
(ФГУ ФНЦ НИИСИ РАН)

ТРУДЫ НИИСИ РАН

ТОМ 8 № 2

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2018

Редакционный совет ФГУ ФНЦ НИИСИ РАН:
В.Б. Бетелин (председатель),
Е.П. Велихов, В.А. Галатенко, В.Б. Демидович (отв. секретарь), Б.В. Крыжановский,
А.Г. Кушниренко, А.Г. Мадера, М.В. Михайлюк, В.Я. Панченко, В.П. Платонов,
В.Н. Решетников

Главный редактор журнала:
В.Б. Бетелин

Научный редактор номера:
М.В.Михайлюк

Тематика номера:
Математические модели в физике, вопросы программирования, математическое моделирование и визуализация, математические исследования и вопросы численного анализа.

Журнал публикует оригинальные статьи по следующим областям исследований: математическое и компьютерное моделирование, обработка изображений, визуализация, системный анализ, методы обработки сигналов, информационная безопасность, информационные технологии, высокопроизводительные вычисления, оптико-нейронные технологии, микро- и наноэлектроника, математические исследования и вопросы численного анализа, история науки и техники.

The topic of the issue:
Mathematical models in physics, problems in programming. mathematical modeling and visualization, mathematical researches and problems of numerical analysis.

The Journal publishes novel articles on the following research arias: mathematical and computer modeling, image processing, visualization, system analysis, signal processing, information security, information technologies, high-performance computing, optical-neural technologies, micro- and nanoelectronics, mathematical researches and problems of numerical analysis, history of science and of technique.

Заведующий редакцией: Ю.Н.Штейников

Издатель: ФГУ ФНЦ НИИСИ РАН,
117218, Москва, Нахимовский проспект 36, к. 1

© Федеральное государственное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук», 2018 г.

СОДЕРЖАНИЕ

I. МАТЕМАТИЧЕСКИЕ МОДЕЛИ В ФИЗИКЕ

Б.В.Крыжановский, Л.Б.Литинский. Спектральная плотность для модели Изинга на гиперкубе ……... 4
М.Ю. Мальсагов, Я.М. Карандашев, Б.В. Крыжановский. Изменения свойств модели Изинга и её термодинамических характеристик при добавлении шума в матричные элементы …… 19
А.Г. Мадера. Воздействие тепловой обратной связи на температуру кристалла микросхемы ……..... 26
Н. В. Масальский. Особенности аномальной подвижности в полностью обедненных КНИ транзисторах …… 29

II. ВОПРОСЫ ПРОГРАММИРОВАНИЯ

В.А.Галатенко, К.А.Костюхин, М.Д.Дзабраев. Тестирование инструментальных средств разработки ПО …… 35
М.Г. Фуругян. Составление многопроцессорных расписаний с ограничениями на связи между процессорами ……... 46
А.А. Бурцев. Разработка собственных управляющих конструкций в среде ДССП для троичной машины …… 52
Н.О.Бесшапошников, М.А.Кузьменко, А.Г.Леонов, М.А.Матюшин. Некоторые вопросы эффективности детерминированных алгоритмов распознавания образов с помощью библиотеки OpenCV …… 65

III. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ВИЗУАЛИЗАЦИЯ

М.В. Михайлюк, Е.В. Страшнов, Д.М. Логинов. Моделирование датчиков в системах виртуального окружения …… 70
Д.А. Кононов, Р.О. Пономарев. Синтез сценариев управления чрезвычайными ситуациями …… 77
Р.О. Пономарев, Д.А. Кононов. Новые функции управления чрезвычайными ситуациями в сложных организационно-технических системах ……... 82
Е.В. Страшнов, Д.В. Омельченко. Определение коллизий аппроксимирующих цилиндров со сферами и плоскостью …… 92
Т. И. Шарипова, В. Г. Редько, Г. А. Бесхлебнова. Модель автономного агента, строящего и использующего когнитивную карту ……... 98
З. Б. Сохова, В. Г. Редько. Исследование процессов самоорганизации в эволюционной модели прозрачной децентрализованной экономики …….. 103
И.В. Афанаскин, С.Г.Вольпин, А.В.Королев. Использование метода последовательных приближений при композиционном моделировании разработки месторождений углеводородов для определения концентраций компонентов ……. 111
К.Д. Ашмян, А.К. Пономарев, О.В. Ковалева. Развитие методики оценки фазового состояния парафинов в пластовых нефтях ……..... 120
М.В.Михайлюк, А.В.Мальцев, П.Ю.Тимохин. Методы стереовизуализации результатов неустойчивого вытеснения нефти из пористых средств ……... 125

IV. МАТЕМАТИЧЕСКИЕ ИССЛЕДОВАНИЯ И ВОПРОСЫ ЧИСЛЕННОГО АНАЛИЗА

Ю.В.Кузнецов, М.М.Петрунин, Ю.Н.Штейников. О множестве вида A/A для плотных подмножеств интервала [1, n] …….. 130

Спектральная плотность для модели Изинга на гиперкубе

Б.В Крыжановский [1], Л.Б. Литинский[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1] kryzhanov@mail.ru, [2] litin@mail.ru

Аннотация. Для модели Изинга на гиперкубе изучаются два способа аппроксимации спектральной плотности $D(E)=\exp(N\cdot\Psi(E))$, которая описывает вырожденность каждого значения энергии E. В первом подходе показатель экспоненты $\Psi(E)$ аппроксимируют полиномами четной степени, во втором – с помощью нашего метода n-окрестностей. Выяснилось, что графики свободной энергии мало зависят от способа аппроксимации. А ее производные, поведением которых и определяются фазовые переходы, зависят от способа аппроксимации существенно. С полиномиальной аппроксимацией оказывается неразрывно связанным то, что система попадает в основное состояние при конечной температуре – что противоречит основным физическим принципам. Аппроксимация с помощью метода n-окрестностей для малых размерностей решетки работает плохо, но для больших размерностей дает хорошее согласие с результатами численного эксперимента.

Ключевые слова: модель Изинга, свободная энергия, спектральная плотность, метод n-окрестностей.

1. Введение

Самый естественный способ вычисления статистической суммы для большого числа $N>>1$ спинов - $Z_N(\beta)=\sum_{\text{all }\mathbf{s}}\exp\ -\beta NE(\mathbf{s})$ - состоит в следующем. Пусть E_r есть различные значения энергии состояний $\mathbf{s}=(s_1,s_2,...,s_N)$, а D_r есть вырождение энергии E_r (число различных состояний с такой энергией). В статистике D_r называют частотой энергии E_r. Мы используем термин, заимствованный из физики и будем называть совокупность всех значений D_r спектральной плотностью: $\sum_r D_r=2^N$.

Если E_r и D_r известны, $Z_N(\beta)$ записывается как сумма по различным значениям энергии:

$$Z_N(\beta)=\sum_r D_r\exp(-\beta NE_r)\,. \qquad (1.1)$$

Проблема в том, что спектральная плотность D_r известна лишь в исключительных случаях, например:

для модели среднего поля с q ближайшими соседями [1]:

$$E_r=-\frac{qJ}{2}\frac{(N-2r)^2-N}{N(N-1)},D_r=2C_N^r,r=0,...N/2;$$

для одномерной модели Изинга [2]:

$$E_r=-J\left(1-\frac{4r}{N}\right),\ D_r=2C_N^{2r},\ r=0,1,...N/2\,.$$

Здесь J - константа взаимодействия между соседними спинами.

Как правило, точное выражение для D_r неизвестно. Можно аппроксимировать спектральную плотность D_r, положив в основу приближения ту или иную идею. В настоящей работе сопоставляются два способа аппроксимации D_r, и сравниваются получающиеся результаты.

Второй и третий разделы посвящены двумерной модели Изинга. Второй раздел начинается с вычисления точной спектральной плотности, с которой будем сравнивать два ее приближения. Точную спектральную плотность для двумерной модели Изинга можно построить, восстанавливая ее с помощью свободной энергии, которая для двумерной модели Изинга известна со времен классической работы [3]. Метод восстановления спектральной плотности изложен в [4-6]. Мы использовали подход [6], чтобы построить точную спектральную плотность – эталон, с которым будем сравнивать обе аппроксимации, изучаемые в нашей работе. Затем во втором разделе излагается полиномиальный подход к задаче аппроксимации спектральной плотности. Его существо состоит в следующем. Условимся записывать спектральную плотность в виде экспоненты $D_r=\exp\ N\varphi_r$, а показатель экспоненты φ_r будем именовать экспонентой спектральной плотности. Авторы [7]

моделировали на компьютере распределение D_r для модели Изинга и установили, что для больших размерностей $N \sim 10^3$ распределение имеет колоколообразную форму, симметричную относительно значения $E = 0$, с дисперсией σ^2, и длинными хвостами, прижатыми к оси абсцисс. Сравнение φ_r с гауссовым показателем экспоненты $g_r = \ln 2 - E_r^2 / 2\sigma^2$ показало, что на «хвостах» значения φ_r сильно отличаются от g_r. Иначе говоря, аппроксимировать истинное распределение гауссовой плотностью нельзя. Тогда авторы [7] наилучшим образом аппроксимировали φ_r полиномами четной степени

$$\varphi_r = \text{const} + a \cdot E_r^2 + b \cdot E_r^4 + c \cdot E_r^6 . \quad (1.2)$$

Для двумерной модели Изинга хорошего качества аппроксимации можно добиться уже с помощью первых трех слагаемых в (1.2), а для трехмерной модели Изинга необходимо использовать все слагаемые. Подогнав константы, и обеспечив хорошее качество аппроксимации, авторы [7] получили приближение спектральной плотности $D_{pl}(E)$, вычислили свободную энергию и исследовали ее поведение. Критические значения обратной температуры удалось получить с погрешностью ~12%. Мы использовали аппроксимацию (1.2) и детально повторили исследование авторов [7]. Мы нашли, что система оказывается в основном состоянии при конечной температуре - что неприемлемо из физических соображений. Кроме того, фазовый переход оказался первого рода, что противоречит точному результату Онсагера. Второй раздел завершается изложением этих наших результатов.

В разделе 3 изложен другой подход к аппроксимации D_r, который опирается на развиваемый нами метод n-окрестностей [8,9]. В этом подходе фиксируют начальную конфигурацию $\mathbf{s}_0 \in \mathbf{R}^{\mathbf{N}}$, и тогда остальные конфигурации распределяются по ее n-окрестностям Ω_n: в Ω_n объединены конфигурации, отличающиеся от начальной $\mathbf{s}_0$ противоположными значениями n спинов:

$$\Omega_n = \{\mathbf{s} : \mathbf{s}, \mathbf{s}_0 = N - 2n, 0 \le n \le N\} .$$

Обозначим через $D^{(n)}(E)$ истинное распределение энергий состояний, принадлежащих n-окрестности Ω_n. В общем случае $D^{(n)}(E)$ неизвестно, однако можно получить точные выражения для среднего значения E_n и дисперсии σ_n^2 распределения $D^{(n)}(E)$, выразив эти моменты через характеристики матрицы связи и начальной конфигурации $\mathbf{s}_0$ [8]. Основная идея метода n-окрестностей состоит в замене неизвестного нам истинного распределения $D^{(n)}(E)$ гауссовым распределением со средним E_n и дисперсией σ_n^2:

$$D^{(n)}(E) \sim \exp\left(-\frac{N}{2} \left(\frac{E - E_n}{\sigma_n} \right)^2 \right). \quad (1.3)$$

Обоснование метода n-окрестностей дано в [8], границы его применимости для модели Изинга исследованы в [9]. Если просуммировать «парциальные» плотности (1.3) по всем n-окрестностям, получим аппроксимацию спектральной плотности, порожденную методом n-окрестностей $D_{nv}(E)$ (здесь “nv” от n-vicinity):

$$D_{nv}(E) \sim \sum_{n=0}^{N} C_N^n \cdot \exp\left(-\frac{N}{2} \left(\frac{E - E_n}{\sigma_n} \right)^2 \right).$$

Применению метода n-окрестностей для планарной модели Изинга посвящен третий раздел. В этом подходе также предсказывается фазовый переход первого рода (что неправильно), однако отсутствует нефизичный переход в основное состояние при конечной температуре.

Результаты исследования трехмерной модели Изинга изложены в четвертом разделе. Поскольку точное выражение для свободной энергии в этом случае неизвестно, отсутствует и эталонная спектральная плотность, с которой можно сравнивать ее аппроксимации. Здесь мы сопоставим друг с другом обе аппроксимации спектральной плотности $D_{pl}(E)$ и $D_{nv}(E)$, а также проанализируем построенные с их помощью варианты свободной энергии. Полиномиальная аппроксимация дает здесь негативные результаты, подобные тем, что получились для планарной модели Изинга. Напротив, аппроксимация методом n-окрестностей дает хорошее совпадение с результатами компьютерного эксперимента, которые для кубической модели являются общепринятыми [10]. Обсуждение полученных результатов происходит в пятом разделе. Здесь объясняется, почему противоречащий физике переход системы в

основное состояние при конечной температуре имманентно присущ полиномиальной аппроксимации - это делает данный подход бесперспективным. Кроме того, обсуждаются способы дальнейшего развития метода n-окрестностей. Технические детали вынесены в Приложение.

Обозначения. Всюду в дальнейшем энергия вычисляется в расчете на один спин:

$$E(\mathbf{s}) = -\frac{1}{2N}\sum_{i\neq j}^{N} T_{ij} s_i s_j \,. \qquad (1.4)$$

Матрицы связи отвечают самому простому варианту модели Изинга: взаимодействуют ближайшие спины, величина взаимодействия $J=1$, граничные условия периодические. Для метода n-окрестностей в качестве начальной конфигурации удобно взять основное состояние: $\mathbf{s}_0 = (1,1,...,1) \in R^N$. Число ближайших соседей у каждого спина $q = 2d$, где d - размерность решетки. Легко вычислить энергию основного состояния:

$$E_0 = E(\mathbf{s}_0) = -\frac{q}{2}\,. \qquad (1.5)$$

2. Двумерная модель Изинга: аппроксимация полиномом.

В этом разделе сначала будет построена точная спектральная плотность

$$D(E) = \exp(N \cdot \Psi(E))\,, \qquad (2.1)$$

а затем - $D_{pl}(E) = \exp\, N \cdot \Psi_{pl}(E)$, ее приближение, основанное на аппроксимации экспоненты $\Psi(E)$ полиномом четвертой степени $\Psi_{pl}(E)$. Сравнение точной $\Psi(E)$ с $\Psi_{pl}(E)$, и сопоставление графиков свободной энергии позволит уяснить некоторые особенности полиномиальной аппроксимации.

2.1. Точная спектральная плотность.

Используем традиционное обозначение для безразмерной свободной энергии $f(\beta)$, стандартным образом выражая ее через статистическую сумму $Z_N(\beta)$:

$$f(\beta) = -\lim_{N\to\infty} \frac{\ln Z_N(\beta)}{N}\,. \qquad (2.2)$$

Здесь β есть обратная температура: $\beta = 1/T$.

Пусть свободная энергия нам известна. Тогда экспонента спектральной плотности $\Psi(E)$ вычисляется по формулам (см. [4]-[6]):

$$\Psi(E) = \beta E - f(\beta)\,, \text{ где } E = f'(\beta)\,. \qquad (2.3)$$

При этом оказывается, что

$$\Psi'(E) = \beta,\ \Psi''(E) = \left[f''(\beta)\right]^{-1} \le 0\,. \qquad (2.4)$$

Соотношения (2.3) (известные как преобразования Лежандра) позволяют построить экспоненту спектральной плотности, а соотношения (2.4) позволяют с хорошей точностью построить ее производные.

Отметим общие свойства $\Psi(E)$. Свободная энергия $f(\beta)$ является невозрастающей функцией – следовательно, соотношение (2.3) определяет $\Psi(E)$ только для отрицательных энергий: $E \le 0$. Для модели Изинга $\Psi(E)$ можно продолжить на положительную полуось, поскольку $\Psi(E) = \Psi(-E)$. Далее, когда $\beta \to \infty$ спиновая система стремится в основное состояние, энергия которого обозначена у нас как E_0 (1.5): $\lim_{\beta\to\infty} E(\beta) = E_0$. Поскольку при $\beta \to \infty$ свободная энергия ведет себя как βE_0, из первого соотношения (2.3) следует, что

$$\lim_{\beta\to\infty} \left[\beta E - f(\beta)\right] = \lim_{E\to E_0} \Psi(E) = 0.$$

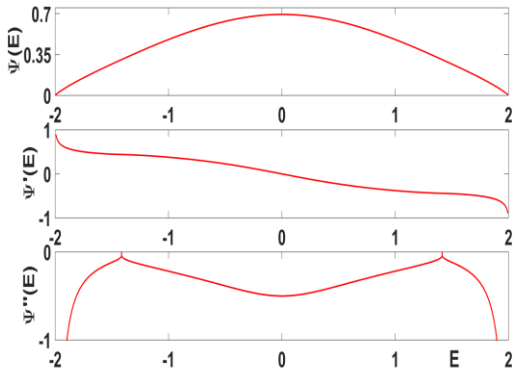


Рис. 1. Для двумерной модели Изинга экспонента $\Psi(E)$ спектральной плотности и ее производные.

Иными словами, экспонента $\Psi(E)$ на краях энергетического интервала стремится к 0.

Первое соотношение в (2.4) свидетельствует о том, что для всех энергий $E \in (E_0, 0)$ первая производная $\Psi(E)$

положительна. Из этой же формулы следует и другой важный вывод об устройстве $\Psi(E)$: при $E \to E_0$ первая производная $\Psi(E)$ стремится к бесконечности: $\lim_{E \to E_0} \Psi'(E) = \infty$.

Как известно, вторая производная свободной энергии пропорциональна взятой с обратным знаком дисперсии распределения энергии при данной температуре.

Дисперсия неотрицательна, чем и определяется знак неравенства во втором соотношении (2.4).

Отметим, что поскольку вторая производная $\Psi(E)$ неположительна, ее первая производная $\Psi'(E)$ должна быть монотонно невозрастающей функцией.

На Рис.1 для двумерной модели Изинга показаны графики экспоненты $\Psi(E)$ и ее производных. На графиках можно наблюдать перечисленные выше свойства этих функций. В работе [6] показано, что на левом краю интервала, в районе $E_0 = -2$, функция $\Psi(E)$ ведет себя как: $\Psi(E) \sim -(E - E_0)\ln(E - E_0)$. Тогда для производных здесь получаются оценки: $\Psi'(E) \sim -\ln(E - E_0)$, $\Psi''(E) \sim -1/(E - E_0)$. Иными словами, на левом краю интервала первая производная $\Psi'(E)$ логарифмически стремится к положительной бесконечности, а вторая производная стремится к отрицательной бесконечности.

Плавно меняющаяся первая производная $\Psi'(E)$ имеет перегиб в точке $E_c = E_0/\sqrt{2}$ - вторая производная обращается при этом в ноль: $\Psi''(E_c) = 0$. Учитывая второе из соотношений (2.4), получаем, что вторая производная свободной энергии при соответствующем значении $\beta = \beta_c$ имеет бесконечный разрыв: $f''(\beta) \to \infty$ при $E \to E_c$. Иными словами, при $\beta = \beta_c$ происходит фазовый переход второго рода.

Интервал энергий вблизи E_c, где вторая производная $\Psi''(E)$ выходит на ноль, очень узок – на графике $\Psi''(E)$ области сингулярности выглядят как миниатюрные рожки. В районе начала координат $E = 0$ для функции $\Psi(E)$ справедлива асимптотика $\Psi(E) = \ln 2 - E^2/4$ (см. [6]). Тогда имеем:

$$\Psi'(E) = -E/2, \ \Psi''(E) = -1/2. \quad (2.5)$$

Эти оценки нам пригодятся.

2.2. Полиномиальная аппроксимация.

Авторы [7] моделировали спектральную плотность для двумерной модели Изинга и установили, что с хорошей степенью точности ее можно представить в виде

$$D_{pl}(E) \sim \exp\ N \cdot \Psi_{pl}(E)\ , \ E \in [-2, +2]$$

где $\Psi_{pl}(E) = \ln 2 - a \cdot E^2 + b \cdot E^4$, (2.6)

$a = 1/(2.16001)^2$, $b = 1/(2.09255)^6$.

Значения коэффициентов получаются в результате наилучшей аппроксимации экспериментальной экспоненты четным полиномом четвертой степени. Поясним, что авторы [7] не пользовались аналитическими выражениями (2.3), (2.4). Для подгонки они использовали экспериментальный аналог экспоненты $\Psi(E)$, полученный в численных экспериментах при больших значениях N.

В левом столбце Рис. 2 показаны графики истинной экспоненты $\Psi(E)$ (2.3) и ее полиномиальной аппроксимации $\Psi_{pl}(E)$ (2.6), а также их производных. В правом столбце приведены графики свободной энергии, отвечающие этим двум вариантам спектральной плотности – см. ниже анализ этих графиков в пп. 2.3, 2.4. На верхней панели даны графики $\Psi_{pl}(E)$ и $\Psi(E)$, на средней панели – графики их производных $\Psi'_{pl}(E)$ и $\Psi'(E)$, на нижней панели – графики вторых производных $\Psi''_{pl}(E)$ и $\Psi''(E)$. Сплошной линией даются кривые, относящиеся к точной спектральной плотности $\Psi(E)$, штрих-пунктирной – относящиеся к $\Psi_{pl}(E)$.

Верхняя панель свидетельствует о том, что $\Psi_{pl}(E)$ в целом хорошо аппроксимирует экспоненту $\Psi(E)$.

Кривые практически совпадают в районе энергий $E \approx -1.5$ и $E \approx 0$, и различаются только в районе энергии основного состояния $E_0 = -2$.

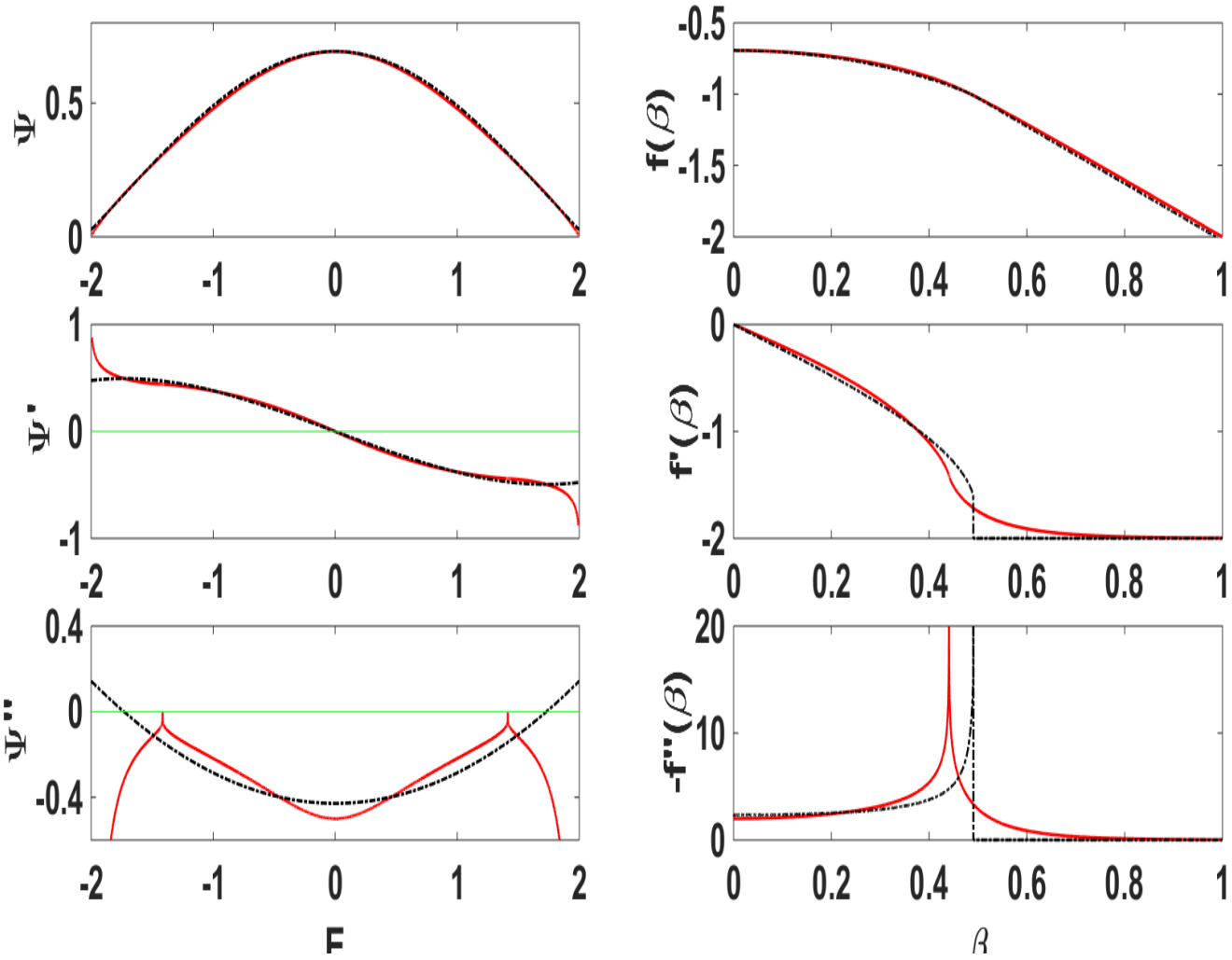


Рис. 2. Для двумерной модели Изинга сопоставление точного решения и полиномиальной аппроксимации. Левый столбец – экспонента $\Psi(E)$ спектральной плотности (сплошная линия) и $\Psi_{pl}(E)$ (штрих-пунктирная), а также их производные. Правый столбец - свободная энергия $f(\beta)$, отвечающая экспоненте $\Psi(E)$ (сплошная линия), и $f_{pl}(\beta)$, отвечающая $\Psi_{pl}(E)$ (штрих-пунктир) – см. текст.

Ситуация выглядит менее оптимистично, если сопоставить производные этих функций – см. графики $\Psi'(E)$ и $\Psi'_{pl}(E)$ на средней панели рисунка. Вообще говоря, на интервале $(-1.8,+1.8)$ графики отличаются друг от друга не так уже сильно. Плохо, что при $E \to E_0$ первая производная $\Psi'_{pl}(E)$ принимает конечное значение, а не стремится подобно $\Psi'(E)$ к бесконечности. Ниже мы увидим, что конечное значение производной $\Psi'_{pl}(E_0)$ является причиной того, что при некотором значении обратной температуры система оказывается в основном состоянии. Такое поведение не имеет физического смысла: при конечной температуре система не может находиться в основном состоянии.

Для истинной спектральной плотности $D(E)$ именно стремление производной $\Psi'(E)$ к бесконечности при $E \to E_0$ гарантирует, что при $\beta \to \infty$ система стремится к основному состоянию асимптотически, не оказываясь в нем ни при каком конечном значении β.

На нижней панели рисунка показаны графики вторых производных $\Psi''_{pl}(E)$ и $\Psi''(E)$. Парабола $\Psi''_{pl}(E)$ плохо воспроизводит замысловатый ход графика $\Psi''(E)$. В точке

$$E_i = -\sqrt{\frac{a}{6b}} \approx -1.7318 \qquad (2.7)$$

вторая производная $\Psi''_{pl}(E)$ равна нулю - E_i является точкой перегиба полинома $\Psi_{pl}(E)$ (i – от английского inflection). Слева от E_i вторая производная $\Psi''_{pl}(E)$ положительна, а поскольку общие соотношения (2.4) запрещают второй производной положительные значения, область энергий от E_0 до E_i является запретной. Таким образом, аппроксимация $\Psi(E)$ полиномом $\Psi_{pl}(E)$ имеет смысл только на внутреннем интервале $E_i \le E \le |E_i|$.

2.3. Уравнение состояния и его решение.

Покажем, как поведение свободной энергии определяется свойствами спектральной плотности. Перейдем к функции $f_0(E) = -\Psi_{pl}(E)$:

$$f_0(E) = -\ln 2 + a \cdot E^2 - b \cdot E^4, \qquad (2.8)$$

и придадим статистической сумме вид:

$$Z(\beta) \sim \int e^{-\beta N E} D_{ply}(E) dE = \\ = \int e^{-N \cdot f_0(E) + \beta E} dE \sim e^{-N \cdot f(\beta)} .$$

Здесь $f(\beta)$ есть безразмерная свободная энергия (2.2) - она вычисляется минимизацией по E функции $f_0(E) + \beta \cdot E$:

$$f(\beta) = \min_E \; f_0(E) + \beta \cdot E \; .$$

Уравнение состояния, определяющее значение свободной энергии, имеет вид:

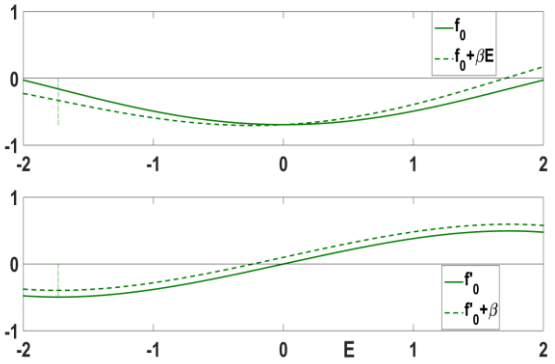
$$f_0'(E) + \beta = 0 . \qquad (2.9)$$

Сразу же отметим очевидный недостаток полиномиальной аппроксимации: производная $f_0'(E)$ на конечном интервале $E_0 \le E \le 0$ ограничена, и уравнение состояния (2.9) не имеет решения при $\beta > -\max f_0'$.

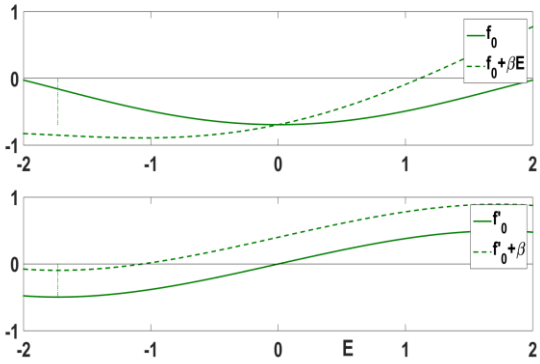
Решение уравнения (2.9) будем анализировать графически. На верхних панелях Рис. 3a), b) показаны графики функций $f_0(E)$ и $f_0(E) + \beta \cdot E$ для двух различных значений параметра β: $\beta = 0.1$ и $\beta = 0.4$. На нижних панелях даны графики их производных $f_0'(E)$ и $f_0'(E) + \beta$. График $f_0'(E) + \beta$ есть просто сдвинутый вверх на величину β график производной $f_0'(E)$.

Минимум функции $f_0(E)$ (2.8) находится в точке $E_z = 0$: $f_0(0) = -\ln 2$. Два ее максимума находятся в точках $\pm\sqrt{a/2b} \approx \pm 3$ за пределами интервала энергий $[-2, +2]$. Для рассмотрений нам потребуется отрицательная точка максимума – обозначим ее $E_M = -\sqrt{a/2b}$.

Точка перегиба E_i функции $f_0(E)$ определяется уравнением $f_0''(E_i) = 0$ - см. для нее выражение (2.7). В точке перегиба меняется знак выпуклости функции $f_0(E)$. Для нас важнее, что точка перегиба функции $f_0(E) + \beta \cdot E$ не зависит от β, и остается неподвижной, когда β меняется.



a). $\beta = 0.1$.



b). $\beta = 0.4$.

Рис. 3. На верхних панелях - графики функций $f_0(E)$ (сплошная линия) и $f_0(E) + \beta \cdot E$ (штриховая линия) для двух значений параметра β, на нижних панелях – графики производных этих функций по E. Вертикальная точечная прямая отмечает положение точки перегиба E_i (2.7).

Будем следить за минимумом функции $f_0(E)+\beta\cdot E$. В начале, при $\beta=0$, минимум функции находится в точке $E_z=0$. Прибавление к $f_0(E)$ слагаемого $\beta\cdot E$ делает суммарную функцию несимметричной – см. штриховые линии на верхних панелях Рис. 3. Точка минимума – обозначим ее $E_z(\beta)$ - смещается из нуля влево. График производной $f_0'(E)+\beta$ показывает, что с ростом β точка минимума $E_z(\beta)$ все время сдвигается влево, в направлении границы интервала E_0 - см. на рисунке нижние панели. Напротив, точка максимума функции – обозначим ее $E_M(\beta)$ - движется слева направо, в направлении начала координат. При $\beta=4E_i a/3\approx 0.4949$ точки $E_z(\beta)$ и $E_M(\beta)$ должны встретиться в неподвижной точке перегиба E_i. Однако, в данном случае этого не произойдет, потому что чуть раньше, когда β станет равным $\beta_j=0.49032$, левый конец графика $f_0(E)+\beta\cdot E$ опустится вниз настолько, что сравняется с минимумом в точке $E_z(\beta)$:

$$f_0(E_0)+\beta_j E_0 = f_0\ E_z(\beta_j)\ +\beta_j E_z(\beta_j)\,. \quad (2.10)$$

На верхней панели Рис. 4 в увеличенном масштабе показано, как ведет себя функция $f_0(E)+\beta_j E$ на левом конце интервала. При $\beta=\beta_j$ глобальный минимум функции перепрыгивает из точки $E_z(\beta_j)\approx -1.6$ в граничную точку интервала $E_0=-2$ (индекс «*j*» от английского jump). В дальнейшем, сколько бы ни возрастало значение β, глобальный минимум функции $f_0(E)+\beta\cdot E$ останется в E_0.

Такое поведение обусловлено тем, что производная $f_0'(E_0)$ на левом конце интервала имеет конечное значение. Взяв β достаточно большим, значение производной на левом конце интервала можно сделать положительным: $f_0'(E_0)+\beta>0$, и добиться того, что минимум функции $f_0(E)+\beta\cdot E$ окажется в точке E_0. Если бы значение производной $f_0'(E)$ на границе интервала стремилось к отрицательной бесконечности: $f_0'(E\to E_0)\to-\infty$, никаким конечным значением β нельзя было бы сделать производную $f_0'(E_0)+\beta$ положительной.

2.4. Поведение свободной энергии.

В правом столбце Рис. 2 показаны графики свободной энергии и ее производных по β для истинной спектральной плотности $\Psi(E)$ (сплошная линия) и ее полиномиальной аппроксимации $\Psi_{pl}(E)$ (штрих-пунктир). Свободную энергию,

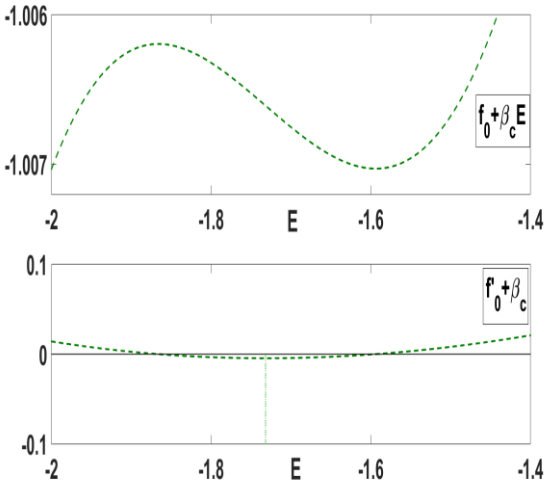


Рис. 4. $f_0(E)+\beta E$ (верхняя панель) и ее производная (нижняя панель), когда $\beta=\beta_j=0.49032$. Вертикальная точечная прямая отмечает положение точки перегиба E_i (2.7).

отвечающую точному решению, обозначим $f(\beta)$; аналогичную характеристику, отвечающую полиномиальной аппроксимации, обозначим $f_{pl}(\beta)$.

Видно, что графики $f(\beta)$ и $f_{pl}(\beta)$ почти неразличимы на всем интервале изменения β – см. верхнюю правую панель Рис.2. Иное дело – поведение их производных. Первая производная $f'(\beta)$ равна среднему значению энергии при данной температуре: $U(\beta)=f'(\beta)$. Когда $\beta\to\infty$ первая производная должна асимптотически стремиться к $E_0=-2$. Именно так ведет себя сплошная кривая на средней правой панели Рис. 2: она плавно убывает от значения, равного нулю, и имеет перегиб в точке $\beta_c\approx 0.4407$. Аналогичная характеристика $U_{pl}(\beta)=f_{pl}'(\beta)$ поначалу ведет себя очень похоже, но при $\beta\approx 0.49$ скачком принимает минимальное значение

$E_0 = -2$ - этим скачком управляет уравнение (2.10). При дальнейшем увеличении β значение $U_{pl}(\beta)$ остается равным E_0. Такое поведение $U_{pl}(\beta)$ противоречит физическим принципам. Пользоваться $U_{pl}(\beta)$ для аппроксимации внутренней энергии $U(\beta)$ можно только до значения $\beta \approx 0.4$. То же справедливо и в отношении вторых производных $f''(\beta)$ и $f''_{pl}(\beta)$ - только до $\beta \approx 0.4$ их графики мало отличаются друг от друга. Вне этого интервала сравнивать графики просто не имеет смысла.

3. Двумерная модель Изинга: метод *n*-окрестностей.

Здесь мы получим общие выражения для аппроксимации спектральной плотности методом n-окрестностей для модели Изинга с числом ближайших соседей q:

$$D_{nv}(E) = \exp\ N \cdot \Psi_{nv}(E)\ .$$

Затем для двумерной модели Изинга ($q = 4$) сопоставим $\Psi_{nv}(E)$ и $\Psi(E)$, и проанализируем графики свободной энергии.

3.1. Аппроксимации спектральной плотности.

Основная идея метода n-окрестностей изложена во Введении. Приведем выражения, позволяющие аппроксимировать спектральную плотность для модели Изинга.

В качестве начальной конфигурации $\mathbf{s}_0$ выбираем основное состояние: $\mathbf{s}_0 = (1,1,...,1) \in R^N$. Остальные конфигурации распределяются по n-окрестностям Ω_n (см. Введение). Заменим неизвестное нам истинное распределение энергий состояний из Ω_n гауссовым распределением с известными выражениями для первых двух моментов распределения E_n и σ_n^2 [8], [9]. Для статистической суммы можно записать:

$$Z_N = \sum_{n=0}^{N} C_N^n \left(\frac{1}{C_N^n} \sum_{\mathbf{s}\in\Omega_n} e^{-\beta NE(\mathbf{s})} \right). \qquad (3.1)$$

Устремляя N к бесконечности, перейдем в последнем выражении от суммирования к интегрированию. Для этого введем переменную $x = n/N$ ($x \in [0,1]$), и воспользуемся формулой Стирлинга:

$$C_N^{xN} \sim \exp(-N \cdot S(x)),\ \text{где}$$
$$S(x) = x\ln x + (1-x)\ln(1-x).$$

В выражении (3.1) суммирование по n заменяется интегрированием по x, а усреднение по состояниям из Ω_n - интегрированием по гауссовой плотности со средним $E_x = \lim_{N\to\infty} E_n$ и дисперсией $\sigma_x^2 = \lim_{N\to\infty} \sigma_n^2$:

$$\frac{1}{C_N^n} \sum_{\mathbf{s}\in\Omega_n} e^{-\beta NE(\mathbf{s})} \to$$
$$\frac{1}{\sqrt{2\pi}\sigma_x} \int_{E_0}^{|E_0|} \exp\left[-\beta NE - \frac{1}{2}\left(\frac{E - E_x}{\sigma_x} \right)^2 \right] dE.$$

Энергия $E(\mathbf{s})$ вычисляется как в (1.4), $E_0 = E(\mathbf{s}_0)$ дается формулой (1.5), а

$$E_x = E_0(1-2x)^2,\ \sigma_x^2 = 8qx^2(1-x)^2/N\ .$$

Выражение (3.1) преобразуется теперь к виду:

$$Z_N(\beta) \sim \int_{E_0}^{|E_0|} e^{-\beta NE} dE \int_0^1 C_N^{xN} \exp\left(-\frac{1}{2}\left(\frac{E - E_x}{\sigma_x} \right)^2 \right) dx$$

В результате интегрирования по x получаем аппроксимацию спектральной плотности:

$$D_{nv}(E) \approx \int_0^1 e^{-N\varphi(E,x)} dx = \exp\ -N\varphi_m(E)\ , \quad (3.2)$$

где

$$\varphi(E,x) = S(x) + \frac{q}{4}\left(1 + \frac{E/E_0 - 1}{4x(1-x)} \right)^2, \qquad (3.3)$$
$$\varphi_m(E) = \min_{0\le x\le 1} \varphi(E,x).$$

В дальнейшем мы увидим, что пока значение параметра $\varepsilon = E/E_0$ меньше некоторого критического значения ε_c, минимум функции $\varphi(E,x)$ находится в точке $x_0 = 1/2$. Тогда для энергий из интервала $[\varepsilon_c E_0, 0]$ можно записать:

$$\varphi_m(E) = -\ln 2 + \frac{E^2}{q},\quad \varepsilon_c E_0 < E < 0\ . \qquad (3.4)$$

Для $E < \varepsilon_c E_0$ задачу (3.3) приходится решать численно. Отыскав для каждого значения E минимум функции $\varphi(E,x)$ получим искомую аппроксимацию экспоненты спектральной плотности $\Psi(E)$:

$$\Psi_{nv}(E) = -\varphi_m(E)\ . \qquad (3.5)$$

3.2. Планарная модель Изинга.

Обратимся к двумерной модели Изинга ($q=4$), и будем менять значение параметра E от 0 до $E_0=-2$.

Несложно убедиться, что пока E не очень велико по модулю – пока $E/E_0<0.43$, единственным минимумом функции $\varphi(E,x)$ является точка $x_0=1/2$.

Согласно (3.4) для таких энергий имеем:

$$\varphi_m(E)=-\ln 2+\frac{E^2}{4}. \qquad (3.6)$$

Заметим, что выражение для $\Psi_{nv}(E)$ на интервале $0.43E_0<E<0$ в точности совпадает с аналитическим представлением спектральной плотности $\Psi(E)$ в районе энергии $E=0$ – ср. (3.6) с формулами (2.5).

При $E\approx 0.4325\cdot E_0$ у функции $\varphi(E,x)$ появляется еще один минимум, а при $E\approx 0.438\cdot E_0$ глубина нового минимума становится равной глубине минимума $\varphi(E,x_0)$ - в этот момент происходит перескок глобального минимума из $x_0=1/2$ в точку нового минимума $x\approx 0.3$. Перескок приведет к скачкообразному изменению производной функции $\varphi_m(E)$. Дальнейшее стремление E к E_0 сопровождается движением точки минимума $\varphi(E,x)$ в направлении 0.

Подчеркнем, что величина минимума $\varphi_m(E)$ будет при этом логарифмически стремиться к 0 (см. Приложение).

В левом столбце Рис.5 показаны графики экспоненты $\Psi(E)$ (2.3) и функции $\Psi_{nv}(E)$ и их производных: на верхней панели - графики самих функций, на средней панели – графики их производных $\Psi'(E)$ и $\Psi'_{nv}(E)$, на нижней панели – графики вторых производных $\Psi''(E)$ и $\Psi''_{nv}(E)$. Сплошной линией, как и прежде, изображаются $\Psi(E)$ и ее производные, штриховой линией – $\Psi_{nv}(E)$ и ее производные.

Как и на аналогичном Рис. 2, графики на верхней панели очень похожи друг на друга. Наибольшее различие между $\Psi(E)$ и $\Psi_{nv}(E)$ наблюдается на отрезке $-1.5\le E\le -0.5$. Вне этого интервала графики практически совпадают.

На средней левой панели Рис. 5 показаны графики производных $\Psi'(E)$ и $\Psi'_{nv}(E)$.

В целом, можно было бы сказать, что и эти графики близки, если бы не особенность у производной $\Psi'_{nv}(E)$ в районе энергии $E_j\approx 0.438E_0=-0.876$. Пока E убывает от 0 до E_j, графиком $\Psi'_{nv}(E)$ является прямая линия – см. выражения (3.5), (3.6). Когда E становится равным E_j, производная $\Psi'_{nv}(E)$ меняется скачком: от значения $\Psi'_{nv}(E_{j+})\approx 0.44$ до $\Psi'_{nv}(E_{j-})\approx 0.4$.

Затем, пока E меняется от E_j до значения -1.05, производная $\Psi'_{nv}(E)$ уменьшается. Это означает, что вторая производная $\Psi''_{nv}(E)$ здесь положительна - что запрещено вторым из соотношений (2.4). Иными словами, при аппроксимации методом n-окрестностей интервал энергий $[-1.05,-0.876]$ является запрещенным. Перешагнув через него, мы опять попадаем в условия применимости нашего метода - вторая производная $\Psi''_{nv}(E)$ вновь становится отрицательной. Дальнейшее стремление E от значения -1.05 к границе E_0 сопровождается плавным изменением производной $\Psi'_{nv}(E)$, а при $E\to E_0$ производная $\Psi'_{nv}(E)$ стремится к бесконечности. Благодаря этому ни при каком конечном значении температуры система не окажется в основном состоянии, но будет асимптотически стремиться к нему - см. обсуждение этого вопроса выше.

На нижней панели Рис. 5 показаны графики вторых производных $\Psi''(E)$ и $\Psi''_{nv}(E)$. Здесь отчетливо видна область запрещенных энергий $[-1.05,-0.876]$ - там, где вторая производная $\Psi''_{nv}(E)$ положительна. Для энергий, меньших -1.5, график $\Psi''_{nv}(E)$ хорошо аппроксимирует ход $\Psi''(E)$, а при $E\to E_0$ $\Psi''_{nv}(E)$ стремится к $\Psi''(E)$. Хорошо воспроизводится значение $\Psi''(E)$ в нуле: согласно (2.5) $\Psi''(0)=-1/2$ и это в точности совпадает с $\Psi''_{nv}(0)\equiv -2/q$. Отметим, что на всем внутреннем интервале $0.43E_0<E<0$ график $\Psi''_{nv}(E)$ есть константа $\Psi''_{nv}(E)\equiv -2/q$, и заметно отличается от графика истинной второй производной $\Psi''(E)$. Это говорит о том, что

- по крайней мере, для $q = 4$ - центральная часть истинной спектральной плотности $D(E) = \exp\ N \cdot \Psi(E)$ не является гауссовой.

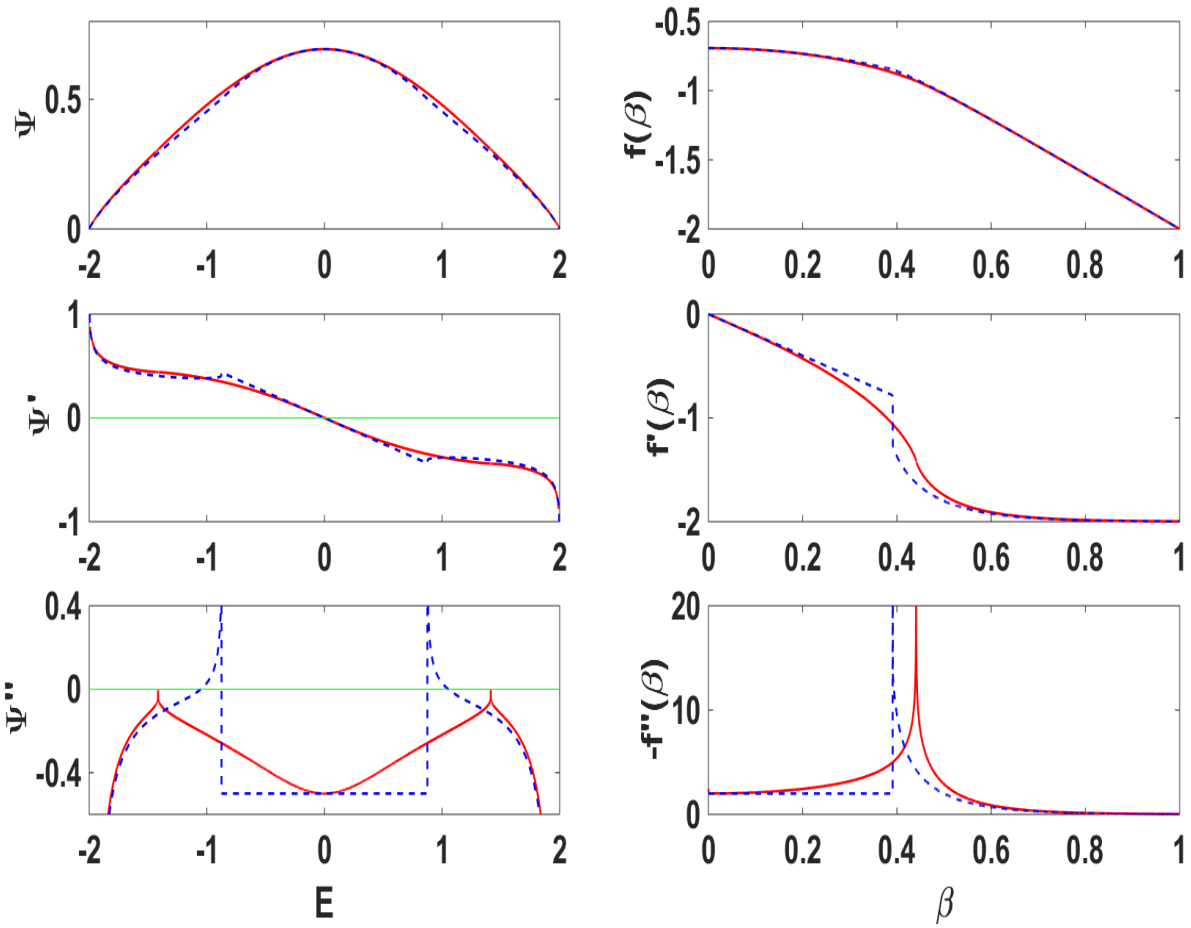


Рис. 5. Для двумерной модели Изинга сопоставление точного решения и аппроксимации методом n-окрестностей. Левый столбец – экспонента $\Psi(E)$ спектральной плотности (сплошная) и ее аппроксимация $\Psi_{nv}(E)$ (штриховая), а также их производные. Правый столбец - свободная энергия $f(\beta)$, отвечающая экспоненте $\Psi(E)$ (сплошная), и $f_{nv}(\beta)$, отвечающая $\Psi_{nv}(E)$ (штриховая).

3.3. Поведение свободной энергии.

В правом столбце Рис.5 показаны графики свободной энергии и ее производных по β для истинной спектральной плотности $\Psi(E)$ (сплошная линия) и ее аппроксимации методом n-окрестностей $\Psi_{nv}(E)$ (штриховая линия). Как и раньше, $f(\beta)$ означает свободную энергию, отвечающую точному решению, а $f_{nv}(\beta)$ - свободную энергию, отвечающую $\Psi_{nv}(E)$. Видно, что графики $f(\beta)$ и $f_{nv}(\beta)$ практически неразличимы на всем интервале изменения β – см. верхнюю правую панель рисунка.

Для значений $\beta < 0.4$, $f_{nv}(\beta)$ описывается простым аналитическим выражением: $f_{nv}(\beta) = -\ln 2 - \beta^2$. Первая производная $f'_{nv}(\beta)$ на этом участке зависит от β линейно - см. на правой средней панели штриховую прямую линию. Затем первая производная меняется скачком, после чего $f'_{nv}(\beta)$ плавно спадает, асимптотически приближаясь к $E_0 = -2$. Скачок производной $f'_{nv}(\beta)$ происходит при $\tilde{\beta}_c \approx 0.39125$, и отвечает фазовому переходу первого рода. При этом же значении β происходит разрыв

второй производной $f''_{nv}(\beta)$. Графики производных функций $f(\beta)$ и $f_{nv}(\beta)$ заметно различаются в районе критической температуры $\beta_c \approx 0.44$.

4. Трехмерная модель Изинга

ля трехмерной модели Изинга точная спектральная плотность неизвестна, поэтому мы построим два ее приближения - основанное на полиномиальной аппроксимации экспоненты - $D_{pl}(E) = \exp\ N \cdot \Psi_{pl}(E)$,- и полученное с помощью метода n-окрестностей $D_{nv}(E) = \exp\ N \cdot \Psi_{nv}(E)$. Сопоставление $\Psi_{pl}(E)$ и $\Psi_{nv}(E)$, а также отвечающих им графиков свободной энергии, позволит сделать некоторые выводы о качестве обеих аппроксимаций.

4.1. Аппроксимация экспоненты $\Psi(E)$ посредством $\Psi_{pl}(E)$ и $\Psi_{nv}(E)$.

Авторы [7] моделировали на компьютере спектральную плотность для трехмерной модели Изинга и установили, что с хорошей точностью ее можно представить в виде

$$\begin{gathered} D_{pl}(E) \sim \exp\ N \cdot \Psi_{pl}(E)\ , \\ \Psi_{pl}(E) = \ln 2 - a \cdot E^2 + b \cdot E^4 - c \cdot E^6, \\ a = 0.1378966, \quad b = 0.01459613, \\ c = 0.0008365155, \quad E \in [-3, +3]. \end{gathered} \tag{4.1}$$

Такие значения коэффициентов получаются в результате оптимальной подгонки экспериментальной экспоненты четным полиномом шестой степени.

Обратимся теперь к методу n-окрестностей и получим экспоненту $\Psi_{nv}(E)$, отвечающую числу ближайших соседей $q = 6$. В этом случае решение задачи (3.3) дает следующее. Пока E по модулю не очень велико - пока $E / E_0 < 0.2113$ - минимум функции $\varphi(E, x)$ находится в точке $x_0 = 1/2$. Для таких энергий величина $\varphi_m(E)$ вычисляется по формуле (3.4):

$$\varphi_m(E) = \min_x \varphi(E, x) = -\ln 2 + \frac{E^2}{6}. \tag{4.2}$$

При $E \approx 0.21135 \cdot E_0$ точка x_0 становится точкой максимума функции $\varphi(E, x)$, а рядом с x_0 появляется точка минимума $x_m(E)$. Дальнейшее движение E к E_0 сопровождается стремлением точки минимума $x_m(E)$ к 0, а само значение минимума $\varphi_m(E)$ логарифмически стремится к 0. Вычислив $\varphi_m(E)$ для всех значений энергии $E \in [-3, 0]$ получим $\Psi_{nv}(E) = -\varphi_m(E)$ - аппроксимацию методом n-окрестностей неизвестной нам истинной экспоненты $\Psi(E)$.

В левом столбце Рис.6 показаны графики функций $\Psi_{pl}(E)$ и $\Psi_{nv}(E)$ (верхняя панель), их первых производных $\Psi'_{pl}(E)$ и $\Psi'_{nv}(E)$ (средняя панель), и вторых производных $\Psi''_{pl}(E)$ и $\Psi''_{nv}(E)$ (нижняя панель). Графики, связанные с полиномиальной аппроксимацией, показаны штрих-пунктирной линией, а относящиеся к функции $\Psi_{nv}(E)$ - штриховой линией.

Графики функций $\Psi_{pl}(E)$ и $\Psi_{nv}(E)$ весьма похожи друг на друга, однако имеются и заметные отличия. Полином $\Psi_{pl}(E)$ при $E = E_0$ отличен от нуля: $\Psi_{pl}(E_0) > 0$. Следовательно, $D_{pl}(E_0) = \exp\ N \cdot \Psi_{pl}(E_0)$ при $N >> 1$ принимает сколь угодно большое значение. Иными словами, в рамках полиномиального приближения основное состояние системы оказывается многократно вырожденным – что неверно. Значение экспоненты при $E = E_0$ должно равняться 0, и функция $\Psi_{nv}(E)$ этому требованию удовлетворяет (см. Приложение).

На средней левой панели приведены графики производных $\Psi'_{pl}(E)$ и $\Psi'_{nv}(E)$. Линейный ход $\Psi'_{nv}(E)$ на интервале $-0.63 < E < 0$ есть следствие квадратичной зависимости (4.2). Угловая точка на графике $\Psi'_{nv}(E)$ порождает скачок второй производной $\Psi''_{nv}(E)$ в этой точке (см. ниже). При $E \to E_0$ производная $\Psi'_{nv}(E)$ стремится к бесконечности.

Напротив, производная $\Psi'_{pl}(E_0)$ конечна, и из-за этого система попадает в основное состояние при конечной температуре (см. обсуждение выше). Еще один неприятный момент в поведении $\Psi'_{pl}(E)$ - ее немонотонность. На интервале $-2.1 < E < -1.5$ производная $\Psi'_{pl}(E)$ возрастает. Следовательно, вторая

производная на этом интервале будет положительной, что запрещено соотношениями (2.4). Для полиномиального приближения энергетический интервал $-2.1 < E < -1.5$ является запрещенным.

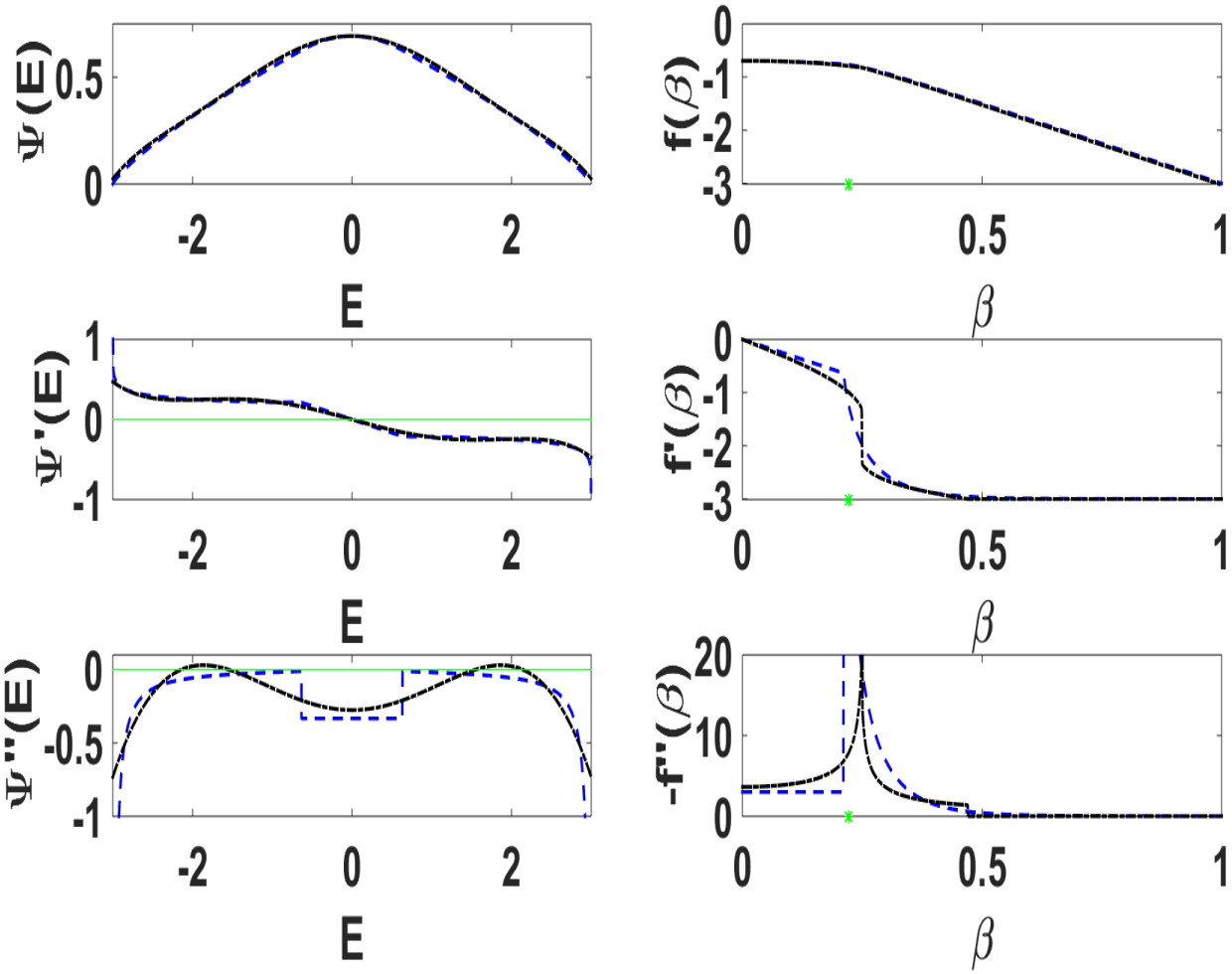


Рис. 6. Для трехмерной модели Изинга сопоставление полиномиальной аппроксимации экспоненты спектральной плотности и аппроксимации методом n-окрестностей. Левый столбец – полиномиальная аппроксимация $\Psi_{pl}(E)$ (4.1) (штрих-пунктир) и аппроксимация методом n-окрестностей $\Psi_{nv}(E)$ (штриховая), а также их производные. Правый столбец - свободная энергия $f_{pl}(\beta)$, отвечающая $\Psi_{pl}(E)$ (штрих-пунктир), и $f_{nv}(\beta)$, отвечающая $\Psi_{nv}(E)$ (штриховая) – см. текст.

На нижней левой панели показаны графики вторых производных. Вторая производная $\Psi''_{nv}(E)$ стремится к отрицательной бесконечности когда $E \to E_0$, а при $E_c \approx -0.63$ испытывает конечный скачок, который отвечает фазовому переходу второго рода – см. ниже. На интервале $[-0.63, 0]$ имеем: $\Psi''_{nv}(E) \equiv -1/3$. Напротив, график второй производной $\Psi''_{pl}(E)$ является гладкой кривой и никаких особенностей не содержит. Там, где производная $\Psi''_{pl}(E)$ положительна, возникает зона запрещенных энергий. Прыжок через запрещенную зону всегда связан с фазовым переходом первого рода, которого для трехмерной модели Изинга никто не предсказывает.

4.2. Поведение свободной энергии.

Обратимся к графикам свободной энергии,. На верхней правой панели Рис. 6 показаны графики свободной энергии, под ними – графики их первых производных, еще ниже – графики вторых производных. Штрих-пунктирная линия использована для $f_{pl}(\beta)$, отвечающей аппроксимации спектральной плотности полиномом. Штриховая линия – для $f_{nv}(\beta)$, отвечающей аппроксимации методом n-окрестностей. Маркером на оси абсцисс отмечено критическое значение обратной температуры $\beta_c \approx 0.2216$, полученное для трехмерной модели Изинга в численном эксперименте [10].

Как и раньше, графики $f_{nv}(\beta)$ и $f_{pl}(\beta)$ почти не различаются – см. верхнюю правую панель рисунка. Иное дело – их производные. График производной $f'_{nv}(\beta)$ не содержит никаких особенностей, имеет угловую точку при $\beta_{nv} \approx 0.2114$, а при $\beta \to \infty$ асимптотически стремится к энергии основного состояния $E_0 = -3$. Естественно, вторая производная $f''_{nv}(\beta)$ при $\beta = \beta_{nv}$ испытывает конечный скачок (см. штриховую линию на нижней панели рисунка). Иными словами, метод n-окрестностей предсказывает для трехмерной модели Изинга фазовый переход второго рода с конечным скачком теплоемкости. Значение критической температуры β_{nv} меньше чем на 5% отличается от оценки, полученной в компьютерном эксперименте. В целом, полученные результаты весьма близки к тем, что являются общепринятыми для трехмерной модели Изинга.

Иную картину дает полиномиальная аппроксимация спектральной плотности. На средней панели мы видим скачок первой производной $f'_{pl}(\beta)$ при $\beta_{pl} \approx 0.25$, означающий фазовый переход первого рода. А затем, при $\beta \approx 0.47$, происходит еще один фазовый переход – на этот раз второго рода, поскольку небольшой скачок имеется только у второй производной $f''_{pl}(\beta)$. Получается, что система при конечной температуре переходит в основное состояние. Можно констатировать, что поведение функции $f_{pl}(\beta)$ и ее производных нефизично.

5. Обсуждение и выводы

Исследован подход к вычислению свободной энергии, основанный на аппроксимации спектральной плотности. Детальные вычисления проделаны для модели Изинга на d-мерном гиперкубе для $d = 2$ (планарная модель) и $d = 3$ (кубическая модель). Число ближайших соседей у каждого спина $q = 2d$. Изучены два способа аппроксимации экспоненты $\Psi(E)$ спектральной плотности (2.1): полиномами четной степени, как это делалось в [7], и с помощью метода n-окрестностей [8], [9].

Для планарной модели ($q = 4$) оба подхода дают неверные результаты – в обоих случаях предсказывается фазовый переход первого рода, что противоречит классическому результату Онсагера [3]. Кроме того, полиномиальная аппроксимация приводит к тому, что система при конечной температуре оказывается в основном состоянии – что неприемлемо с физической точки зрения (пп. 2.3, 2.4).

Для кубической модели Изинга ($q = 6$) полиномиальная аппроксимация приводит к тем же самым неверным результатам. Напротив, метод n-окрестностей дает здесь совершенно иную картину: во-первых, предсказывается фазовый переход второго рода при температуре, которая меньше чем на 5% отличается от общепринятого значения; во-вторых, при $\beta \to \infty$ система стремится к основному состоянию асимптотически. Это согласуется с общепризнанными представлениями о ситуации для трехмерной модели Изинга.

Обсудим один принципиальный момент. В п. 2.1 показано, что когда энергия E стремится к энергии основного состояния E_0, экспонента $\Psi(E)$ истинной спектральной плотности стремится к 0, а ее производная стремится к ∞.

Если использовать полиномиальную аппроксимацию $P(E) = a_k E^k + a_{k-1} E^{k-1} + \ldots + a_0 \approx \Psi(E)$, то нетрудно удовлетворить требованию $P(E_0) = 0$. Однако никаким образом нельзя удовлетворить требованию $P'(E_0) = \infty$. Полиномиальная аппроксимация $P(E)$ экспоненты $\Psi(E)$ всегда дает конечную производную $P'(E_0) < \infty$, а это приводит к

тому, что система оказывается в основном состоянии при конечной температуре – см. рассуждения в конце п. 2.3.

Аппроксимация экспоненты $\Psi(E)$ методом n-окрестностей функцией гарантирует, что при $E \to E_0$ имеет место $\Psi'_{nv}(E) \to \infty$ (см. графики на средних панелях рисунков 6 и 8), а для $q > 4\ln 2$ выполняется и второе требование: $\lim\limits_{E \to E_0} V(E) = 0$ (см. Приложение). В области $q < 16/3$ метод n-окрестностей работает плохо – см. разделы 2 и 3. Мы надеемся расширить границы применимости метода, использовав для аппроксимации истинного распределения энергий в n-окрестности вместо гауссовой плотности более общий тип распределение, в котором учитывались бы не только первые два момента истинного распределения, но и более высокие моменты. Основанием для этих надежд является то, что положительный эффект дает даже небольшое обобщение гауссовой аппроксимации, когда с помощью разложения Эджворта [11] учитывается третий момент распределения в n-окрестности: для модели Изинга на решетках больших размерностей - $3 \le d \le 7$ - на 20% улучшается согласие получающихся результатов с компьютерным экспериментом (см. ссылки на литературу в [9]). Эти результаты готовятся к публикации.

В итоге можно сказать: полиномиальная аппроксимация спектральной плотности не годится для вычисления свободной энергии. Использование здесь метода n-окрестностей в его нынешней формулировке не применимо при $q < 4\ln 2$, но дает результаты, хорошо согласующиеся с общепринятыми, для больших значений q: $q > 16/3$. Мы надеемся усовершенствовать метод n-окрестностей и расширить границы его применимости.

Приложение

Пусть $\Psi_{nv}(E)$ - результат аппроксимация методом n-окрестностей экспоненты спектральной плотности для модели Изинга с координационным числом q.

Основное состояние модели Изинга является невырожденным – следовательно, должно выполняться $D_{nv}(E_0) = \exp\left[N \cdot \Psi_{nv}(E_0)\right] = 2$.

Ясно, что при $N \to \infty$ равенство $D_{nv}(E_0) = 2$ выполняется только когда $\Psi_{nv}(E) \to 0$ при $E \to E_0$. Выясним условия, при которых выполняется это условие.

Обратимся к выражениям (3.2), (3.3). Необходимо, чтобы решение $\varphi_m(E)$ задачи минимизации

$$\varphi_m(E) = \min_{0 \le x \le 1} \varphi(E, x),$$
$$\varphi(E, x) = S(x) + \frac{q}{4}\left(1 - \frac{1 - E/E_0}{4x(1-x)}\right)^2, \qquad (\Pi 1)$$

стремилось к 0, когда $E \to E_0$.

Минимум функции $\varphi(E, x)$ определяется равенством нулю ее производной по x, что дает уравнение:

$$z(x)\left[1 - z(x)\right] = \alpha(x), \text{ где}$$
$$\alpha(x) = \frac{2x(1-x)}{q(1-2x)} \ln\frac{1-x}{x}, \qquad (\Pi 2)$$
$$z(x) = 1 - \frac{1 - E/E_0}{4x(1-x)}.$$

Функция $\alpha(x)$ монотонно возрастает на интервале $[0, 0.5]$ от значения $\alpha(0) = 0$ до значения $\alpha(1/2) = 1/q$.

Для $q > 1$ при любом x имеем $\alpha(x) < 1$, и по $\alpha(x)$ можно вести разложение как по малому параметру. Разрешая (П2) относительно $z(x)$ получаем новое уравнение:

$$z_1(x) = \frac{1 - \sqrt{1 - 4 \cdot \alpha(x)}}{2} \approx \alpha(x).$$

Это уравнение связывает E/E_0 с тем значением координаты x_E, которое доставляет минимум функции $\varphi(E, x)$. Когда $E \to E_0$, точка минимума x_E стремится к 0: $x_E \to 0$. Следовательно, при $E \to E_0$ решение задачи (П1) стремится к 0:

$$\lim_{E \to E_0} \varphi_m(E) \approx \lim_{E \to E_0}\left(S(x_E) + \frac{q}{4}\alpha(x_E)\right) = 0.$$

С другой стороны, в точке $x_0 = 1/2$ всегда имеется тривиальный минимум, и его значение при $E \to E_0$ есть:

$$\lim_{E\to E_0}\varphi(E,x_0)=\lim_{E\to E_0}\left(S(x_0)+\frac{q}{4}\left(\frac{E}{E_0}\right)^2\right)=-\ln 2+\frac{q}{4}$$

Для того, чтобы $V_q(E)$ стремилось к 0 при $E\to E_0$, необходимо, чтобы $\lim_{E\to E_0}\varphi_m(E)$ был меньше $\lim_{E\to E_0}\varphi(E,x_0)$. Это дает:

$$q>4\ln 2\,. \qquad (\text{П3})$$

Полученное неравенство устанавливает границу значений параметра q, ниже которой применять метод n-окрестностей нельзя, поскольку это дает спектральную плотность с многократно вырожденным основным состоянием.

Публикация выполнена в рамках выполнения госзадания ГП 14 по проекту «35.14. Исследование топологии энергетической поверхности многоэкстремального квадратичного функционала в пространстве состояний с q-нарными переменными» (0065-2018-0001).

Spectral density for Ising model on hypercube

Boris Kryzhanovsky and Leonid Litinskii

Abstract. For the Ising model on a hypercubic lattice we examine two ways of approximation of a spectral density $D(E)=\exp(N\cdot\Psi(E))$ that describes a degeneracy of each value of the energy E. When using the first approach we approximate the exponent $\Psi(E)$ by polynomials of even degrees and in the framework of the second approach we apply the n-vicinity method we worked out previously. We found that the graphs of the free energy only slightly depend on the way of approximation. At the same time, the derivatives of the free energy whose behavior define phase transitions substantially depend on the way of approximation. In the case of the polynomial approximation the system always finds itself in the ground state at a finite temperature. This fact contradicts to the basic physical principles. The n-vicinity method provides correct approximation when the lattice dimension is large. In this case the obtained results are in good agreement with the known computer simulations. However, the n-vicinity method works poorly when the dimension of the system is small.

Key words: Ising model, free energy, spectral density, n-vicinity method.

Литература

[1] R.J. Baxter. Exactly solved models in statistical mechanics. London, Academic Press, 1982.
[2] R. Becker, W. Doring. Ferromagnetism. Berlin, Springer Verlag, 1939.
[3] L. Onsager. Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition. “Phys. Rev.”, v.65 (1944), 117-125.
[4] P. D. Beale. Exact distribution of energies in the two-dimensional Ising model. “Phys. Rev. Lett.”, v. 76 (1996), 78-81.
[5] R. Häggkvist, A. Rosengren, D. Andrén, P. Kundrotas, P.H. Lundow, K. Markström. Computation of the Ising partition function for 2-dimensional square grids. “Phys. Rev. E”, v. 69 (2004), No. 4, 046104.
[6] B.V. Kryzhanovsky. The Spectral Density of a Spin System Calculated for Solvable Models. arXiv: 1704.01351 (2017).
[7] J.M. Dixon, J.A. Tuszynski, E.J. Carpenter. Analytical Expressions for Energies, Degeneracies and Critical Temperatures of the 2D Square and 3D Cubic Ising Models. “Physica A”, v. 349, (2005), 487–510.
[8] B. Kryzhanovsky, L. Litinskii. Generalized Bragg-Williams Equation for System with an Arbitrary Long-Range Interaction. “Doklady Mathematics”, v. 90 (2014), No. 3, 784–787.
[9] B. Kryzhanovsky and L. Litinskii. Applicability of n-vicinity method for calculation of free energy of Ising model. “Physica A”, v. 468 (2017), pp. 493–507.
[10] R.Häggkvist, A. Rosengren, P.H. Lundow , K. Markström, D. Andren, P. Kundrotas. On the Ising model for the simple cubic lattice. “Advances in Physics”, v. 56 (2017), No.5, 653-755.
[11] H. Cramer. Mathematical Methods of Statistics, Princeton, Princeton University Press, 1999.

Изменения свойств модели Изинга и её термодинамических характеристик при добавлении шума в матричные элементы

М.Ю. Мальсагов [1], Я.М. Карандашев[2], Б.В. Крыжановский [3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]malsagov@niisi.ras.ru, [2]karandashev@niisi.ras.ru, [3]kryzhanov@mail.ru

Аннотация. Подсчёт вероятностей во многих задачах компьютерного зрения и машинного обучения сводится к подсчёту нормировочной константы (статистической суммы). В данной работе рассматривается двумерная модель Изинга с ближайшими соседями, когда взаимодействие между соседями имеет постоянное среднее значение с небольшим шумом. Проводятся численные эксперименты по вычислению нормировочной константы для случая, когда шум в матричных элементах плавно возрастает, приводится анализ полученных результатов и сравнение с решением Онсагера.

Ключевые слова: Нормировочная константа, двумерная решётка, взаимодействие между ближайшими соседями, критическая температура, зашумление матричных элементов.

1. Введение

Современные тенденции в области нейронных сетей направлены главным образом на методы обучения с учителем. Однако есть области, в которых мало или нет совсем размеченных данных. В таких случаях единственным вариантом является обучение без учителя. Задача обучения без учителя напрямую связана с уменьшением размерности и кластеризацией данных. Автоэнкодеры – это яркий пример такого подхода. К сожалению, интерес к ним сильно уменьшился с появлением быстрых методов оптимизации нейронных сетей прямого распространения. Такие сети формируют однонаправленную функцию, для которой обратную построить крайне трудно.

Одним из первых методов обучения глубоких автоэнкодеров, таких как restricted Boltzman machine, основан на максимизации функции правдоподобия генеративной нейронной сети. Задача вычисления градиента функции правдоподобия сводится к вычислению нормировочной константы, что осуществляется приближёнными методами, такими как contrastive divergence (CD) [1, 2].

В большинстве случаев нет возможности вычислить нормировочную константу точно. Фактически есть лишь ограниченное число моделей, когда её можно посчитать. В настоящей работе мы рассматриваем одну из таких моделей – это двумерная модель Изинга без внешнего поля. Двумерная модель Изинга является идеальным объектом исследования, во-первых, потому что плоская решётка может рассматриваться как набор пикселей изображения, а, во-вторых, из статфизики для неё известно точное аналитическое решение, полученное Онсагером [3] для случая одинаковых матричных элементов. Более того, для этой модели при помощи алгоритма Фишера-Кастелейна [4–7] удаётся получить численное решение для любой конечной размерности и при любых коэффициентах взаимодействия.

В настоящей статье мы посчитали свободную энергию для планарных графов, когда взаимодействие между соседями имеет постоянное среднее значение с небольшим шумом. Было обнаружено, что с ростом дисперсии матричных элементов пик второй производной свободной энергии, всё дальше и дальше удаляется от предсказанного Онсагером по мере увеличения разброса матричных элементов, до тех пор, пока совсем не исчезает.

Анализ поведения теплоёмкости при небольших значениях дисперсии шума показал, что сдвиг критической температуры происходит линейно с ростом дисперсии. Вычисляя свободную энергию методом n-окрестностей [8], нам удалось получить

приближённое описание отклонения критической температуры с ростом дисперсии шума, хорошо совпадающее с данными эксперимента.

2. Постановка задачи

Будем пользоваться физической терминологией. Пусть задана система N спинов бинарных спинов, описываемая гамильтонианом:

$$E = -\frac{1}{2N}\sum_{i,j=1}^{N} J_{ij} s_i s_j \,, \quad s_i = \pm 1 \,. \tag{1}$$

Данный функционал довольно часто встречается в задачах машинного обучения и задачах обработки изображений. Величины $s_i = \pm 1$ имеют смысл принадлежности к одному из двух классов (фон/объект) пикселей на картинке, либо являются активностью нейронов в байесовой нейронной сети. При заданной температуре, вероятность того, что система находится в состоянии s, описывается формулой:

$$P(s) = \frac{1}{Z} e^{-\beta E(s)}, \tag{2}$$

где β - обратная температура, а $Z = \sum_s e^{-N\beta E(s)}$ - нормировочная константа (статистическая сумма) есть сумма по всем возможным конфигурациям s. Как видно из формулы (2), для оценки вероятности конфигурации необходимо знать нормировочную константу, которую оценить не так-то просто. Традиционно вместо самой константы Z обычно ищут удельную свободную энергию системы:

$$f(\beta) = -\frac{\ln Z}{N} \,. \tag{3}$$

Знание свободной энергии в том числе позволяет вычислять основные измеряемые физические параметры системы:

$$\begin{gathered} U = \frac{\partial f}{\partial \beta} \\ \sigma^2 = -\frac{\partial^2 f}{\partial \beta^2}, \\ C = -\beta^2 \frac{\partial^2 f}{\partial \beta^2} \end{gathered} \tag{4}$$

где внутренняя энергия $U = \overline{E}$ есть среднее по ансамблю при заданном β, $\sigma^2 = \overline{E^2} - \overline{E}^2$ - дисперсия энергии, а $C = \beta^2 \sigma^2$ - удельная теплоемкость. Отметим здесь, что дисперсия и теплоёмкость отличаются лишь множителем β^2, при этом в физике обычно пользуются понятием теплоемкости.

В настоящей работе, мы исследуем планарную модель, в которой спины расположены на квадратной решетке, а взаимодействие имеется только с ближайшими четырьмя соседями.

Если ограничиться случаем, когда взаимодействие с ближайшими соседями задается константой $J_{ij} = J = 1$, то для него известно решение Онсагера [3], полученное в пределе $N \to \infty$ с периодическими граничными условиями, которое имеет вид:

$$f(\beta) = -\frac{\ln 2}{2} - \ln\left(\cosh 2\beta J\right) - \frac{1}{2\pi}\int_0^{\pi} \ln\left(1 + \sqrt{1 - k^2 \cos^2\theta}\right) d\theta \,, \tag{5}$$

$$U = -\coth 2\beta \cdot \left[1 + \frac{2}{\pi} \mathrm{E}_1(k)\left(2\tanh^2 2\beta - 1\right)\right], \tag{6}$$

$$C = \frac{4\beta^2}{\pi \tanh^2 2\beta} \cdot \left\{ \mathrm{E}_1(k) - \mathrm{E}_2(k) - \left(1 - \tanh^2 2\beta\right)\left[\frac{\pi}{2} + \left(2\tanh^2 2\beta - 1\right)\mathrm{E}_1(k)\right]\right\}, \tag{7}$$

где

$$k = \frac{2\sinh 2\beta}{\cosh^2 2\beta} \,, \tag{8}$$

а $\mathrm{E}_1 = \mathrm{E}_1(k)$ и $\mathrm{E}_2 = \mathrm{E}_2(k)$ - полные эллиптические интегралы первого и второго рода соответственно:

$$\mathrm{E}_1(x) = \int_0^{\pi/2} (1 - x^2 \sin^2\phi)^{-1/2}\, d\phi, \quad \mathrm{E}_2(x) = \int_0^{\pi/2} (1 - x^2 \sin^2\phi)^{1/2}\, d\phi \,. \tag{9}$$

Решение Онсагера (5-7) даёт асимптотическое поведение при бесконечных размерах решётки. Из этих формул следует логарифмическая расходимость удельной теплоёмкости при некоторой критической температуре:

$$\beta_{\mathrm{ONS}} = \frac{1}{2}\ln\left(1 + \sqrt{2}\right) \approx 0.4407 \,. \tag{10}$$

Как было показано в работе [9] для решёток конечной размерности логарифмической расходимости не наблюдается. Вместо этого, присутствует пик дисперсии при температуре близкой к β_{ONS}, причём высота пика растёт логарифмически с ростом размерности решётки.

Можно предположить, что логарифмический рост с увеличением размерности, а также логарифмическая расходимость при бесконечной размерности, являются следствием того, что все матричные элементы в решётке J_{ij} равны между собой и таким образом возникает нечто вроде когерентности, приводящей к неограниченному росту удельной теплоёмкости с увеличением размера решётки. Чтобы избавиться от данной когерентности, в настоящей работе мы исследуем случай, когда коэффициенты взаимодействия имеют то же среднее значение, равное 1, но с небольшим шумом:

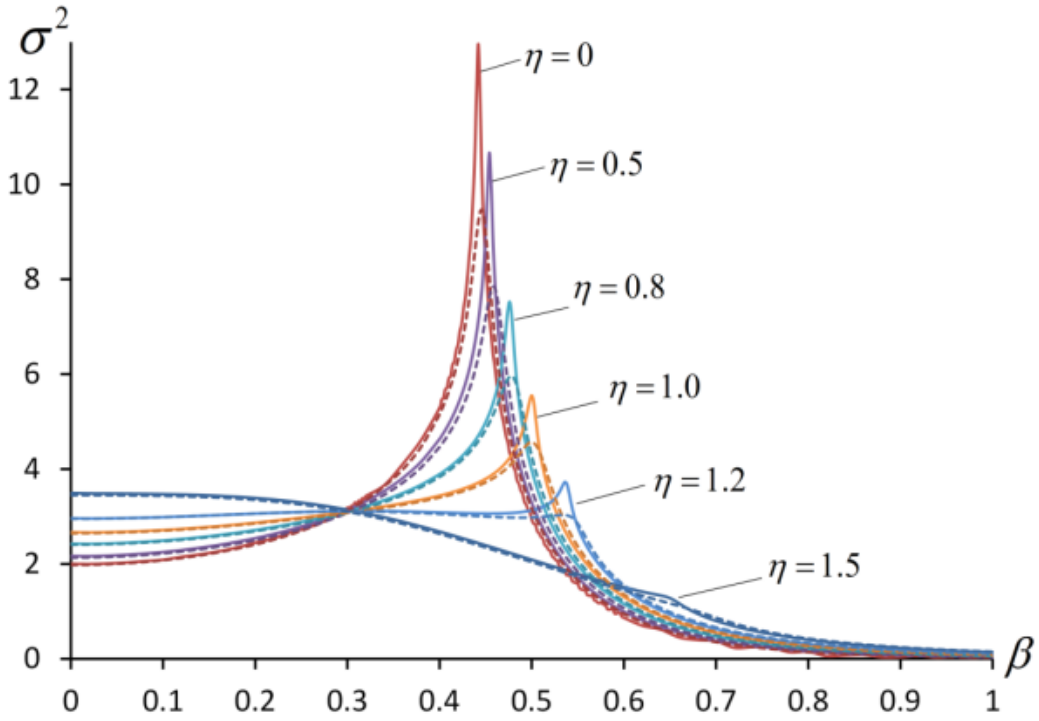


Рис. 1. Зависимость дисперсии энергии σ^2 от обратной температуры β для разных значений амплитуды шума η . Пунктирные и сплошные линии получены для $L = 100$ и $L = 400$ соответственно.

$$J_{ij} = 1 + \varepsilon, \qquad \varepsilon \in (-\eta, \eta), \qquad (11)$$

ε - аддитивный и равномерный на интервале шум, максимальная амплитуда которого равна параметру η .

3. Экспериментальные результаты

Для проведения экспериментов мы использовали алгоритм Кастелейна-Фишера [4–7], который позволяет с машинной точностью вычислять величину свободной энергии для произвольной планарной модели. При помощи этого алгоритма нам удалось проанализировать поведение свободной энергии и её производных (внутренняя энергия U и теплоёмкость C) для ряда решёток различной размерности $N = L \times L$. Подчеркнем, что используемый нами алгоритм применим только к планарным решеткам. Это означает, что мы рассматривали только случай решеток со свободными граничными условиями, поскольку решетка с периодическими граничными условиями,

также как и двумерная модель с внешним полем, не является планарным графом.

При помощи данного алгоритма мы провели ряд экспериментов с решётками, линейный размер которых варьировался от $L=100$ до $L=500$ и для различных значений амплитуды шума η от 0 до 1.5.

Поскольку нас больше всего интересовало значение критической температуры, то наиболее показательным является поведение не самой свободной энергии, а её второй производной, т.е. σ^2. На Рис. 1 показаны результаты проведённых численных экспериментов.

Как видно из Рис. 1, положение пика дисперсии плавно сдвигается вправо, в сторону больших значений β, по мере того, как растёт амплитуда шума η. При этом высота пика плавно снижается и при $\eta=1.5$ пик становится еле заметным (при $L=400$), а при малых размерах решётки (при $L<100$) и вовсе исчезает.

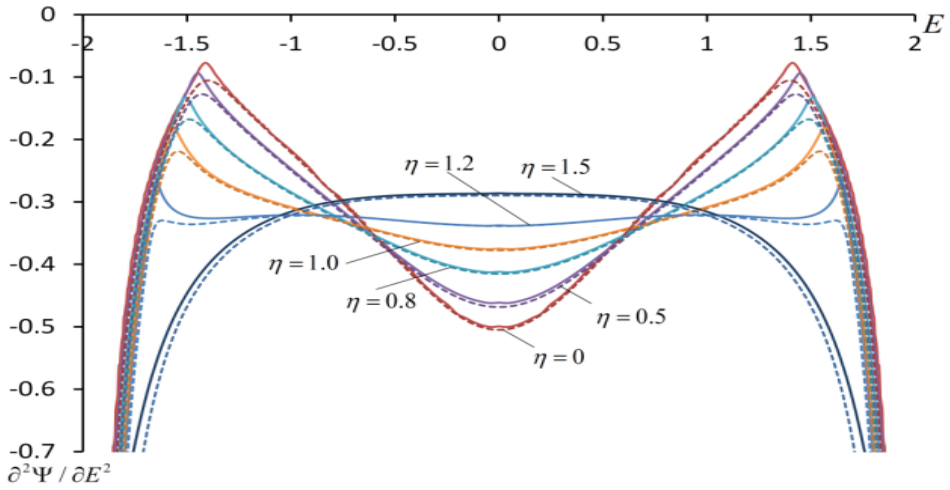


Рис. 2. Зависимость второй производной плотности распределения от энергии E для разных значений амплитуды шума η. Пунктирные и сплошные линии получены для $L=100$ и $L=400$ соответственно.

Интересно отметить на Рис. 1 точку вблизи $\beta\approx 0.3$, где все кривые пересекаются в одном месте. Мы не знаем, имеет ли данное наблюдение какой-то физический смысл, но оно имеет место лишь при небольших значениях шума ($\eta<1.5$), а при большом шуме область пересечения графиков размывается.

Из Рис. 1 можно сделать вывод, что начиная с некоторого значения амплитуды шума, пик вовсе пропадает. Вероятно вместе с пиком пропадёт и фазовый переход, который характерен для двумерной модели Изинга.

Помимо зависимости от температуры термодинамических характеристик, мы исследовали также как с добавлением шума изменяется спектральные характеристики системы. Пусть вероятность состояния с заданной энергией определяется выражением:

$$P(E)=\frac{e^{N\Psi(E)}}{2^N}, \qquad (12)$$

где $\Psi(E)$ - спектральная плотность распределения состояний по энергиям. Легко показать [10], что спектральная плотность $\Psi(E)$ связана со свободной энергией и её производными следующими неявными закономерностями:

$$E=\frac{df(x)}{dx}, \quad \Psi(E)=xE-f(x), \qquad (13)$$

где $f(x)$ получается из функции $f(\beta)$ заменой β на x. Здесь мы ввели некую произвольную переменную x, чтобы подчеркнуть, что спектральная плотность не связана с понятием температуры. Изменяя x от 0 до ∞ мы прописываем весь спектр энергий: для каждого значения x выражениями (12) находится энергия E и соответствующее значение $\Psi(E)$. Первая и вторая производные спектральной плотности энергии также выражаются неявными формулами:

$$\partial\Psi/\partial E=-x\ , \qquad (14)$$

$$\partial^2\Psi/\partial E^2=-\sigma^{-2}(x)\ . \qquad (15)$$

График самой спектральной плотности является малоинформативным, поскольку

слабо зависит от шума матричных элементов. Более информативным является график второй производной $\Psi(E)$ по энергии, показанный на Рис. 2. Понятно, что чем выше пик дисперсии σ^2, тем ближе $\partial^2\Psi/\partial E^2$ приближается к нулю. В асимптотическом случае $N\to\infty$, дисперсия в критической точке $\sigma_c^2\to\infty$ и значит, кривая $\partial^2\Psi/\partial E^2$ вплотную подходит к нулевому значению. При конечных размерностях, значение дисперсии всегда конечно, поэтому и $\partial^2\Psi/\partial E^2$ проходит на некотором конечном расстоянии от нуля. С добавлением шума происходит плавное изменение структуры спектра, что отчётливо видно на графике Рис. 2. При $\eta<1.2$ явно проглядывается прогиб в центре, а также пики по бокам. При $\eta>1.5$ этого нет, что вероятно может означать качественную перестройку системы.

4. Аналитика

Рассмотрим зависимость критической температуры от дисперсии матричных элементов. Решение Онсагера не содержит такой зависимости, поэтому воспользуемся приближением, основанным на так называемом «методе n-окрестностей» [8]. Приведенное в [8] уравнение состояния после ряда упрощений принимает вид:

$$\ln\frac{1+m}{1-m}=8mb\left[\frac{2-b}{2+\sigma_\eta^2}-b\left(\frac{1}{2}-m^2\right)\right], \quad (16)$$

где переменная m - это намагниченность, σ_η^2 - дисперсия шума матричных элементов, а b - безразмерный параметр, связанный с β соотношением:

$$\beta=\frac{\pi b}{\sqrt{2}\,(2+\sigma_\eta^2)}. \quad (17)$$

Отыскание критической точки сводится к численному решению (16) при заданных значениях параметров b и σ_η^2. При малых значениях b уравнение (16) имеет единственное решение $m=0$. При значениях b, больших некоторого критического значения b_c, появляется еще одно решение при $m\neq 0$. Подстановка b_c в (17) дает критическое значение обратной температуры $\beta=\beta_c$, при которой происходит фазовый переход.

В таблице 1 приведены величины β_c при разных значениях σ_η^2: в первой строке приведены экспериментальные значения β_c, а в второй – значения, полученные численным решением уравнения (16).

Как видим, теоретические и экспериментальные значения отличаются менее чем на 0.5%, что соответствует точности эксперимента. Нетрудно заметить, что на приведенном интервале изменений дисперсии σ_η^2 зависимость $\beta_c=\beta_c(\sigma)$ с большой точностью аппроксимируется выражением:

$$\beta_c=\beta_{\mathrm{ONS}}\left(1+\frac{1}{2}\sigma_\eta^2\right), \quad (18)$$

где β_{ONS} - полученное Онсагером критическое значение (10). Графики зависимости $\beta_c=\beta_c(\sigma_\eta)$ приведены на Рис. 3. Как видим, критическое значение β_c возрастает практически линейно по σ_η^2.

η	0.0	0.2	0.5	0.8	1.0	1.2
$\sigma_\eta^2=\eta^2/3$	0.000	0.013	0.083	0.213	0.333	0.480
β_c эксперимент	0.442	0.444	0.454	0.476	0.500	0.536
β_c теория	0.440	0.442	0.452	0.477	0.501	0.534

Таблица 1. Зависимость β_c от величины σ_η^2: эксперимент и теория.

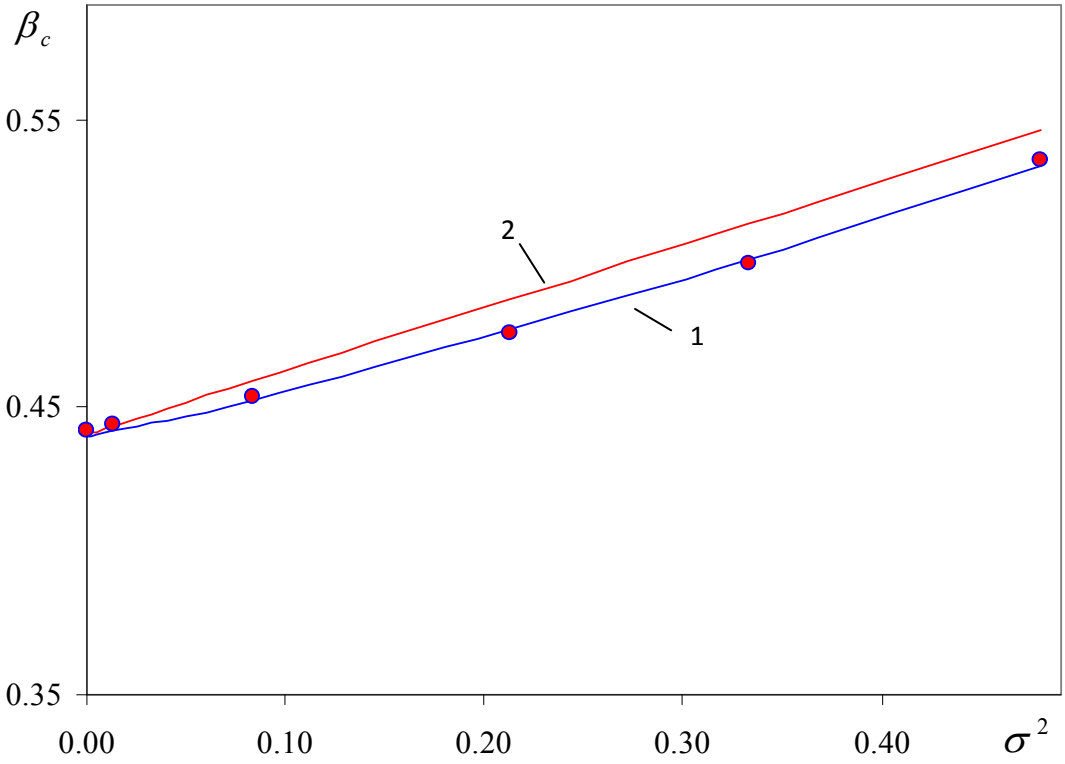


Рис. 3. Зависимость $\beta_c = \beta_c(\sigma_\eta)$: ромбы – экспериментальные точки; кривая 1 – теория; кривая 2 – приближённое выражение (18) .

5. Заключение

Настоящая работа несёт двоякую цель.

Во-первых, нам было интересно, как изменяется система и её термодинамические характеристики при добавлении шума в матричные элементы. Известная модель Онсагера, в которой все элементы строго равны 1, кажется далёкой от практики. Мы показали, что небольшое добавление шума не приводит к резким изменениям, а влечёт плавное изменение термодинамических характеристик, таких как теплоёмкость, а также спектральной плотности. При этом начиная с некоторого значения амплитуды шума вероятно происходит качественная перестройка системы, возможно даже исчезновение фазового сдвига. Впрочем, понять наличие или отсутствие фазового сдвига в системах конечной размерности - не такая простая задача.

Во-вторых, мы проверили разработанную прежде методику n-окрестностей. Приближённый аналитический метод n-окрестностей показал отличное качественное и количественное согласие с численным экспериментом, что даёт основания надеяться на дальнейшее его развитие и применение к задачам машинного обучения и обработки изображений. Главной задачей в этом направлении является добавление в модель внешнего поля.

Публикация выполнена в рамках госзадания по проведению фундаментальных научных исследований (ГП 14) по проекту «35.14. Исследование топологии энергетической поверхности многоэкстремального квадратичного функционала в пространстве состояний с q-нарными переменными» (0065-2018-0001).

Changes in the Ising model properties and its thermodynamic characteristics with noised matrix elements

M.Yu. Malsagov, I.M. Karandashev, B.V. Kryzhanovsky

Abstract. The calculation of probabilities in many problems of computer vision and machine learning is reduced to the finding of a normalizing constant (statistical sum). In the paper a two-dimensional nearest-neighboring Ising model with almost constant average interaction between neighbors is considered. We carry out numerical experiments to compute the normalizing constant for the case in which the noise in grid elements grows smoothly, analyze the results and compare them with Onsager's solution.

Keywords. The normalizing constant, two-dimensional grid, nearest-neighbors interaction, critical temperature, noise of grid elements.

Литература

1. A.Krizhevsky and G.E.Hinton. Using Very Deep Autoencoders for Content-Based Image Retrieval. European Symposium on Artificial Neural Networks ESANN-2011, Bruges, Belgium
2. G.E.Hinton and R.R.Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, Vol. 313. no. 5786, pp. 504 – 507 (2006).
3. L. Onsager. Crystal statistics. I. A two-dimensional model with an order–disorder transition. Phys. Rev. 65 (3–4), 117–149 (1944).
4. P. Kasteleyn. Dimer statistics and phase transitions. J.Math. Phys., 4(2), 1963.
5. M. Fisher. On the dimer solution of planar Ising models. J. Math. Phys., 7(11), 1966.
6. Ya. M. Karandashev, M. Yu. Malsagov. Polynomial algorithm for exact calculation of partition function for binary spin model on planar graphs. Optical Memory & Neural Networks (Information optics), v. 26, #2, 2017. https://arxiv.org/abs/1611.00922
7. N. Schraudolph and D. Kamenetsky. Efficient exact inference in planar Ising models. In NIPS, 2008. https://arxiv.org/abs/0810.4401
8. B.Kryzhanovsky, L.Litinskii. Generalized Bragg-Williams Equation for System with an Arbitrary Long-Range Interaction. Doklady Mathematics, Vol. 90, No. 3, pp. 784–787 (2014).
9. I.M. Karandashev, B.V. Kryzhanovsky, and M.Yu. Malsagov. The Analytical Expressions for a Finite-Size 2D Ising Model. Optical Memory and Neural Networks (Information Optics). 2017, v. 26(3), pp. 165–171.
10. B.V. Kryzhanovsky. The Spectral Density of a Spin System Calculated for Solvable Models. http://arxiv.org/abs/1704.01351

Воздействие тепловой обратной связи на температуру кристалла микросхемы

А.Г. Мадера

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: alexmadera@mail.ru

Аннотация. Потребляемая микросхемой мощность преобразуется в тепло, которое приводит к нагреванию микросхемы. В силу значительной зависимости электрических параметров микросхемы от температуры ее мощность потребления подвергается изменению, которое, в свою очередь, изменяет температуру и, как следствие, мощность потребления. Взаимно-обусловленное взаимодействие и влияние параметров электрического и теплового режимов друг на друга, возникающее в процессе работы микросхемы, называется тепловой обратной связью. Тепловая обратная связь присуща любым электронным полупроводниковым устройствам и неустранима. Воздействие тепловой обратной связи может как положительным, так и отрицательным. При положительной тепловой обратной связи потребляемая мощность и температура микросхемы увеличиваются, а при отрицательной уменьшаются. Результаты расчетов показывают, что адекватное моделирование электрических и тепловых параметров микросхем невозможно без учета воздействия тепловой обратной связи, и пренебрежение ею приводит к значительным погрешностям проектирования микросхем и созданию неконкурентоспособных электронных систем.

Ключевые слова: микросхема, тепловая обратная связь, саморазогрев, мощность потребления, температура

Потребляемая интегральной микросхемой (МС) мощность в значительной своей части (92 – 97%), рассеивается в виде тепла [1, 2, 3] и лишь оставшаяся часть (3 – 8%) преобразуется в полезные электрические сигналы. Возникающее при этом тепло приводит к нагреванию МС и возникновению распределения температуры как в самой МС, так и электронной системе (ЭС), в которой она функционирует [4]. Физические параметры полупроводникового материла – подвижность носителей, удельная электропроводность, ширина запрещенной зоны и пр., – на котором создается активная схема кристалла МС, характеризуются сильной зависимостью от температуры [3, 5]. В силу того, что физические параметры полупроводникового кристалла определяют значения электрических параметров МС, – быстродействие, помехоустойчивость, пороговые напряжения и т.д., – последние также сильно зависеть от температуры, приводя к взаимообусловленности и взаимовлиянию параметров электрического и теплового режимов в МС и ЭС в целом.

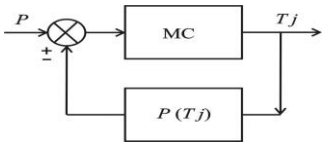


Рис. 1. Принцип тепловой обратной связи. $P(T_j)$ – мощность потребления МС, зависящая от температуры T_j кристалла МС

Действительно, потребляемая МС мощность разогревает ее и, в силу температурной зависимости электрических параметров МС, обусловливает изменение как самих электрических параметров, так и зависящей от них мощности электропотребления МС. Поскольку изменение мощности потребления ведет к изменению величины потоков тепла и температуры в МС, то последние, воздействуя на электрические параметры МС, снова приводят к изменению мощности электропотребления. Процесс взаимно-обусловленного между собой влияния параметров электрического и теплового режимов в МС и построенных на МС ЭС, носит название тепловой обратной связи (ТОС) или саморазогрева [3, 5, 6]. Явление ТОС присуще любым электронным устройствам, коль скоро используют полупроводниковые материалы, и неустранимо, однако, может быть уменьшено за счет принятия специальных мер, наиболее эффективными из которых являются, во-первых, разработка новых технологий изготовления МС менее чувствительных к температуре и, во-вторых, снижение температурных уровней вследствие применения более эффективных средств теплоотвода и охлаждения.

Воздействие ТОС в МС может как положительным, так и отрицательным (рис. 1). При положительной ТОС величина потребляемой мощности и уровни температуры в МС увеличиваются лавинообразно, что зачастую является причиной сбоев, выгорания МС, а также выхода параметров электрического и теплового режимов за пределы допустимых значений или достижения равновесных режимов при более высоких уровнях температур, не оставляющих запаса сопротивления воздействию дестабилизирующих факторам. Отрицательная обратная связь характеризуется снижением мощности энергопотребления и температуры

МС, обусловливая достижение устойчивого установившегося теплового и электрического режимов при пониженных уровнях температуры. Таким образом, положительная ТОС приводит к аномально высоким уровням температуры кристалла МС, что делает невозможным

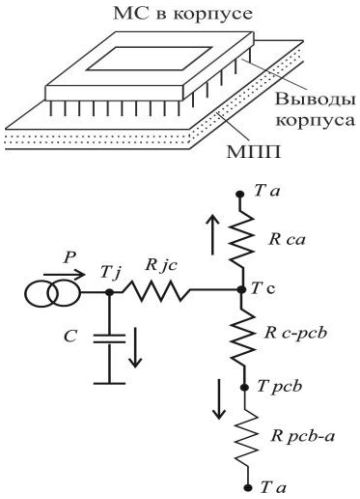


Рис. 2. МС в корпусе, установленная на МПП и ее тепловая модель. T_a – температура среды, T_j – температура кристалла МС

их функционирование, а отрицательная ТОС – к достижению устойчивого электрического и теплового режимов, поэтому при создании конкурентоспособных МС отрицательная ТОС представляет собой полезное явление, в то время как положительная ТОС – вредное.

Несмотря на существенное влияние ТОС на электрические и тепловые параметры полупроводниковых электронных устройств, расчеты электрических и тепловых режимов МС и ЭС, в настоящее время, проводятся, в основном, в предположении, согласно которому температура и мощность тепловыделения независимы и не связаны между собой. Указанное обстоятельство в значительной мере обусловливается сложностью математического моделирования ТОС, выражающимся также в недостаточном количестве литературы, посвященной проблеме ТОС.В данной работе на основании тепловой и математической моделей МС рассматривается эффект воздействия ТОС и проводится сравнение температур кристалла МС как с учетом, так и без учета ТОС.

Рассмотрим конструкцию ЭС (рис. 2), включающую в себя МС в корпусе, установленном на многослойной печатной плате (МПП), и ее тепловую модель. В процессе теплообмена в ЭС, потребляемая кристаллом МС мощность электропитания (P), трансформируется в тепло, которое распространяется по конструкции корпуса (тепловое сопротивление R_{jc}), затем частично поступает с поверхности корпуса МС в среду (тепловое сопротивление R_{ca}) и частично – через выводы корпуса и зазор между корпусом и платой (тепловое сопротивление R_{c-pcb}) – в окружающую среду с температурой T_a (тепловое сопротивление R_{pcb-a}).

Математическая модель теплообмена в конструкции ЭС имеет вид:

$$C\frac{dT_j}{dt}+\frac{T_j-T_a}{R_{ja}}=P(T_j),$$

где $R_{ja}=R_{jc}+R_{ca}||(R_{c-pcb}+R_{pcb-a})$.

Примем для простоты анализа и не умаляя при этом общности, что температурная зависимость мощности потребления МС является

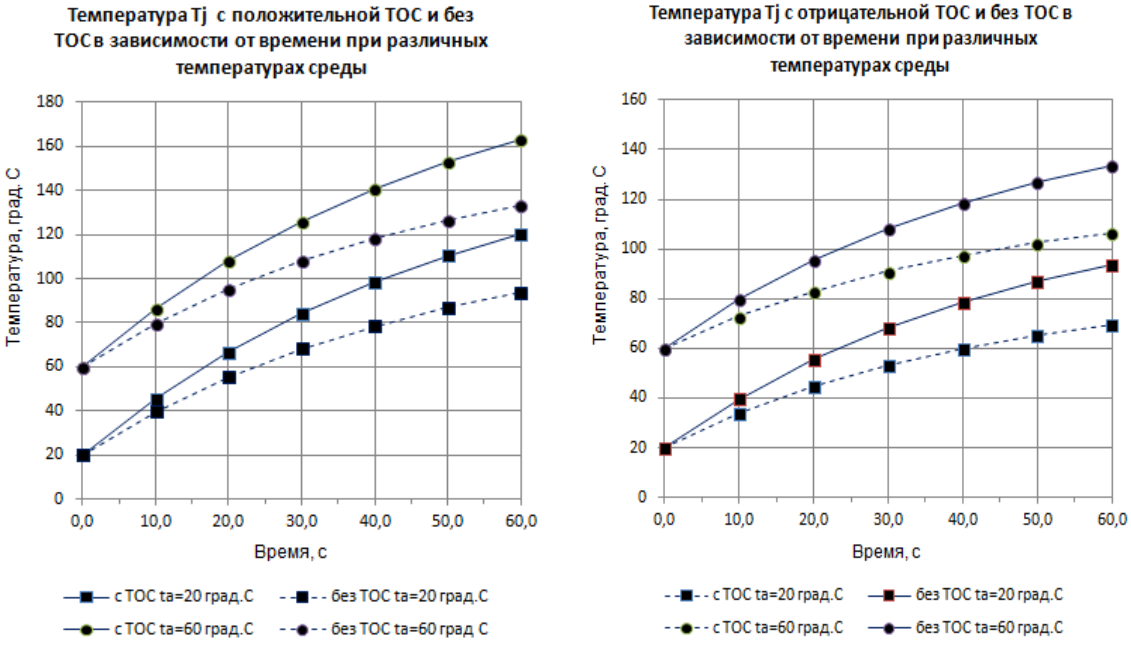


Рис. 3. Температура кристалла МС при положительной и отрицательной ТОС и в отсутствии ТОС

линейной функцией $P(T_j) = P_0 + kT_j$. Тогда решением уравнения математической модели будет

$$T_j(t) = \frac{P_0 + g_{ja}T_a}{g_{ja} - k} - \frac{P_0 + kT_a}{g_{ja} - k} e^{-\frac{g_{ja}-k}{C}t},$$
$$T_j(0) = T_a.$$

Результаты расчетов температуры кристалла $T_j(t)$ приведены для двух значений температуры среды $T_a = 20$ и $60°C$ как для положительной, так и отрицательной ТОС (рис. 3). Для сравнения приведены также результаты расчетов в случае пренебрежения воздействием ТОС. Анализ результатов показывает следующее:

– температура кристалла МС при положительной ТОС значительно, более чем на 35ºС, превышает температуру кристалла МС при отрицательной ТОС, причем это превышение с течением времени будет только возрастать;

– в тех случаях, когда влияние эффекта ТОС не учитывается (пунктирные кривые на каждом графике) то погрешность расчета температуры кристалла МС может превышать 30ºС.

Таким образом, результаты расчетов температуры кристалла МС как с учетом, так и без учета ТОС, позволяют сделать вывод, что адекватное моделирование электрического и теплового режимов МС может быть достигнуто только при учете ТОС, в то время как пренебрежение эффектом ТОС может приводить к значительным погрешностям при проектировании МС и построенных на ЭС различного назначения.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП14) по теме (проекту) «36.18. Исследование и разработка моделей и методов математического и компьютерного моделирования тепловых процессов в электронных системах различного назначения в условиях неопределенности ». (0065-2018-0003).

Thermal feedback effect on microchip temperature

A.G. Madera

Abstract. The power consumed by the microchip is converted into heat, which leads to the heating of the microcircuit. Due to the significant dependence of the electrical parameters of the semiconductor chip on temperature, its power consumption is subject to a change, which in turn changes the temperature and, as a consequence, the power consumption. The mutually conditioned interaction and the influence of the electric and thermal parameters on each other, arising during the operation of the microcircuit, is called thermal feedback. Thermal feedback is inherent in any electronic semiconductor devices and is not removable. The thermal feedback effect can be either positive or negative. At the positive thermal feedback, the power consumption and temperature of the chip increase, and with a negative decrease. The results of the calculations show that adequate simulation of the electrical and thermal parameters of microcircuits is impossible without taking into account the effect of thermal feedback, and neglecting it leads to significant errors in the design of microchips and the creation of noncompetitive electronic systems.

Keywords. Microcircuit, chip, thermal feedback, self-heating, power consumption, temperature.

Литература

1. С.Г.Бобков, А.Г.Мадера. Энергетические затраты, быстродействие и проблема теплоотвода в микропроцессорах // Программные продукты и системы. 2013. № 4. С.29 – 35
2. А.Г.Мадера. Моделирование теплообмена в технических системах. – М.: Науч. фонд «Первая исслед. Лаб. им. акад. В.А. Мельникова», 2005
3. В.А.Сергеев, А.М.Хадаков. Нелинейные тепловые модели полупроводниковых приборов. – Ульяновск: УлГТУ, 2012
4. А.Г.Мадера, П.И.Кандалов. Моделирование трехмерных температурных полей в электронных модулях // Программные продукты и системы. 2010. №2. С. 36
5. А.Г.Мадера. Расчет динамического режима полупроводниковых микросхем с учетом тепловой обратной связи // Микроэлектроника. 1982. Т. 11. № 2. С. 175 – 177.
6. B.M.Tenbroek, M.S.Lee, W.Redman-White, R.J.T.Bunyan, M.J.Uren. Self-heating effects in SOI MOSFET's and their measurement by small signal conductance techniques // IEEE Trans. On Electron Devices. 1996. Vol. 43. N. 12. P. 2240 – 2248

Особенности аномальной подвижности в полностью обедненных КНИ транзисторах

Н. В. Масальский

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: volkov@niisi.ras.ru

Аннотация. Обсуждаются вопросы моделирования аномальной подвижности, которая наблюдается в полностью обедненном полевом транзисторе со структурой «кремний на изоляторе» с толстым слоем подзатворного окисла фронтального затвора. Найдено, что аномальное поведение связано с доминированием заряда в канале у обратной поверхности в общем заряде. Это поведение описывается моделью с двумя подвижностями, которые соответствуют фронтальному каналу и каналу вдоль обратной поверхности. Эти две подвижности физически связаны при помощи взвешенной функции следующей из приближения зарядового разделения, и она экстрагируется из экспериментальных данных.

Ключевые слова: полностью обедненный полевой транзистор, структура кремний на изоляторе, подвижность, поверхностный потенциал

Введение

Полностью обедненные КНИ КМОП транзисторы находят широкое применение в микросхемах для приложений с низким уровнем питанием.

Данный класс транзисторов характеризуется прогнозируемым поведением коротко-канальных эффектов, удобной с точки зрения конструкции возможностью их подавления, использование низколегированной рабочей области положительным образом влияет на подвижность носителей и снижает roll-off порогового напряжения [1-2].

Пороговое напряжение полностью обедненного КНИ КМОП транзистора может быть скорректировано при помощи напряжения обратного затвора (Ub). Для других транзисторных структур это достигается путем варьирования концентрации легирования рабочей области, что, например, характерно для квази планарных устройств [3].

Кроме того, из-за высокого смещения обратного затвора, канал инверсии может сформироваться у обратной поверхности, что приводит к сложной физики подвижности, и, как следствие, к изменению токовых характеристик [4].

Например, в [5] сообщается, что подвижность на фронтальной поверхности и на обратной поверхности отличаются из-за удаленного кулоновского рассеивания и наличия заряженных центров в затворе поликристаллического кремния.

Кроме того, свойства этих поверхностей отличаются из-за разных топологических параметров затворов. Смещение обратного затвора также влияет пространственное распределение заряда, приводящее к значительному отклонению от обычного поведения подвижности.

На самом деле определение эффективного электрического поля (для учета деградации подвижности) неоднозначно в случае полностью обедненных КНИ КМОП транзисторов при условии инверсии обратной поверхности [6]. В результате из-за сложных эффектов поведения заряда и мобильности, вследствие изменения напряжения на обратном затворе, поведение активной проводимости полностью обедненного КНИ КМОП транзистора значительно отличается от объемного МОП транзистора [7, 8].

В данной статье рассматривается механизм аномальной подвижности в полностью обедненных КНИ транзисторах с утолщенным подзатворным окислом фронтального затвора. В свою очередь этот эффект определяет аномальное поведение активной проводимости в рассматриваемых транзисторных структурах.

Как известно, полностью обедненные транзисторы обычно используется для схем с низкой мощностью, для того чтобы выполнить высоковольтные требования, в настоящее время тенденции к применению более толстых диэлектриков показывает перспективность [9].

Толстый окисел фронтального затвора также используется в проектах устройств обработки изображений на полностью обедненных КНИ транзисторах, к которых плотность шума снижается посредством толстых окислов фронтального затвора [10-12]. В таких проектах, новый подход требуется, чтобы достоверно описать усиление аналогового сигнала в сигнальном процессоре, который в дальнейшем будет

обработан АЦП. Таким образом, необходимо разработать модель подвижности для полностью обедненных КНИ транзисторов с толстым подзатворным диэлектриком фронтального затвора.

В данной модели учитывается механизм аномальной подвижности. При помощи двух компонент - фронтальной компоненты и обратной компоненты. Они связаны с фронтальной и обратной поверхностями, соответственно.

Общая эффективная подвижность представляет собой их сумму с соответствующими весовыми коэффициентами.

1. Модель аномальной подвижности

Одно из преимуществ полностью обедненного КНИ КМОП нанотранзистора (его схема приведена на Рис. 1) – возможность регулировки значения порогового напряжения при помощи изменения напряжения на обратном затворе (Ub).

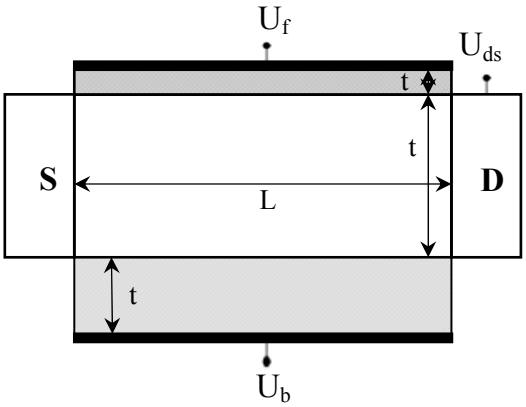


Рис. 1. Схема полностью обедненного КНИ КМОП транзистора

Если инверсия обратной поверхности отсутствует (обратная поверхность в режиме обеднения), величина порогового напряжения и, таким образом, значение активной проводимости корректируются только смещением на обратном затворе. При помощи, как известно, емкостной связи [2].

В данном случае процесс переноса заряда в транзисторе происходит в области, связанной с фронтальной поверхностью. Что позволяет рассматривать только один канал, ток зарядов в котором характеризуется одной подвижностью. Что иллюстрируется Рис. 2а.

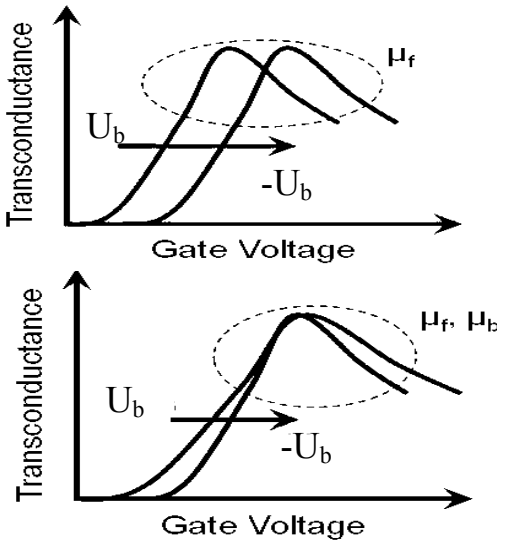


Рис. 2: а - (верхний) типичное, б – (нижний) аномальное поведение проводимости

Напротив, если есть инверсия обратной поверхности, вклад канала связанного с обратной поверхностью резко возрастает. И процесс переноса заряда нужно рассматривать с учетом этого механизма. В этом случае, поведение проводимости для различных Ub уже не коррелирует со сдвигом порогового напряжения, показано на рис. 2б.

Пик и наклон проводимости может варьироваться в результате изменения соотношения фронтальной и обратной подвижностей, в зависимости от состояний фронтальной и обратной поверхностей. Этот механизм - “аномальная” подвижность приводит к сложному поведению активной проводимости, которая может наблюдаться когда:

1) при высоких уровнях Ub (положительное/ отрицательное для n/p транзисторов), чтобы инвертировать обратную поверхность
2) фронтальный/обратный подзатворный диэлектрик - толстый (тонкий).

Кроме того, распределение потенциала в рабочей области транзистора определяется емкостями фронтального и обратного затворов. Утолщение окисла фронтального затвора приводит к меньшему количеству заряда во фронтальном канале и, таким образом, к более высокому влиянию заряда и подвижности на обратной поверхности. Поэтому и одновременное рассмотрение подвижности фронтальной и обратной поверхностей крайне важно для полностью обедненной структуры для широкого

диапазона смещений и различных топологий устройства.

Вольтамперные характеристики (в частности, области затворных напряжений выше порогового) двух затворного полностью обедненного КНИ полевого транзистора с утолщенным и тонким подзатворным окислом фронтального затвора, которые получены по традиционной (коммерчески доступной) технологии, приведены в [10]. Эти два устройства демонстрируют различные поведения активной проводимости. Пик проводимости транзистора с утолщенным окислом фронтального затвора немонотонным образом мигрирует сначала направо при низком значении напряжения на фронтальном затворе и затем налево при высоком напряжении с ростом напряжения на нижнем затворе. Это характеризуют как проявление “аномальной” проводимости. Для транзистора с тонким подзатворным диэлектриком фронтального затвора характерна монотонная характеристика (сдвиг влево) с напряжением смещения обратного затвора. При моделировании при помощи модели с одним каналом переноса носителей, ни при каких условиях невозможно получить немонотонный сдвиг пика активной проводимости. Такое поведение межэлектродной проводимости непосредственно связано с генерацией второго канала (вдоль обратной поверхности в случае полностью обедненного КНИ транзистора), а также разными подвижностями присущими этим каналам. В результате необходимо расширить модель подвижности, чтобы воспроизводить этот немонотонный эффект.

Из-за смещения напряжения на обратном затворе происходит перераспределение заряда в рабочей области транзистора. Инверсия обратной поверхности также влияет на потенциал и его производную. В результате различных сценариев изменения Ub эффективное электрическое поле характеризуется неоднозначным поведением. Тогда структура с более утолщенной окисью фронтального затвора обнаруживает большее влияние нижнего затвора в ее поведении из-за более слабого управления фронтального затвора. Для такого устройства, фронтальная и обратные поверхностные каналы сосуществуют и характеризуются различными эффективными подвижностями носителей из-за отличных свойств поверхностей [7]. Кроме того, инверсионные заряды на фронтальной и обратной поверхностях испытывают неодинаковые электрические поля из-за различных прикладываемых смещений на затворах. Поэтому, необходимо расширить модель переноса носителей, базируемой только на фронтальной поверхностной электростатике включением в нее и модели переноса, связанной с обратной поверхностью. Для двух затворных транзисторных структур известна модель подвижности, которая включает поверхностное рассеивание на шероховатости и кулоновское рассеивание [13]. Она широко используется и хорошо согласуется с экспериментальными данными. Таким образом, выражение для подвижности для фронтальной и обратной поверхности подобны, но значения параметров, входящих в них различны. В результате общая эффективная подвижность является суммой эффективных подвижностей на фронтальной и обратной поверхностях с учетом их весовых функций:

$$\mu_{tot} = w_f \mu_{eff_f} + w_b \mu_{eff_b}, \qquad (1)$$

где μ_{eff_f}, μ_{eff_b} - эффективные подвижности на фронтальной и обратной поверхностях, соответственно, w_f, w_b - весовые коэффициенты для фронтальной и обратной подвижностей, соответственно.

Весовые коэффициенты в (1) являются безразмерной функцией отношения зарядов ассоциирующихся с фронтальной и обратной поверхностям. Чем больший заряд на поверхности, тем больший вклад в эффективную подвижность делает эта поверхность. Весовые функции в [14] применимы только для сильной инверсии. В настоящей работе они рассматривается их новый вариант, который применим во всей области функционирования транзистора. Сумма инверсионных зарядов - показательная функция поверхностного потенциала, являющаяся решением 2D уравнение Пуассона. Оно получено в приближении статистики Больцмана (приближение из статистики Ферми-Дирака) [15]. В подпороговой области, поверхностный потенциал варьируется линейно со смещением затвора, таким образом, инверсионный заряд - показательная функция напряжения затвора. В сильной инверсии, так как поверхностный потенциал имеет слабую зависимость от смещения затвора, инверсионный заряд характеризуется линейной зависимостью от напряжения затвора. Следовательно, весовые функции для фронтальной и обратной поверхности можно записать в виде:

$$w_f = \frac{\exp(\frac{\varphi_f - \varphi_{F_1}}{\varphi_T})}{\exp(\frac{\varphi_f - \varphi_{F_1}}{\varphi_T}) - \exp(\frac{\varphi_b - \varphi_{F_2}}{\varphi_T})}, \quad (2)$$

$$w_b = \frac{\exp(\frac{\varphi_b - \varphi_{F_2}}{\varphi_T})}{\exp(\frac{\varphi_f - \varphi_{F_1}}{\varphi_T}) - \exp(\frac{\varphi_b - \varphi_{F_2}}{\varphi_T})}, \quad (3)$$

где φ_f и φ_b - поверхностные потенциалы на фронтальной и обратной поверхностях, $\varphi_{F_{1(2)}}$ – квази фермиевский потенциал на фронтальной и обратной поверхностях, соответственно, φ_T - тепловой потенциал. Тонкий фронтальный (обратный) подзатворный диэлектрик повышают соответствующий поверхностный потенциал, обуславливая этим более высокий весовой коэффициент. Если фронтальный окисел достаточно толстый (приблизительно одна пятая толщины окисла обратного затвора), вклад обратного канала доминирует в gm из-за большого весового коэффициента для обратной поверхности.

2. Результаты моделирования

Для реализации модели с двумя подвижностями используется базовая модель (с одной подвижностью), которая достоверно воспроизводит эффект инверсии обратной поверхности. На основе точного решения для заряда и поверхностного потенциала на основе базовой модели, весовые функции (2) и (3) однозначно определяются.

Рис. 3 иллюстрирует результаты алгоритма извлечения взвешенной функции для рассматриваемых транзисторов из экспериментальных данных для активной проводимости.

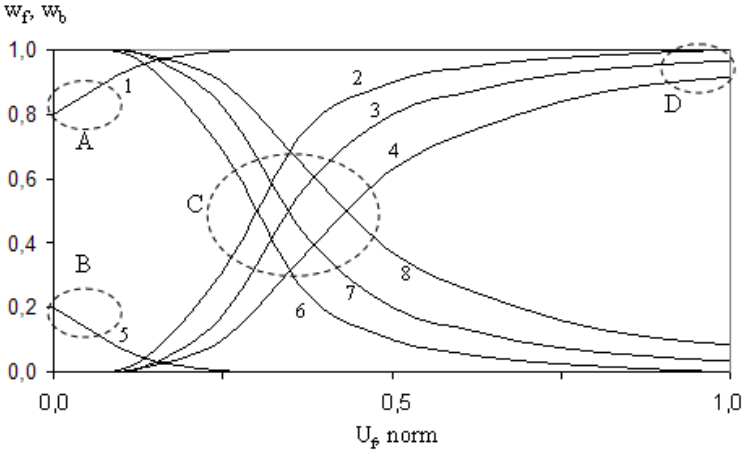


Рис. 3. Взвешенные функция для полностью обедненного КНИ КМОП транзистора с толстым подзатворным диэлектриком фронтального затвора при линейном возрастании Ub, где 1-4 – wf, 5-8 - wb.

При высоком уровне Ub и низких значениях Uf (подпороговая область), т.е. выполнены условия инверсии обратной поверхности, заряд в канале доминируют в канале у обратной поверхности и весовая функция для обратной поверхности больше, чем та же для фронтальной поверхности. Это выделено областями A и B на Рис. 3. При увеличении напряжения Uf, весовая функция wf возрастает и весовые функции пересекаются, что отмечено областью C. В конечном счете весовой коэффициент wf становится намного больше, чем коэффициент wb, указывая на доминирование заряда, ассоциированного с фронтальным затвором, как следствие превалирование мобильности. Эта область отмечена кругом D на рис. 3. Точка пересечения фунций показывает возникновение поверхностного рассеивания на шероховатости. Другими словами, во фронтальном канале концентрация носителей достаточно высока и напряжение фронтального затвора притягивает их к фронтальной поверхности, приводя к дополнительному рассеиванию.

На Рис. 4 приведены экстрагированные зависимости эффективной подвижности для фронтального канала и каналу у обратной поверхности от смещения обратного затвора.

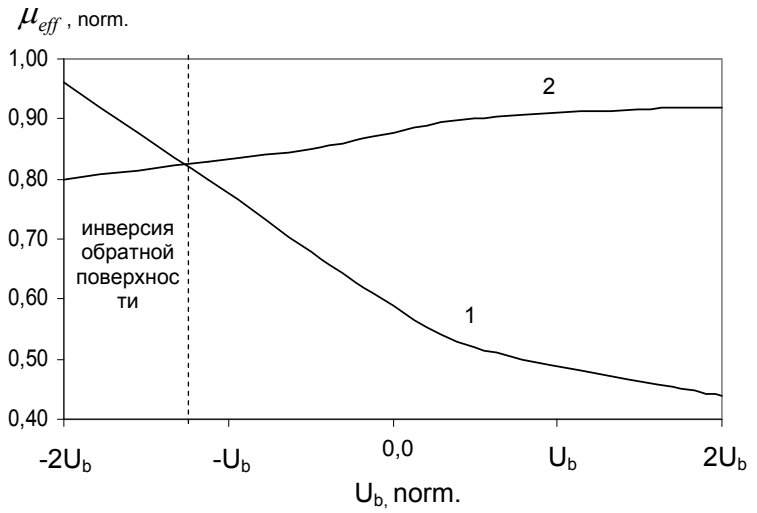


Рис. 4. Экстрагированная (из данных приведенных в [13]) зависимость эффективной подвижности для фронтального (μ_{eff_f}) и обратного (μ_{eff_b}) каналов от напряжения на обратном затворе, где 1- $\mu_{eff_f}(U_b)$, 2 - $\mu_{eff_b}(U_b)$.

Параметр μ_{eff_b} практически во всем диапазоне превышает подвижность μ_{eff_f} , кроме случая высоких значений $|U_b|$. Это вызвано тем, что при больших значениях |Ub| носители из фронтального канала смещаются в глубь рабочей области, удаляясь от поверхности, что приводит к уменьшению поверхностного рассеяния и улучшения мобильности носителей. Для канала у обратной поверхности при высоких |Ub| носители на оборот прижимаются к поверхности, тем самым увеличивая поверхностное рассеяние и уменьшая подвижность. Кроме того, зависимость $\mu_{eff_f}(U_b)$ имеет меньшую крутизну и противоположный наклон характеристики, чем зависимость $\mu_{eff_b}(U_b)$.

Заключение

Рассмотрены физические особенности аномальной подвижности, которая наблюдается в полностью обедненном полевом транзисторе со структурой «кремний на изоляторе» с толстым слоем подзатворного окисла фронтального затвора.

Анализ вольт-амперных характеристик транзисторов с тонким и утолщенным подзатворными диэлектриками фронтального затвора позволил оценить вклад канала проводимости вдоль обратного затвора в общую структуру переноса носителей.

В результате уточнено выражение для эффективной подвижности, которое состоит из двух компонент: эффективной подвижности фронтального канала и эффективной подвижности канала вдоль обратной поверхности.

Эти два параметра связаны между собой взвешенной функцией следующей из приближения зарядового разделения.

Рассмотрен алгоритм экстракции взвешенной функции из экспериментальных данных. Формат выражения общей подвижности таков, что он легко встраивается в известные компакт-модели транзистора. Применение предложенной модели будет полезным при разработке высоковольтных схем и устройств обработки изображений.

Работа выполнена при поддержке Программы фундаментальных исследований РАН I.2.П27 НИР № 0065-2018-0113.

Features of anomalous mobility in the fully depleted SOI transistors

N. Masalsky

Abstract. Issues of simulation of anomalous mobility which is observed in fully depleted field transistor with structure "silicon on the insulator" with a thick oxide layer of a front gate are discussed. It is found that the anomalous behavior is defined of domination of a charge in the channel at the back surface in the total charge. This behavior is described by model with two mobility which correspond to the front channel and the channel along the back surface. These two mobility are physically connected by means of the weighed function following from approach of charging division and is extracted from experimental data.

Keywords: fully depleted SOI CMOS transistor, silicon on isolator structure, mobility, surface potential distribution

Литература

1. J.P. Colinge. Multiple-gate SOI MOSFETs. “Solid-State Electronics”. v. 48(2004), № 3, 897-909.
2. С.М. Захаров, Н.В. Масальский, М.М. Шаффигуллин. Проблемы схемотехнического моделирования интегральных схем. “Успехи радиоэлектроники”, т. 12(2005), № 2, 43-50.
3. S. Khandelwal. BSIM-IMG: A compact model for ultrathin-body SOI MOSFETs with back-gate control.”IEEE Trans. Electron Devices”, v. 59(2012), № 8, 2019–2026.
4. Y.-K. Lin. Modeling of back-gate effects on gate-induced drain leakage and gate currents in UTB SOI MOSFETs. “IEEE Trans. Electron Devices”, v. 64(2017), № 10, 3986–3990.
5. A. Ohata, S. Cristoloveanu, M. Casse. Mobility comparison between front and back channels in ultrathin silicon-on-insulator metal-oxide-semiconductor field-effect transistors by the front-gate split capacitance-voltage method. ”Appl. Phys. Lett.”, v. 89(2006), № 3, 032104-1–032104-3.
6. S. Cristoloveanu, N. Rodriguez, F. Gamiz. Why the universal mobility is not. ”IEEE Trans. Electron Devices”, v. 57(2010), № 6, 1327–1333.
7. T. Rudenko. Experimental study of transconductance and mobility behaviors in ultra-thin SOI MOSFETs with standard and thin buried oxides. “Solid-State Electron.”. v. 54(2010), № 2, 164–170.
8. H. Agarwal, C. Gupta, S. Dey, S. Khandelwal, C. Hu, Y. S. Chauhan. Anomalous transconductance in long channel halo implanted MOSFETs: Analysis and modeling. “IEEE Trans. Electron Devices”. v. 64(2017), № 2, 376–383.
9. Н.В. Масальский. Характеристики двух затворных КНИ КМОП нанотранзисторов для перспективных технологий с низким уровнем потребляемой мощности. «Микроэлектроника» . т. 41(2012). № 6, 436-444.
10. V. Suntharalingam. Megapixel CMOS image sensor fabricated in three-dimensional integrated circuit technology. in IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers, 2005, 356–357
11. M. Goto. Pixel-parallel 3-D integrated CMOS image sensors with pulse frequency modulation A/D onverters developed by direct bonding of SOI layers. “IEEE Trans. Electron Devices”. v. 62(2015), № 11, 3530–3535.
12. P. Coudrain. Investigation of a sequential three-dimensional process for back-illuminated CMOS image sensors with miniaturized pixels. “IEEE Trans. Electron Devices”. v. 56(2009), № 11, 2403–2413.
13. Y. S. Chauhan. BSIM6: Analog and RF compact model for bulk MOSFET. “IEEE Trans. Electron Devices”, v. 61(2014), № 2, 234–244.
14. Y. Sahu, P. Kushwaha, A. Dasgupta, C. Hu, Y. S. Chauhan. Compact modeling of drain current thermal noise in FDSOI MOSFETs including back-bias effect. “ IEEE Trans. Microw. Theory Techn.”, v. 65(2017), № 7, 2261–2270.
15. Н.В. Масальский. Моделирование распределения поверхностного потенциала в двух затворных симметричных КНИ КМОП нанотранзисторах с гауссовым вертикальным профилем легирования. «Труды НИИСИ РАН», т. 6(2016), № 1, 78-83.

Тестирование инструментальных средств разработки ПО

В.А.Галатенко [1], К.А. Костюхин [2], М.Д.Дзабраев [3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1] galat@niisi.ras.ru, [2] kost@niisi.ras.ru, [3] mdzabraev@niisi.ras.ru

Аннотация: Настоящая статья посвящена обзору инструментария автоматизированного тестирования DejaGnu, на основе которого в проекте GNU были разработаны тесты для GCC и GDB. Также в рамках данной статьи будут подробно рассмотрены тесты GCC и GDB, применяемые в ФГУ ФНЦ НИИСИ РАН для проверки качества разрабатываемых программных изделий.

Ключевые слова: автоматическое тестирование, gcc, gdb, компилятор, отладчик.

1. Введение

Качество программного обеспечения напрямую зависит от того, насколько хорошо оно, то есть ПО, протестировано. На протяжении многих лет люди со всего мира активно используют инструменты, разработанные в рамках проекта GNU (gnu.org). Наиболее востребованными являются такие инструменты, как набор компиляторов – GCC (GNU Compiler Collection), и отладчик GDB (GNU Debugger). Упомянутые программы являются очень громоздкими и сложными, если рассматривать их с точки зрения разработчика. Однако, вместе с тем, они весьма надежны. Можно сказать, что надлежащий уровень надежности был достигнут, в том числе, и благодаря правильной методике тестирования и обширному набору тестов.

2. Инструментарий DejaGnu

DejaGnu [4] является инструментарием для автоматизированного тестирования ПО. Изначально данный фреймворк был разработан для тестирования программ, разрабатываемых в рамках проекта GNU.

DejaGnu может использоваться без привлечения сторонних программ, однако, становится более полезным, если он используется вместе с GNU autotools (autoconf, automake и libtool).

Тесты, которые исполняются в рамках фреймворка DejaGnu должны быть написаны на языке программирования Tool Command Language (TCL). DejaGnu включает в себя код библиотек на языке TCL. Совокупность библиотечного кода DejaGnu и языка TCL называется *expect*.

DejaGnu предназначается для использования в POSIX 1003.3 совместимых операционных системах. DejaGnu включает в себя команду runtest для запуска тестов, язык программирования, на котором пишутся тесты (TCL), стандартную структуру каталогов для тестов, и набор конфигурационных файлов.

2.1. Краткий обзор DejaGnu

Команда runtest.

Наиболее важной для DejaGnu является команда runtest. Данная команда имеет следующий формат запуска

runtest [options] [test]

Выделим следующие опции:

--tool toolname

Данная опция определяет название группы тестов, которые будут выполняться. В частности, это может быть имя утилиты, но также, toolname может описывать тип проводимых тестов(например, unit).

Аргумент toolname важен, поскольку он используется для создания имен различных сущностей: каталоги, в которых могут быть найдены тесты, а также имена конфигурационных файлов, имеющих отношение к объекту тестирования (тестируемой утилите).

--srcdir dir

Данная опция описывает путь, где могут быть найдены каталоги с тестами. Каталог, содержащий конфигурационные файлы тестируемой утилиты (lib) также может быть найден по этому пути.

К именам каталогов, которые содержат тесты, может быть добавлен

префикс с названием тестируемой утилиты, а также может быть добавлен суффикс .tests. Таким образом для gdb, тесты могут находиться в каталоге с именем gdb.tests, или gdb-profiling.tests.

--outdir dir

Данная опция описывает каталог, в который будут помещаться все служебные сообщения, возникшие во время исполнения тестов.

--target_board board

Определяет целевую платформу, для которой будут выполняться тесты. Также она определяет, какие конфигурационные файлы будут использоваться.

Если параметр test не был передан команде runtest, то в этом случае будут выполнены все тесты, которые runtest сможет найти для значения опции toolname. В противном случае будут выполнены только указанные тесты.

Expect.

Общая схема тестирования выглядит следующим образом. Есть тестируемая программа, которая может принимать на свой стандартный вход команды от пользователя, и обработав их, выдать результат на стандартный вывод. Также есть программа, написанная на expect, которая проверяет тестируемую программу. Проверка заключается в том, что тестируемой программе на стандартный ввод посылаются команды, после чего захватывается вывод тестируемой программы, и захваченный вывод сопоставляется шаблону.

2.2. Стандартная структура каталогов DejaGnu

DejaGnu ожидает, что некоторые файлы будут размещаться в заранее оговоренных каталогах. Однако данное поведение фреймворка может быть изменено из командной строки через опции утилиты runtest.

Существует соглашение, что тестовый фреймворк располагается в каталоге testsuite. Отмеченная директория содержит следующие каталоги:

--config

Это необязательный каталог, который содержит конфигурационные файлы для различных целевых платформ. Файлы данного каталога будут использоваться как часть конфигурации при тестировании.

--lib

Данный каталог содержит файлы для каждой из тестируемых программ. Файлы этого раздела носят названия toolname.exp, где toolname является именем тестируемой программы.

Каталоги с тестами.

Каталоги, которые носят имена toolname[type].tests содержат тесты на expect для программы toolname.

Необязательное поле type позволяет группировать тесты и расположить группы тестов в разные каталоги. Например, компилятор с именем xcc может обладать тестами компиляции (в каталоге xcc-compile.tests) и тестами исполнения (xcc-execute.tests).

Строго говоря, суффикс .tests не является обязательным, однако это общепринятое соглашение. Внутри каждого каталога с тестами располагаются файлы с тестами, которые написаны на expect, причем каждый такой файл имеет суффикс .exp.

2.3. Конфигурационные файлы DejaGnu

Конфигурационные файлы используются для инициализации глобальных переменных и определения вспомогательных функций.

DejaGnu имеет ряд глобальных переменных, причем задавать значения этих переменных можно через аргументы командной строки утилиты runtest. Однако данные переменные могут быть установлены в различных конфигурационных файлах (если используется оба способа задания значения переменной, то задание через аргументы командной строки обладает большим приоритетом). Ниже приводятся наиболее полезные опции runtest.

runtest.	Описание:
--all	Отображать (на консоль) все тестовые результаты. По умолчанию отображаются только тесты, завершившиеся неуспешно.
--srcdir	Каталог, содержащий подкаталоги с тестами и каталог lib.
--tool	Название программы, которая будет подвергаться тестированию.
--verbose	Уровень отладочного вывода. По умолчанию данное значение

равняется нулю.

DejaGnu обладает множеством конфигурационных файлов, каждый из которых представляет из себя программу на expect. Конфигурационные файлы исполняются перед каждым тестом в следующем порядке.

Персональные конфигурационные файлы.

Конфигурационные файлы данного типа располагаются в каталоге ~/.dejagnu.

Локальный конфигурационный файл.

Данный файл всегда называется site.exp и он находится в директории, где запускаются тесты. Данный файл используется для инициализации переменных, которые относятся к тестируемой программе.

Если для управления DejaGnu используются GNU autotools (autoconf, automake и libtool), то site.exp будет генерироваться автоматически, всякий раз, когда будет введена команда make. Сгенерированный файл будет располагаться в корневом каталоге тестового фреймворка. Стоит отметить, что если сборка программы происходила в отдельном от исходных кодов каталоге, то будет два корневых каталога testsuite. Один будет находиться вместе с исходными кодами, второй в каталоге, где происходила сборка. В этом случае тесты должны запускаться в каталоге testsuite, который располагается там, где происходила сборка программы. Файл site.exp будет сгенерирован в этом же каталоге.

Глобальный конфигурационный файл.

Данный файл предназначается для распространения настроек на группу тестов. Он определяется при помощи переменной окружения DEJAGNU.

Конфигурационный файл целевой системы

Данный файл предназначен для определения специфических настроек для определенной целевой платформы. Он определяется при помощи передачи опции --target_board команде runtest. Если эта опция не передавалась, тогда DejaGnu выберет в качестве целевой платформы unix платформу и, как следствие, будет использован конфигурационный файл unix.exp.

Тестовый фреймворк будет производить поиск конфигурационных файлов в каталогах, которые указаны в переменной boards_dir, по умолчанию включающей каталоги

/usr/share/dejagnu/baseboards

и

/usr/share/dejagnu/config.

Добавление дополнительных каталогов для поиска конфигурационных файлов является обычной практикой для предоставления дополнительной информации о целевой платформе. Операция добавления дополнительных каталогов может быть выполнена при помощи функции lappend. Например, lappend boards_dir "$srcdir/boards" добавит к списку boards_dir каталог "$srcdir/boards".

Замечание: Если будет найдено несколько конфигурационных файлов для описания целевой платформы, то они будут выполнены в том же порядке, в каком были найдены.

Конфигурационный файл тестируемой программы.

Последний конфигурационный файл, который DejaGnu будет выполнять, это конфигурационный файл, предназначенный для тестируемой программы. Тестовый фреймворк будет производить поиск данного файла в каталоге lib. Для программы, которая носит имя toolname, файл должен называться lib/toolname.exp. Этот конфигурационный файл предназначается для определения вспомогательных функций тестирования для программы toolname.

2.4. Создание тестов

Каждый DejaGnu тест представляет из себя последовательность команд языка expect (TCL). Это стандартные TCL команды с некоторыми дополнениями, которые делают язык практически применимым для тестирования. Наиболее полезными являются следующие команды:

expect

Данная команда является наиболее важной среди всех команд.

В качестве аргументов подается последовательность пар шаблон-действие, после чего команда ожидает, пока один из шаблонов не совпадет с выводом тестируемой программы, или пока не истечет время ожидания.

Если вывод программы удовлетворяет шаблону, то будет выполнено соответствующее действие.

send
Эта команда используется для отправки данных на стандартный ввод тестируемой программе.
spawn
Команда spawn создает дочерний процесс и запускает в нем программу, которой будут переданы аргументы. Также при порождении дочернего процесса spawn переопределит стандартные потоки, чтобы expect мог читать из них и отправлять в них данные.

Замечание: Словом expect называется расширение языка TCL, и, вместе с этим, также называется команда данного расширения.

Помимо этого, DejaGnu определяет ряд функций, которые облегчают тестирование. Множество таких функций описано в руководстве по DejaGnu [3], в разделе Unit Testing. Наиболее полезными являются следующие функции: fail, pass, xfail, xpass, untested, unresolved, unsupported. Данные функции сообщают пользователю о результатах тестирования.

warning
Данная функция записывает сообщение в лог-файл, добавляя в начало сообщения WARNING.

Perror
Эта функция записывает сообщение в лог-файл, предваряя его словом ERROR.

2.5. Результаты тестирования

Результаты выполнения каждого теста, который встретил runtest, выводятся в файл-журнал, причем runtest ведет подсчет результатов тестов, чтобы вывести краткую статистику по окончании тестирования. Результат тестирования отдельного теста определяется тем, какая из функций fail, pass, xfail, xpass, untested, unresolved, unsupported была вызвана. Каждой из этих функций на вход подается строка, которая характеризует текущий тест.
Возможные результаты тестирования могут быть следующими:

FAIL
Эта строка означает, что тест был провален. Результат получается посредством вызова функции fail.

PASS
Данная строка означает что тест пройден. Результат порождается вызовом функции pass.

XFAIL
Эта строка означает что тест был провален, причем этот провал был ожидаемым. Данный результат достигается вызовом функции xfail.

Замечание: Стандарт POSIX 1003.3 не поддерживает ожидаемые провалы тестов (тест или проваливается или выполняется), таким образом тесты, совместимые с данным стандартом, не должны порождать результатов типа XFAIL.

XPASS
Данный результат означает, что для данного теста ожидался провал, однако, тест был пройден успешно. Этот результат достигается посредством вызова функции xpass.

Замечание: POSIX 1003.3 не поддерживает данный результат.

UNTESTED
Этот результат означает, что данная возможность не была протестирована. Данный результат тестирования достигается при помощи вызова функции untested. Он является хорошим способом отметить тесты, которые в настоящее время недоделаны.

NRESOLVED
Результат означает, что вывод теста должен быть проверен экспертом вручную. Результат достигается при помощи вызова функции unresolved. Нормальной практикой является вызов данной функции, когда истекло время ожидания в функции expect.
Замечание: если во время теста была вызвана функция perror, или более чем warning_threshold раз была вызвана функция warning, то вызов любой из функций, сигнализирующих о результате выполнения теста будет вести себя, как функция unresolved.

UNSUPPORTED
Означает, что этот тест не поддерживается. Результат достигается при помощи вызова unsupported. Данная функция используется, когда тест зависит от наличия некоторой возможности. Например, некоторые тесты не могут быть выполнены на определенных целевых платформах.

2.6. Команда expect

Синтаксис команды expect выглядит следующим образом:
expect -flags1 pat1 body1 ... -flagsn patn bodyn

Данная команда ожидает, пока один из шаблонов не совпадет с выводом тестируемой программы, или пока не наступит таймаут, или конец файла не будет достигнут. В случае совпадения вывода тестируемой программы и шаблона будут выполнены соответствующие инструкции.
Если команда занимает более одной строки (а как правило это так), то аргументы команды должны заключаться в фигурные скобки.
Простой пример применения команды expect:

```
expect {
  ERROR           {fail "Error encountered.\n";}
  "Test complete" {pass "Test completed.\n";}
  timeout         {unresolved "Timeout.\n";}
}
```

Команда expect возвращает в качестве результата результат выполнения соответствующего блока инструкций. Под соответствующим блоком понимается тот блок, парный шаблон которого совпал с выводом тестируемой программы.

2.7. Шаблоны в команде expect

Существует широкий диапазон шаблонов, которые могут быть использованы в команде expect.

Обычные строки

В предыдущем примере были использованы строки ERROR и "Test complete". Двойные кавычки используются для задания строк, которые содержат пробелы.

eof
Совпадение с данным шаблоном происходит, если в потоке вывода тестируемой программы был достигнут end of file. Обычно end of file наступает, если тестируемая программа завершила свое исполнение.

timeout
Совпадение происходит, если прошло timeout секунд и вывод тестируемой программы не совпал ни с одним из шаблонов команды expect. По умолчанию timeout равняется 10 секундам, но его можно изменить при помощи присваивания глобальной переменной timeout нужного значения (например, set timeout 15;). Значения таймаута может быть задано только для текущего вызова команды expect, при помощи использования флага -timeout. Например,
expect { -timeout 5 timeout {puts "timeout occurs"; } }.
default
Данный шаблон эквивалентен сопоставлению или eof или timeout. Перед данным паттерном можно использовать флаг -timeout t для задания таймаута.

full_buffer
По умолчанию команда expect буферизует до 2000 байтов. В случае заполнения буфера старые байты будут забываться. Данный шаблон будет сопоставлен, если буфер будет заполнен, тем самым данный шаблон позволяет отловить ситуацию заполнения буфера.

Замечание: Размер буфера может быть увеличен при помощи использования процедуры match_max. Однако очень большие значения могут замедлить процедуру сопоставления шаблону.

null
Этот шаблон соответствует одиночному ASCII символу 0.

2.8. Пример теста

По существу, написание теста заключается в следующем.

Сначала порождается дочерний процесс (при помощи команды spawn), в рамках которого будет исполняться тестируемая программа.

Затем используется expect для сопоставления вывода и сигнализировании об провалах, в случае если не удалось сопоставить вывод.

Ниже приводится простейший пример.

```
# уменьшение таймаута до 3х секунд.
set timeout 3

# Имя теста и команда для запуска
программы, которая будет тестироваться
set test_name "Simple test"
set command_line "or32-elf-sim -f default.cfg
test-prog.or32"

# Если задан отладочный вывод, то сказать
пользователю, что тест запущен.
if { $verbose > 1 } {
  send_user "starting $command_line\n"
}
```

```
# Запуск тестируемой программы
spawn $command_line

expect {
  # Проверка на наличие warning'ов в выводе
тестируемой программы.


Warning {
     fail "$test_name: warning:
$expect_out(buffer)"
  }

  # проверка на наличие ошибок в выводе.
  ERROR {
     fail "$test_name: error:
$expect_out(buffer)"
  }

  # Строка обозначающая, что тест
успешно пройден.
  "Test complete" {
      pass "$test_name\n"
  }

  # EOF и timeout идут после всего
остального. EOF считается ошибкой.
  eof {
     fail "$test_name: EOF\n"
  }

  # Timeout требует проверки вывода
экспертом для установления причины
ошибки.
  timeout {
     unresolved "$test_name: timeout"
  }
}
```

Этот тест сначала пытается сопоставить строки "Warning" или "ERROR" выводу тестируемой программы.

Если вывод программы не совпал с первыми двумя шаблонами, то происходит попытка сопоставления вывода шаблону "Test complete".

Последние два шаблона представляют из себя проверки конца файла и таймаут.

Порядок расположения шаблонов имеет значение.

Если тестируется маленькая программа, то к моменту сопоставления шаблону в буфере команды expect мог поместиться весь вывод программы: "Test complete" и end-of-file. В связи с этим "Test complete" должен проверяться первым.

Также программа во время своего исполнения могла сгенерировать предупреждения и сообщения об ошибках, и, в конечном итоге, вывести строку "Test complete".

Поэтому сначала нужно проверить предупреждения и ошибки.

2.9. Автоматизация тестов

Пример из предыдущего раздела достаточно общий, поэтому из него можно сделать функцию для тестирования.

Этой функции будут подаваться на вход имя теста и команда для запуска программы.

```
proc runmytest { test_name command_line } {
 global verbose
 if { $verbose > 1 } {
    send_user "starting $command_line\n"
 }
 spawn $command_line
 expect {
    Warning {
    fail "$test_name: warning:
$expect_out(buffer)"
    }
    ERROR {
    fail "$test_name: error: $expect_out(buffer)"
    }
    "Test complete" {
        pass "$test_name\n"
    }
    eof {
          fail "$test_name: EOF\n"
    }
    timeout {
       unresolved "$test_name: timeout"
    }
 }
}
```

Код аналогичен коду из предыдущего раздела, за исключением того, что теперь этот код содержится внутри функции, а также, для того, чтобы переменная verbose была видна, нужно указать global verbose.

Таким образом, группа тестов может выглядеть следующим образом:

```
# Уменьшение таймаута до 3х секунд.
set timeout 3
runmytest "Simple test"  "or32-elf-sim -f
default.cfg test-prog.or32"
runmytest "Harder test"  "or32-elf-sim -f
default.cfg test-prog2.or32"
```

runmytest "Hardest test" "or32-elf-sim -f default.cfg test-prog3.or32"

Стоит отметить, что представленная функция используется в нескольких тестах. В связи с этим, эта функция должна находиться в конфигурационном файле, который соответствует тестируемой программе.

2.10. Использование DejaGnu без autotools

Для запуска тестового фреймворка DejaGnu без autotools нужно выполнить следующие действия.

1 Создать структуру каталогов для тестового фреймворка.
2 Написать программу, которая будет подвергаться тестированию, и откомпилировать ее.
3 Написать тесты на expect и положить их в каталоги с тестами, причем каталоги должны соответствовать названию тестируемой программы.
4 Поместить различные вспомогательные TCL-функции в конфигурационные файлы, соответствующие тестируемой программе.
5 Запустить программу runtest, передав ей имя тестируемой программы и каталог с исходными кодами фреймворка.

3. Использование фреймворка DejaGnu для тестирования GDB

Отладчик Gnu Debugger (GDB) является программой с открытым исходным кодом и предназначается для интерактивной отладки программ, написанных на языке C/C++, Ada, Fortran и др.

В качестве пользовательского интерфейса GDB предоставляет интерфейс командной строки. Типичный сеанс отладки заключается в том, что пользователь набирает текстовые команды, и затем отладчик в текстовом формате возвращает результат выполнения команды. Наиболее распространенными командами являются такие команды, как:

1 break
Поставить точку останова на функцию/строку кода
2 step
Сделать шаг на следующую строку кода
3 print
Распечатать значение переменной
4 backtrace
Распечатать стек вызовов функций от точки входа в программу до текущей функции.

Тестирование отладчика выполняется посредством выполнения следующих действий.

1 Компиляция тестовой программы.
2 Запуск отладчика GDB, загрузка скомпилированного файла в отладчик и запуск программы.
3 Отправка текстовых команд на стандартный вход отладчика и проверка результата выполнения команды при помощи expect.
4 Завершение тестирования.

Тесты отладчика GDB распространяются вместе с его исходными кодами. Тесты располагаются в каталоге gdb/testsuite.

Наибольший интерес представляет каталог gdb/testsuite/gdb.base. Данный каталог содержит файлы с программами, написанными на языке C, а также сценарии, написанные на expect. Рассмотрим два теста из этого каталога, которые называются bitfields.exp и funcargs.exp.

3.1. Библиотека lib/gdb.exp

Для облегчения написания тестов отладчика GDB вместе с тестами распространяется библиотека, которая располагается в каталоге gdb/testsuite/lib/gdb.exp. Практически все тесты отладчика используют данную библиотеку. Далее приводятся названия наиболее часто используемых функций и их описание.

gdb_test COMMAND PATTERN MESSAGE QUESTION RESPONSE

Данная функция отправляет команду на стандартный вход отладчика и сопоставляет шаблону результат выполнения команды.

1. COMMAND
команда, которая будет отправлена на стандартный вход отладчика. Если данная строка является null-строкой, то команда не будет посылаться в отладчик.
2. PATTERN
представляет из себя шаблон для тестирования результата выполнения

команды. Данный аргумент может быть опущен.

3. MESSAGE

является необязательным аргументом. Если данный аргумент опускается, то в качестве MESSAGE будут использоваться строки pass или fail.

4. QUESTION

В процессе сеанса отладки некоторые команды GDB могут выводить пользователю вопрос. Аргумент QUESTION является строкой и может либо совпадать с вопросом, который отобразит GDB, либо не совпадать. В случае совпадения будет возвращаться аргумент RESPONSE.

runto FUNCTION ARGS

Данная команда выполняет операцию установки точки останова на FUNCTION, после чего выполняет операцию continue.

1. FUNCTION

Данный аргумент представляет название функции, на которую будет установлена точка останова.

2. ARGS

представляет из себя список аргументов.

Этот список может содержать произвольное число аргументов. аргумент ARGS будет передаваться в функцию gdb_breakpoint. По умолчанию в функции runto не вызываются функции pass и fail. Если в ARGS будет содержаться "message", то функции pass и fail будут вызываться. Если после "message" встретится аргумент "no-message", то функции pass, fail вызываться не будут.

gdb_breakpoint FUNCTION ARGS

Установить точку останова на функцию FUNCTION.

1. FUNCTION название функции, на которую будет установлена точка останова.

2. ARGS является списком и может содержать следующие опции: "allow-pending", "temporary", "message", "no-message". Опция "allow-pending" позволяет ставить отложенные (pending) точки останова. Если была передана опция “temporary” , то будет поставлена временная точка останова (команда tb).

delete_breakpoints

Данная функция удаляет все точки останова.

gdb_prompt

Представляет глобальную переменную, которая содержит строку приглашения к вводу. Стандартное приглашение к вводу в GDB является строка "(gdb) ".

3.2 . Тесты funcargs.exp

Тесты, находящиеся в файле funcargs.exp, очень важны, поскольку практически полностью воспроизводят типичный сеанс отладки.

Тесты integral args

В группу входит ряд тестов (название которых содержит суффикс _integral_args). Типичный тест из этой группы выглядит так. На начало функции ставится точка останова, затем в отладчик отправляется команда continue. После данных действий должен произойти останов на начале функции. Функции, на которые ставятся точки останова, имеют следующий вид.

```
void call0a (char c, short s, int i, long l)
{
  c = 'a';
  s = 5;
  i = 6;
  l = 7;
}
```

При останове на функции отладчик должен вывести подобную строку:

Breakpoint 1, call0a (c=97 'a', s=1, i=2, l=3) at funcargs.c:82.

Получив эту строку, expect проверит корректность названия функции, имен и значений аргументов.

В теле функций, подобных call0a, содержатся инструкции изменения аргументов. Поскольку точка останова ставится на начало функции, то ни одна инструкция из тела функции не должна быть выполнена. После проверки названия функции и названия аргументов происходит тестирование значений аргументов при помощи команды print.

Описанный тест проводится для нескольких функций, похожих на call0a, разница заключается в количестве аргументов функции и в том, какой тип данных они имеют.

Тестирование backtrace и breakpoint

funcargs.exp содержит ряд тестов для тестирования команды backtrace. В данных тестах есть функции f1,f2,...,fn. Функция f1

вызывает f2, f2 вызывает f3, и так далее, пока не будет вызвана fn.

На каждую функцию устанавливается точка останова, после каждого останова осуществляется печать стека вызовов (команда backtrace). При помощи expect проверяется, что представленные функции в стеке вызовов имеют правильные имена, и что каждая функция отобразилась с правильными (заранее известными) аргументами.

Также для стека вызовов есть тест с рекурсивной функцией.

Другой вид теста представляет из себя останов в функциях с пустым телом. Например,

void marker_indirect_call () {}.

В рамках данного теста на функции такого рода ставятся точки остановка, и затем проверяется, что исполнение программы было остановлено в нужной функции.

3.3. Запуск тестов GDB

Для запуска тестов необходимо перейти в каталог gdb/testsuite/, который является подкаталогом каталога, где происходила сборка отладчика.

Для запуска всех тестов необходимо выполнить команду make check.

Для запуска конкретного теста используется команда следующего вида:

make check RUNTESTFLAGS="gdb.base/funcargs.exp",

где в кавычках указывается относительный путь до файла со сценарием на expect.

В результате выполнения теста генерируется два файла gdb.log и gdb.sum. Файл gdb.sum имеет вид:

PASS: gdb.base/funcargs.exp: set print frame-arguments all
PASS: gdb.base/funcargs.exp: run to call0a
PASS: gdb.base/funcargs.exp: print c after run to call0a
PASS: gdb.base/funcargs.exp: print s after run to call0a
PASS: gdb.base/funcargs.exp: print i after run to call0a
PASS: gdb.base/funcargs.exp: print l after run to call0a
PASS: gdb.base/funcargs.exp: continue to call0b
...

В данном файле представлены результаты тестирования: какие тесты были выполнены и вердикт выполнения теста.

Файл gdb.log содержит полный протокол действий отладчика.

4. Использование DejaGnu для тестирования GCC

Тестирование компилятора GCC [5], как и тестирование отладчика GDB осуществляется при помощи фреймворка dejaGnu. Тесты для GCC распространяются вместе с его исходными кодами и располагаются в каталоге gcc/testsuite.

4.1. Структура каталогов тестовой системы GCC

Каталог с тестами обладает следующей структурой.

estsuite/gcc.dg
Данный каталог содержит тесты компилятора языка Си. Тесты данного каталога проверяют различные возможности компилятора. Данный каталог содержит файлы на языке Си, которые будут компилироваться. Причем запуск скомпилированных программ не производится. При компиляции GCC может выводить сообщения: предупреждения (warning) и ошибки (error). Каждое сообщение сопоставляется шаблону. В случае, если сопоставления шаблону не произошло, то будет вызвана функция fail.

testsuite/gcc.dg/compat
Тесты этого подкаталога проверяют бинарную совместимость (binary compability) при помощи lib/compat.exp.

testsuite/gcc.dg/cpp
Тесты данного подкаталога проверяют работоспособность препроцессора.

testsuite/gcc.dg/debug
Подкаталог содержит тесты, предназначенные для проверки поддержки различных форматов отладочной информации (например, dwarf).

testsuite/gcc.dg/noncompile
Этот подкаталог содержит тесты с кодом, который не должен компилироваться. Компиляция осуществляется с различными ключами оптимизации, покрывая различные варианты использования GCC.

testsuite/gcc.c-torture
Это так называемые регрессионные тесты. Компиляция исходных файлов здесь осуществляется с различными ключами оптимизации, поскольку некорректная компиляция может иметь место только при наличии некоторых ключей, ответственных за оптимизацию. Данный каталог также

содержит тесты, направленные на проверку алгоритмов оптимизации.

testsuite/gcc.c-torture/compile
Каталог содержит программы, которые должны быть скомпилированы. Предупреждения, выводимые компилятором, игнорируются. Если программа не компилируется, то соответствующий тест считается проваленным.
testsuite/gcc.c-torture/execute
Этот каталог содержит программы, которые должны быть скомпилированы, слинкованы и выполнены в тестовой целевой среде.

testsuite/gcc.c-torture/execute/ieee
Тесты, направленные на проверку плавающей арифметики IEEE.

4.2. Названия тестов

Программы, предназначенные для тестирования компилятора, могут быть названы несколькими способами. Если две тестовые программы выполняют схожие действия, то они могут быть названы схожим образом, например: filename-1.c, filename-2.c. Отличие в названиях содержится в суффиксах -n.c.

Если тестовая программа предназначена для проверки определенной возможности (feature) компилятора, то файл с программой носит название feature-1.c.
Если данная программа представляет программу, при компиляции которой было замечено проявление ошибки компилятора, которая зарегистрирована в базе данных ошибок GCC, то файл с данной программой носит имя prbug-number-1.c, где bug-number является номером ошибки в базе данных ошибок.

Если файл представляет программу, при компиляции которой была замечена ошибка компилятора, но данная ошибка не зарегистрирована в базе данных ошибок GCC, то файл с такой программой получает имя в виде date.c, где date является датой, когда данная программа была добавлена в систему тестов.

4.3. Принципы построения тестов

В системе тестов gcc наибольшее число тестов представлены файлом с программой на языке Си.

Существуют тесты, которые компилируются и выполняются.

Если в результате выполнения произошла ошибка, то тестируемая программа завершается при помощи вызова функции abort().

Также существуют файлы с тестовыми программами и в комментариях к этим программам даны специальные директивы для тестового фреймворка. Директивы такого рода начинаются с префикса dg-.

4.4. Запуск тестов

Для запуска тестов необходимо перейти в каталог, где осуществлялась сборка, после чего вызвать команду make check.

5. Заключение

При тестировании сложных программ, таких как GDB или GCC, необходимо иметь постоянно пополняемый набор тестов, осуществляющий проверку новой функциональности, а также регрессивное тестирование при переходе на новую версию программы.

Фреймворк Dejagnu позволяет производить автоматическое тестирование программ непосредственно сразу после сборки.

Дальнейшее развитие этого фреймворка видится в реализации механизмов непрерывной интеграции и доставки.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований по теме (проекту)

«38. Проблемы создания глобальных и интегрированных информационно-телекоммуникационных систем и сетей, развитие технологий и стандартов GRID. Разработка средств распараллеливания программ для отечественных многоядерных микропроцессоров (0065-2018-0010)».

Software development tools testing

V.A.Galatenko, K.A.Kostiukhin, M.D.Dzabraev

Abstract: This article is devoted to a review of the automated testing tool DejaGnu. This tool is used in various GNU projects such as GDB and GCC. Also this work discuss tests of GCC and GDB, which used in SRISA RAS for testing the quality of these software products.

Keywords: automated testing, gcc, gdb, compiler, debugger.

Литература

1. Tcl Tutorial Clif Flynt.Version for TCL 8.5 available at ww.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html.
2. Exploring Expect: A Tcl-Based Toolkit for Automating Interactive Programs Don Libes. O'Reilly and Associates 1995, 1-56592-090-2.
3. https://www.gnu.org/software/dejagnu/manual/
4. http://www.embecosm.com/appnotes/ean8/ean8-howto-dejagnu-1.0.html#sec_example
5. https://gcc.gnu.org/onlinedocs/gccint/Testsuites.html#Testsuites

Составление многопроцессорных расписаний с ограничениями на связи между процессорами

М.Г. Фуругян

ВЦ ФИЦ ИУ РАН, Москва, Россия, E-mail: rtsccas@yandex.ru

Аннотация: Решается задача составления многопроцессорного расписания с прерываниями и ограничениями на связи между процессорами. Кроме того, учитываются затраты на обработку прерываний и переключений с одного процессора на другой. Разработаны алгоритмы, основанные на сведении исходной задачи к многопродуктовой потоковой задаче и целочисленной системе линейных ограничений.

Ключевые слова: многопроцессорная система, допустимое расписание с прерываниями, директивный интервал, многопродуктовый поток.

.

1. Введение

Одна из основных задач, решаемых при разработке математического обеспечения для систем проектирования и эксплуатации сложных технических объектов заключается в планировании вычислений, т.е. построении расписания выполнения программных модулей. Задачам построения расписаний с прерываниями без учета ограничений на связи между процессорами и затрат на обработку прерываний и переключений с одного процессора на другой посвящено большое число работ. Отметим такие работы, как [1] (идентичные процессоры), [2] (произвольные процессоры, одинаковые директивные интервалы), [3, 4] (произвольные процессоры, произвольные директивные интервалы). В [5] рассматривалась задача построения многопроцессорного расписания в системе с произвольными связями между процессорами и учетом затрат на обработку прерываний и переключений. В [6] получены условия, при выполнении которых существует полиномиальный алгоритм нахождения допустимого расписания в многопроцессорной системе с неполным графом связей между процессорами. В настоящей статье, в отличие от [7], рассматривается задача построения многопроцессорного допустимого расписания с ограничениями на количество прерываний и переключений. Эта задача сводится к потоковой задаче и целочисленной системе линейных ограничений.

2. Постановка задачи

Рассматривается вычислительная система, состоящая из m процессоров. Производительность j-го процессора равна s_j, $j = \overline{1,m}$. Имеется множество работ (заданий) $N=\{\overline{1,n}\}$, подлежащих выполнению. Каждое задание $i \in N$ характеризуется своим директивным интервалом $[b_i, f_i]$ (работа i может быть начата не ранее момента времени b_i и должна быть завершена не позднее момента времени f_i) и объемом Q_i работы процессоров, необходимым для его выполнения. По аналогии с [7], работа процессоров представляется как набор последовательных тактов. Такты всех процессоров синхронизованы во времени, т.е. начала и концы соответствующих тактов процессоров совпадают. Один такт является единицей времени, величины b_i и f_i, $i \in N$, задаются в тактах, которые занумерованы от 1 до T, где $\min\limits_{i\in N} b_i = 1$, $\max\limits_{i\in N} f_i = T$. Таким образом, работа i может быть начата не ранее начала b_i-го такта и должна быть завершена не позднее f_i-го такта. Производительность s_j – это объем работы процессора j за один такт. Для того, чтобы полностью выполнить работу i, процессору j требуется $\lceil Q_i / s_j \rceil$ тактов. На фиксированном такте каждый процессор может выполнять не более одной работы, а каждая работа выполняется не более чем одним процессором. При выполнении заданий допускаются прерывания и переключения с одного процессора на другой. Связи между процессорами могут изменяться во времени и, в отличие от [7], задаются следующим образом. Для каждого такта k, $1 \le k \le T-1$ определено множество $P_1(k)$ процессоров, для ко-

торых в конце данного такта допускаются прерывания и переключения на другие процессоры, а также номера $H(k)$ тактов, на которых можно возобновить прерываемую или переключаемую работу. Число работ, выполнение которых в конце такта k переключается на другой процессор или прерывается для возобновления на последующих тактах, не должно превосходить заданной величины $p_1(k)$. Отметим, что если на тактах k и $k+1$, $1 \le k \le T-1$, некоторая работа выполнялась одним и тем же процессором, то прерывания и переключения этой работы в конце такта k нет. Далее, для каждого такта $k_2 \in H(k_1)$, $1 \le k_1 \le T-1$, определено множество $P_2(k_2)$ процессоров, на которые допускается переключение или возобновление выполнения прерванной ранее работы. Число таких работ для такта k_2, $2 \le k_2 \le T$, не должно превышать заданной величины $p_2(k_2)$. Прерывание выполнения задания на процессоре j_1 в конце такта k_1 и его переключение на другой процессор требует дополнительной работы процессора j_1 в объеме $\tau_1(j_1, k_1)$. Возобновление прерванной работы на такте k_2 процессором j_2, а также переключение на этот процессор выполнения задания с другого процессора требует дополнительной работы процессора j_2 в объеме $\tau_2(j_2, k_2)$.

Требуется определить, существует ли допустимое расписание выполнения работ N (т.е. такое расписание, при котором каждая работа полностью выполняется в своем директивном интервале $[b_i, f_i]$), и найти его, если оно существует.

3. *NP*-трудность задачи

Покажем, что сформулированная задача является *NP*-трудной. Для этого достаточно показать, что исходная задача в форме распознавания свойств (когда требуется только определить, существует или не существует допустимое расписание) является *NP*-полной. Действительно, легко показать, что эта задача принадлежит классу *NP*. Кроме того, к ней полиномиально сводится известная *NP*-полная задача о разбиении [8, 9] (задано множество натуральных чисел $a_1, a_2, ..., a_n$; можно ли разбить его на два непересекающихся подмножества с одинаковой суммой?). Для доказательства этого факта в исходной задаче достаточно положить $N=\{\overline{1,n}\}$, $m=2$, $s_1 = s_2 = 1$, $b_i = 1,$, $f_i = T$, $Q_i = a_i$, $i \in N$, $P_1(k) = \{1, 2\}$, $p_1(k) = 2$, $k = \overline{1,\ T-1}$, $H(k) = \{k+1, ..., T\}$, $k = \overline{1,\ T-1}$, $P_2(k) = \{1, 2\}$, $p_2(k) = 2$, $k = \overline{2,\ T}$, $T = \left[\left(\sum_{i \in N} a_i\right)/2\right]$ и $\tau_1(j_1, k_1) = \tau_2(j_2, k_2) = 1$ для всех j_1, k_1, j_2, k_2. В этом случае прерывания и переключения приведут к тому, что не все работы успеют завершиться к окончанию T-го такта. Поэтому в задаче о разбиении ответ будет положительным тогда и только тогда, когда положительный ответ имеет рассматриваемая задача.

4. Допустимое расписание

Будем задавать допустимое расписание выполнения работ N в виде n списков $A(i)$, $i \in N$, где

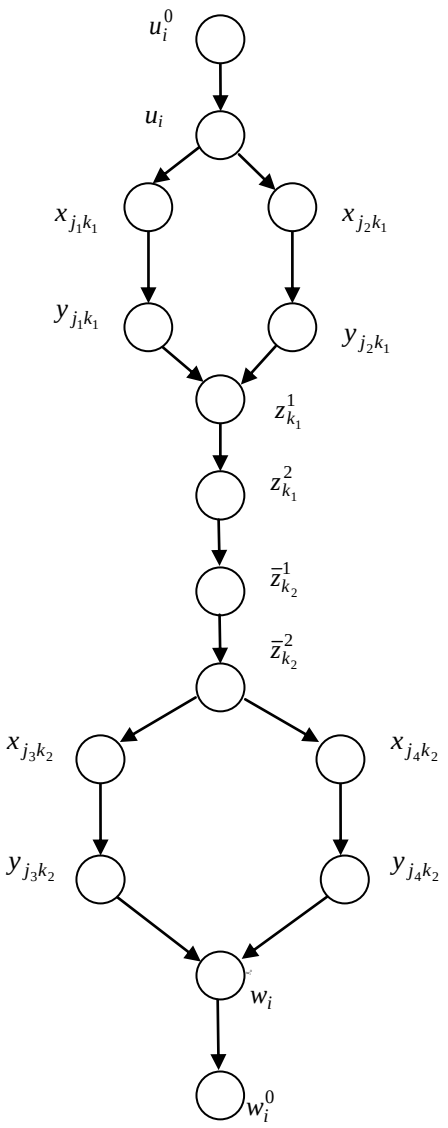
$$A(i) = \{(j_1, k_1), (j_2, k_2), ..., (j_h, k_h)\},\ 1 \le j_1, j_2, ..., j_h \le m,\ 1 \le k_1 < k_2 < ... < k_h \le T, \quad (1)$$

где $k_t \in [b_i, f_i]$, $t \in \overline{1,h}$. Наличие пары (j, k) в списке $A(i)$ означает, что работа i на такте k выполняется процессором j. Очевидно, что если для двух работ i_1, $i_2 \in N$, $i_1 \ne i_2$, $(j, k) \in A(i_1)$, $(j', k) \in A(i_2)$, то $j \ne j'$, поскольку на одном такте один процессор может выполнять не более одной работы. Кроме того, если $(j_t, k_t) \in A(i)$ и $(j_{t+1}, k_{t+1}) \in A(i)$, $1 \le t < h$ – две последовательные пары в $A(i)$, то существует соответствующая связь между процессорами j_t и j_{t+1}, т.е если на такте k_t происходит прерывание или переключение работы i на другой процессор, то $j_t \in P_1(k_t)$, $j_t \in P_1(k_t)$, $k_{t+1} \in H(k_t)$, $j_{t+1} \in P_2(k_{t+1})$.

5. Построение сетевой модели

Для решения поставленной задачи по аналогии с тем, как это сделано в [5, 7], построим сеть $G = (V, E)$, где V – множество узлов, E – множество дуг (см. рисунок).

Множество V состоит из следующих вершин: u_1^0, $u_2^0, ..., u_n^0$ – источники, w_1^0, $w_2^0, ...,$ w_n^0 – стоки, u_i, w_i, $i \in N$, $z_{k_1}^1$, $z_{k_1}^2$, $k_1 \in \overline{1,\ T-1}$, $\bar{z}_{k_2}^1$, $\bar{z}_{k_2}^2$, $k_2 \in \overline{2,\ T}$, x_{jk}, y_{jk}, $j = \overline{1,m}$, $k = \overline{1,\ T}$ – внутренние вершины. Множество E состоит из следующих дуг: (u_i^0, u_i), $i \in N$, (u_i, x_{jk}), $i \in N$, $j = \overline{1,m}$, $k \in [b_i, f_i]$; (x_{jk}, y_{jk}), $j = \overline{1,m}$, $k = \overline{1,T}$;



$k_1,\ k_2 \in [b_i,\ f_i],\quad k_2 \in H(k_1)$
$j_1,\ j_2 \in P_1(k_1),\quad j_3,\ j_4 \in P_2(k_2)$

Рисунок. Фрагмент сети G

$(\bar{z}_{k_2}^2, x_{j_2k_2})$, $k_1 \in \overline{1,\ T-1}$, $k_2 \in H(k_1)$, $j_1 \in P_1(k_1)$, $j_2 \in P_2(k_2)$, (y_{jk}, w_i), $i \in N$, $j = \overline{1,m}$, $k \in [b_i,\ f_i]$, (w_i, w_i^0), $i \in N$, Узел u_i соответствует началу, а w_i – окончанию выполнения работы i. Узел x_{jk} соответствует процессору j и началу k-го такта, а узел y_{jk} – процессору j и окончанию k-го такта. Наличие в сети G_1 дуги (u_i, x_{jk}) означает возможность начать выполнение работы i в начале k-го такта процессором j, а наличие дуги (y_{jk}, w_i) означает возможность завершить выполнение работы i в конце k-го такта процессором j.

Дуга (x_{jk}, y_{jk}) соответствует выполнению некоторого задания процессором j на k-м такте. Цепочка дуг $(y_{j_1k_1}, z_{k_1}^1)$, $(z_{k_1}^1, z_{k_1}^2)$, $(z_{k_1}^2, \bar{z}_{k_2}^1)$, $(\bar{z}_{k_2}^1, \bar{z}_{k_2}^2)$, $(\bar{z}_{k_2}^2, x_{j_2k_2})$ означает возможность переключения выполнения некоторой работы с процессора j_1 на такте k_1 на процессор j_2 на такте k_2. Отметим, что $|V| = O(n + mT + T^2)$, $|E| = O(nmT + m^2T^2)$.

6. Необходимые и достаточные условия существования допустимого расписания

Сформулируем исходную задачу в виде потоковой. В сети G_1 будем рассматривать целочисленный n-продуктовый поток. Узлы u_i^0 и w_i^0 – соответственно источник и сток i-го продукта. Каждая дуга $(a, b) \in E$ имеет три параметра: нижнюю $L(a, b)$ и верхнюю $U(a, b)$ границы потока, а также стоимость $C(a, b)$ единицы потока. Значения этих параметров для дуг сети G указаны в таблице.

Таблица. Значения параметров дуг сети G

Дуги	L	U	C
(u_i^0, u_i)	1	1	0
(u_i, x_{jk})	0	1	0
(x_{jk}, y_{jk})	0	1	s_j
$(y_{jk}, z_{k_1}^1)$	0	1	$-\tau_1(j, k_1)$
$(z_{k_1}^1, z_{k_1}^2)$	0	$p_1(k_1)$	0
$(z_{k_1}^2, \bar{z}_{k_2}^1)$	0	N	0
$(\bar{z}_{k_2}^1, \bar{z}_{k_2}^2)$	0	$p_2(k_2)$	0
$(\bar{z}_{k_2}^2, x_{jk})$	0	1	$-\tau_2(j, k_2)$
(y_{jk}, w_i)	0	1	0
(w_i, w_i^0)	1	1	0

Пусть g – целочисленный n-продуктовый поток в сети G и пусть g_i – поток i-го продукта, $C(g_i) = \sum_{(a,b) \in E_1} C(a,b)\, g_i(a,b)$ – его стоимость. Отметим, что поскольку поток g целочисленный (т.е. $g_i(a,b)$ – целая величина для всех дуг $(a, b) \in E$), то в силу определения нижних и верхних границ потоков по дугам (см. таблицу) величины $g_i(a,b)$ и $g(a,b)$ для каждой дуги $(a, b) \in E_1$, кроме $(z_{k_1}^1, z_{k_1}^2)$,

$(z^2_{k_1}, \bar{z}^1_{k_2})$, $(\bar{z}^1_{k_1}, \bar{z}^2_{k_2})$, принимают значения 0 или 1.

Л е м м а . Для существования допустимого расписания необходимо и достаточно существования целочисленного n-продуктового потока g в сети G, для которого u_i^0 и w_i^0 – соответственно источник и сток i-го продукта и

$$C(g_i) \geq Q_i \qquad (2)$$

при всех $i \in N$.

Д о к а з а т е л ь с т в о. Необходимость. Предположим, что существует допустимое расписание A выполнения работ N. Согласно (1) $A(i) = \{(j_1, k_1), (j_2, k_2), ..., (j_h, k_h)\}$. Определим поток g по следующим правилам:

1) $g_i(u_i^0, u_i) = g_i(w_i^0, w_i) = 1$;

2) $g_i(u_i, x_{j_1 k_1}) = g_i(y_{j_h k_h}, w_i) = 1$;

3) если $(j_t, k_t) \in A(i)$, то

$g_i(x_{j_t k_t}, y_{j_t k_t}) = 1$;

4) если $(j_t, k_t) \in A(i)$, $(j_{t+1}, k_{t+1}) \in A(i)$ и в конце такта k_t было прерывание выполнения работы i или переключение с процессора j_t на процессор j_{t+1} и ее возобновление в начале такта k_{t+1}, то полагаем $g_i(y_{j_t k_t}, z^1_{k_t}) =$

$= g_i(z^1_{k_t}, z^2_{k_t}) = g_i(z^2_{k_t}, \bar{z}^1_{k_{t+1}}) = g_i(\bar{z}^1_{k_{t+1}}, \bar{z}^2_{k_{t+1}}) =$

$= g_i(\bar{z}^2_{k_{t+1}}, x_{j_{t+1} k_{t+1}}) = 1$

Поток i-го продукта по всем остальным дугам $(a, b) \in E_1$ сети G полагаем равным нулю: $g_i(a, b) = 0$. Согласно правилам 1) – 4) единица потока i-го продукта исходит из узла u_i^0, проходит по дугам (u_i^0, u_i), $(u_i, x_{j_1 k_1})$, далее, по дугам (x_{jk}, y_{jk}), соответствующим тактам и процессорам, на которых задание i выполняется, по дугам $(y_{j_1 k_1}, z^1_{k_1})$, $(z^1_{k_1}, z^2_{k_1})$, $(z^2_{k_1}, \bar{z}^1_{k_2})$, $(\bar{z}^1_{k_2}, \bar{z}^2_{k_2})$, $(\bar{z}^2_{k_2}, x_{j_2 k_2})$, соответствующим прерываниям и переключениям при его выполнении (если $j_1 = j_2$, то переключения нет), и дугам $(y_{j_h k_h}, w_i)$ и (w_i^0, w_i). Определенный таким образом поток является целочисленным, для каждого продукта выполнены условия сохранения во внутренних узлах сети, источником и истоком i-го продукта являются узлы u_i^0 и w_i^0 соответственно. Ограничения сверху и снизу на величины потоков g_i по дугам выполнены, поскольку g_i принимает значения 0 или 1 и один процессор на одном такте не может выполнять более одной работы. Кроме того, расписание удовлетворяет ограничениям на число прерываний и переключений и на число возобновляемых работ на каждом такте, что обеспечивается ограничениями на пропускную способность дуг $(z^1_{k_1}, z^2_{k_1})$ и $(\bar{z}^1_{k_1}, \bar{z}^2_{k_2})$. Также выполнено неравенство (2) для каждого $i \in N$. Действительно, единица потока по дуге (x_{jk}, y_{jk}) увеличивает величину $C(g_i)$ на s_j, а единица потока по дугам $(y_{j_1 k_1}, z^1_{k_1})$ и $(\bar{z}^2_{k_2}, x_{j_2 k_2})$ уменьшает величину $C(g_i)$ на $\tau_1(j, k_1)$ и $\tau_2(j, k_2)$ соответственно. Поскольку согласно допустимому расписанию суммарная величина работы процессоров по выполнению задания i не меньше Q_i, то неравенство (2) выполняется при всех $i \in N$.

Достаточность. Пусть в сети G существует целочисленный n-продуктовый поток, для которого i-й продукт исходит из источника u_i^0, входит в сток w_i^0 и выполнено неравенство (2) для всех $i \in N$. Поскольку $L(u_i^0, u_i) = U(u_i^0, u_i) = 1$, то $g_i(u_i^0, u_i) = 1$. Аналогично, $g_i(w_i, w_i^0) = 1$. Поскольку поток g_i целочисленный, то его величина по каждой дуге сети G равна 0 или 1. В силу сохранения потока g_i в каждом внутреннем узле существует единственный путь Π_i из u_i^0 в w_i^0, величина потока g_i по которому равна 1. Из (2) и структуры сети G следует, что путь Π_i задает допустимое расписание для работы i. Из ограничений на пропускную способность каждой дуги следует, что пути $\Pi_1, \Pi_2, ..., \Pi_n$ попарно не пересекаются по дугам (x_{jk}, y_{jk}). Следовательно, эта совокупность путей определяет допустимое расписание выполнения работ N. А именно, если $(x_{jk}, y_{jk}) \in \Pi_i$, то работа i выполняется на k-м такте процессором j; если дуги $(y_{j_1 k_1}, z^1_{k_1})$, $(z^1_{k_1}, z^2_{k_1})$, $(z^2_{k_1}, \bar{z}^1_{k_2})$, $(\bar{z}^1_{k_2}, \bar{z}^2_{k_2})$, $(\bar{z}^2_{k_2}, x_{j_2 k_2}) \in \Pi_i$, то в конце такта k_1 происходит переключение работы i с процессора j_1 на процессор j_2, на котором она возобновляет выполнение в начале такта k_2 (если $j_1 = j_2$, то переключения не происходит). Ограничения на пропускную способность дуг $(z^1_{k_1}, z^2_{k_1})$ и $(\bar{z}^1_{k_1}, \bar{z}^2_{k_2})$ обеспечивают выполнение условий на число прерываемых и переключаемых работ, а также на число возобновляемых работ на каждом такте. Лемма доказана.

Таким образом, для построения допустимого расписания выполнения работ N следует

найти в сети G многопродуктовый поток, удовлетворяющий условиям леммы, и, если он существует, построить расписание так, как это описано при доказательстве достаточности. Если же такого потока не существует, то и допустимого расписания не существует.

Опишем теперь необходимые и достаточные условия существования допустимого расписания, сформулированные в лемме1, в виде целочисленной системы линейных ограничений. Найти неотрицательные целочисленные величины $g_i(u_i^0, u_i)$, $g_i(u_i, x_{jk})$, $g_i(x_{jk}, y_{jk})$, $g_i(y_{j_1k_1}, z_{k_1}^1)$, $g_i(z_{k_1}^1, z_{k_1}^2)$, $g_i(z_{k_1}^2, \bar{z}_{k_2}^1)$, $g_i(\bar{z}_{k_2}^1, \bar{z}_{k_2}^2)$, $g_i(\bar{z}_{k_2}^2, x_{j_2k_2})$, $g_i(y_{jk}, w_i)$, $g_i(w_i, w_i^0)$, $1 \le k_1 < k_2 \le T$, $1 \le k \le T$, $k_2 \in H(k_1)$, $j_1 \in P_1(k_1)$, $j_2 \in P_2(k_2)$, $(u_i, x_{jk}) \in E_1$, $(y_{jk}, w_i) \in E$, удовлетворяющие ограничениям:

$$g_i(u_i^0, u_i) = 1,\ g_i(w_i, w_i^0) = 1,\ i \in N, \qquad (3)$$

$$g_i(u_i, x_{jk}) \le 1,\ g_i(y_{jk}, w_i) \le 1,\ i \in N,\ j = \overline{1,m},\ k \in [b_i,\ f_i], \qquad (4)$$

$$\sum_{i \in N} g_i(x_{jk}, y_{jk}) \le 1,\ j = \overline{1,m},\ k = \overline{1,T}, \qquad (5)$$

$$\sum_{i \in N} g_i(y_{j_1k_1}, z_{k_1}^1) \le 1,\quad j = \overline{1,m},\quad k_1 = \overline{1,T-1}, \qquad (6)$$

$$\sum_{i \in N} g_i(z_k^1,\ z_k^2) \le p_1(k),\qquad k = \overline{1,T-1}, \qquad (7)$$

$$\sum_{i \in N} g_i(z_{k_1}^2,\ \bar{z}_{k_2}^1) \le n,\quad 1 \le k_1 < k_2 \le T, \qquad (8)$$

$$\sum_{i \in N} g_i(\bar{z}_k^1,\ \bar{z}_k^2) \le p_2(k),\ k = \overline{2,T}, \qquad (9)$$

$$\sum_{i \in N} g_i(\bar{z}_k^2,\ x_{j\,k})) \le 1,\ j = \overline{1,m},\ k = \overline{2,T}, \qquad (10)$$

$$g_i(u_i^0, u_i) = \sum_{\substack{j=\overline{1,m},\\ k \in [b_i,\ f_i]}} g_i(u_i, x_{jk}),\quad i \in N, \qquad (11)$$

$$\sum_{\substack{j=\overline{1,m},\\ k \in [b_i,\ f_i]}} g_i(y_{jk}, w_i) = g_i(w_i, w_i^0),\ i \in N, \qquad (12)$$

$$g_i(u_i, x_{j_2k_2}) + g_i(\bar{z}_{k_2}^2,\ x_{j_2k_2}) = g_i(x_{j_2k_2}, y_{j_2k_2}),\quad i \in N,\quad j_2 \in P_2(k_2),\ k_2 \in [b_i,\ f_i], \qquad (13)$$

$$g_i(x_{j_1k_1}, y_{j_1k_1}) = g_i(y_{j_1k_1}, z_{k_1}^1) + g_i(y_{j_1k_1}, w_i),\quad i \in N,\ j_1 \in P_1(k_1),\ k_1 \in [b_i,\ f_i], \qquad (14)$$

$$\sum_{j \in P_1(k)} g_i(y_{jk}, z_k^1) = g_i(z_k^1,\ z_k^2),\quad i \in N,\ k = \overline{1,T-1}, \qquad (15)$$

$$g_i(z_{k_1}^1,\ z_{k_1}^2) = \sum_{k_2 \in H(k_1)} g_i(z_{k_1}^2,\ \bar{z}_{k_2}^1),\quad i \in N,\ k_1 = \overline{1,T-1}, \qquad (16)$$

$$\sum_{k_1 = \overline{1,T-1}} g_i(z_{k_1}^2,\ \bar{z}_{k_2}^1) = g_i(\bar{z}_{k_2}^1,\ \bar{z}_{k_2}^2),\quad i \in N,\quad k_2 \in H(k_1), \qquad (17)$$

$$g_i(\bar{z}_k^1,\ \bar{z}_k^2) = \sum_{j \in P_2(k)} g_i(\bar{z}_k^2,\ x_{jk}),\quad i \in N,\quad +k = \overline{2,T}, \qquad (18)$$

$$\sum_{j=1}^{m}\sum_{k=1}^{T} g_i(x_{jk}, y_{jk})\, s_j - \sum_{k_1=1}^{T-1}\sum_{j_1 \in P_1(k_1)} g_i(y_{j_1k_1}, z_{k_1}^1)\, \tau_1(j_1, k_1) - \sum_{k_2=2}^{T}\sum_{j_2 \in P_2(k_2)} g_i(\bar{z}_{k_2}^2,\ x_{j_2k_2})\, \tau_2(j_2, k_2) \ge Q_i,\quad i \in N. \qquad (19)$$

Равенства (3) обеспечивают доставку единицы потока i-го продукта из источника u_i^0 в сток w_i^0. Неравенства (4) – (10) обеспечивают выполнение ограничений сверху на поток по дугам, равенства (11) – (18) обеспечивают сохранение потоков каждого продукта во всех внутренних узлах сети G_1, а неравенства (19) эквивалентны неравенствам (2). Отметим, что в системе (3) – (19) $O(nmT + m^2T^2)$ переменных и $O(nmT + m^2T^2)$ линейных ограничений.

Calculation planning in multiprocessor system with cost accounting for interrupts

M.G. Furugyan

Abstract. The problem of deadline scheduling in multiprocessor system with identical directive intervals and with cost accounting for interrupts is studied. The algorithm of construction of feasible schedule was elaborated. This algorithm is pseudopolynomial and has linear complexity depending on number of jobs when number of processors is fixed.

Keywords: multiprocessing system, feasible schedule, directive interval, multicommodity network flow.

ЛИТЕРАТУРА

1. В.С. Танаев, В.С. Гордон, Я.М. Шафранский. Теория расписаний. Одностадийные системы. М., Наука, 1984.

2. T. Gonzales, S. Sahni. Preemptive Scheduling of Uniform Processor Systems. «J. Association for Computing Machinery», v. 25 (1978), № 1, 92 – 101.

3. A. Federgruen, H. Groenevel. Preemptive Scheduling of Uniform Machines by Ordinary Network Flow Technique. «Management Science», v. 32 (1986), № 3, 341 – 349.

4. P. Brucker. Scheduling Algorithms. Heidelberg, Springer, (2007), 371.

5. Д.С. Гуз, М.Г. Фуругян. Планирование вычислений в многопроцессорных АСУ реального времени с ограничениями на память процессоров. «АиТ», (2005), №2, 138-147.

6. Б.В. Гречук, М.Г. Фуругян. Составление оптимальных расписаний с прерываниями в многопроцессорных системах с неполным графом связей. «Кибернетика и системный анализ», (2005), № 3, 94 – 102.

7. М.Г. Фуругян. Составление расписаний в многопроцессорных системах с дополнительными ограничениями. «Изв. РАН, ТиСУ», (2017), № 2, 57 – 66.

8. Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. Алгоритмы. Построение и анализ. (Второе издание.). М.: Вильямс, 2005.

9. Б. Корте, Й. Фиген. Комбинаторная оптимизация. Теория и алгоритмы. М.: МЦНМО, 2015.

10. Д. Филлипс, А. Гарсиа-Диас. Методы анализа сетей. М.: Мир, 1984.

Разработка собственных управляющих конструкций в среде ДССП для троичной машины

А.А. Бурцев

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: burtsev@niisi.msk.ru

Аннотация: Системы программирования со словарной организацией (ФОРТ и ДССП) предоставляют пользователю-программисту уникальную возможность путём наращивания словаря сотворить по сути свой собственный язык программирования. При этом в качестве новых в словарь можно добавлять и такие слова, которые предназначены для управления ходом исполнения программы. В статье подробно рассматриваются приёмы разработки в среде ДССП-ТВМ (версии ДССП для троичной машины) особых процедур для таких слов (команд), которые обеспечивают функционирование добавляемых в язык ДССП своих собственных управляющих конструкций.

Ключевые слова: структурированное программирование, словарная организация, ФОРТ, ДССП, троичный компьютер, сшитый код, управляющие конструкции.

Введение

Системы программирования со словарной организацией, такие как ФОРТ[1] и ДССП[2,3], предоставляют пользователю (программисту) уникальную возможность путём наращивания словаря сотворить по сути свой собственный язык программирования. При конструировании лексики нового языка достаточно лишь соблюдать простые синтаксические правила базового языка. В языках ФОРТ и ДССП ключевой лексической единицей является слово. За небольшим исключением словом считается любая последовательность символов, отличных от пробела (при этом знаки табуляции и перехода на новую строку считаются эквивалентными пробелу), и путём последовательного сочленения (через пробел) имеющихся слов можно строить любые предложения языка. ДССП и ФОРТ являются интерпретируемыми системами, а всякое действие в них вызывается словом.

Программирование, как процесс созидания программного изделия, в этих системах заключается в расширении словаря, т.е. в наполнении первоначально предоставленного словаря новыми словами. Причём, наряду с обычными словами, обозначающими просто названия новых операций, в создаваемый словарь можно также добавлять и особые слова, которые далее будут сами использоваться при построении программы для объявлений в ней новых сущностей. Таких, как константы, переменные, строки, массивы, структуры, классы для определения новых типов данных вместе с процедурами-исполнителями их методов, сопрограммы, исключительные ситуации и т.п. Подобные особые слова, служащие для определения новых слов и управляющие формированием компонуемой программы, в ДССП и ФОРТе принято называть компилирующими словами или словами-компиляторами.

В обычных языках программирования для управления ходом исполнения программы, как правило, предлагается жёстко установленный ассортимент управляющих конструкций (операторов), для употребления которых в языке заранее закрепляется определённый набор ключевых слов и устанавливаются синтаксические правила их применения. И программист вынужден строить свою программу, ориентируясь лишь на базовые управляющие конструкции, которые были предусмотрены при создании используемого языка и его транслятора (компилятора и/или интерпретатора).

Как правило, в современных языках программирования предусматривается условный оператор **if then else** для организации ветвлений, операторы **while do** и **repeat until** – для организации циклов. Обычно также предусматривается какой-то вариант цикла со счётчиком с перебором всех возможных значений из целочисленного отрезка [a,b]: **for** i**:=** a **to** b **do**. В некоторых языках предлагаются ещё и разновидности операторов выбора для организации множественного ветвления: **case of** или **switch**.

Но каким бы совершенным ни был ассортимент управляющих конструкций, предусмотренный в языке на момент его создания, рано или поздно он может оказаться устаревшим. И история развития языков

программирования уже не раз это подтверждала.

Так, например, в конце 60-х годов XX века после публикаций работ Дейкстры [4, с.7-97] о важности соблюдения структурированного подхода при разработке программ не только в новые (Паскаль, Си), но и в уже существующие на тот момент языки программирования (Фортран, Бейсик) стали добавлять особые структурированные конструкции для организации ветвлений и циклов в программе [5]. Чуть позже в языки программирования стали внедрять и особые управляющие конструкции для структурированной обработки так называемых исключительных ситуаций [6].

Впоследствии в тех или иных языках программирования стали предлагаться и другие полезные управляющие конструкции. Например, команда троичного ветвления по знаку [7, с.29]: **<BR** `ProcN,Proc0,ProcP`**>;** оператор многовариантного выбора с множественными условиями [8 с. 56-59]:

```
if
  усл1 => действие1
  усл2 => действие2 . . .
  услN => действиеN
fi
```

оператор цикла с множественными условиями и телами [8, с.59-60]:

```
do
   усл1 => тело1
   усл2 => тело2  . . .
   услN => телоN
od
```

конструкция цикла для перебора всех значений множества: **for** x **in S do** [4, с.144]; а также так называемые итераторы – итерационные методы для перебора всех элементов обрабатываемой структуры данных (массива, линейного списка, дерева, коллекции) для совершения над каждым из них заданного действия [9, с. 458-460].

А сколько разнообразных управляющих конструкций [10-13] было предложено ввести в языки программирования, чтобы создавать программы, предусматривающие запуск и взаимодействие параллельных процессов!

Так что можно смело утверждать, что потребность встраивать новые управляющие конструкции в существующий язык программирования у программистов, его использующих, будет оставаться всегда. Вопрос в том, как же эту потребность можно удовлетворить?

Понятно, что при использовании обычного языка программирования программист не может сам добавлять в язык новую нужную ему управляющую конструкцию. Ведь для этого надо вносить изменения в компилятор (или интерпретатор) языка, исходный текст которого ему, как правило, недоступен.

А вот системы программирования со словарной организацией (ФОРТ и ДССП) такую возможность ему как раз таки предоставляют. Но чтобы реализовать такую возможность, программисту надо освоить технику разработки в этих системах особых процедур, осуществляющих доступ (через стек возвратов) к тем позициям кода, откуда эта особая процедура вызывалась. И научиться добавлять в словарь новые служебные (ключевые) слова, вызов которых приводит к исполнению этих процедур.

Далее в статье рассматриваются приёмы разработки подобных особых процедур в среде ДССП для троичной [виртуальной] машины (ТВМ) – ДССП-ТВМ [14,15,16]. В результате освоения таких приёмов программист – пользователь ДССП-ТВМ – получает возможность добавлять в язык ДССП новые управляющие конструкции, в том числе и свои собственные, т.е. разработанные им собственноручно на языке самой системы.

1. Базовые управляющие конструкции ДССП-ТВМ

В ДССП-ТВМ вызов слова эквивалентен вызову процедуры (или команды), предусмотренной для его исполнения. Для организации ветвлений и циклов в ДССП помимо обычного (безусловного) вызова слова предусмотрены команды для условного и повторяемого вызова слова, а также разнообразные команды выбора слова из нескольких в зависимости от условия.

В ДССП (и в ФОРТе) обработка данных осуществляется через арифметический стек. Поэтому в качестве условия в таких командах принято оценивать значение (**T**), сформированное в вершине стека: нулевое оно (=0), положительное (>0) или отрицательное (<0), а затем удалять его из стека непосредственно перед вызовом выбранного слова на исполнение.

В ДССП для одиночного ветвления предлагаются команды вызова слова по условию:

```
{T} IF- P {} { вызов слова P, если T <0 }
{T} IF0 P {} { вызов слова P, если T =0 }
{T} IF+ P {} { вызов слова P, если T >0 }
```

Для двоичного ветвления предлагаются команды вызова (выбора) одного из двух слов:

```
{ выбор: либо вызов слова P1, либо слова P2 }
{T} BR- P1 P2 {если T<0,то P1, иначе P2}
{T} BR0 P1 P2 {если T=0,то P1, иначе P2}
{T} BR+ P1 P2 {если T>0,то P1, иначе P2}
```

Команда троичного ветвления предлагает выбрать на исполнение одно из трёх слов в зависимости от знака вершины стека (**T**):

```
{T} BRS P- P0 P+ {если T<0,то вызов P-}
{если T=0,то вызов P0; если T>0,то вызов P+}
```

Управляющая конструкция для многовариантного выбора, аналогичная оператору **case** (языка Паскаль), представлена в ДССП с помощью двух команд **BR** и **ELSE** :

{V} **BR** V1 **P1** V2 **P2** … VX **PX ELSE P0**
{ вызов слова Pi (если V=Vi), либо P0 }

Заметим, что в качестве слов Vi здесь можно задавать не только слова-константы, но и любые другие слова, которые могут поместить в стек одно значение для его последующего сравнения с заданным значением V. Порядок исполнения этой нетривиальной управляющей конструкции поясняет блок-схема на Рис. 1.

{V}**BR** V1 **P1** V2 **P2** … VX **PX ELSE P0**

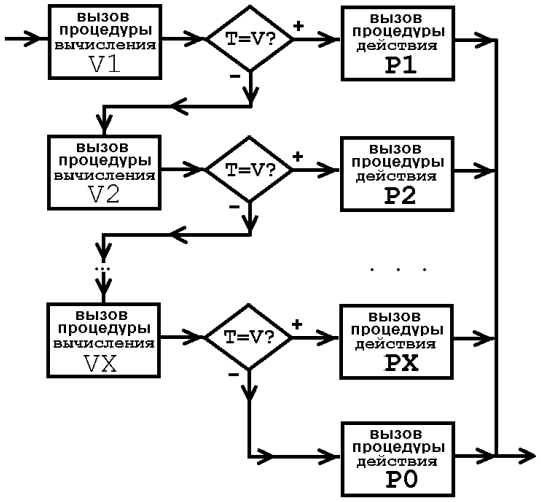


Рис. 1. Блок-схема конструкции **BR … ELSE**

Для организации циклов в ДССП предусмотрены команды **LOOP DO- DW** :

LOOP Body { многократно повторяемый вызов слова **Body** возможно бесконечное число раз }
{**n**} **DO- Body** **≈ for k=n-1 downto 0 do Body** **{** вызов слова **Body n** раз с убывающим значением параметра (в вершине стека) k=n-1,..,1,0 }
Cond {**T**} **DW Body** {} **≈ while Cond do Body** **{**многократный вызов слова **Body** (тела цикла), пока вызываемое слово **Cond** (условие цикла) помещает в вершину стека ненулевое значение**}**

Порядок исполнения цикла со счётчиком поясняет блок-схема команды **DO-** на Рис. 2., а цикла с условием — схема команды **DW** на Рис. 3.

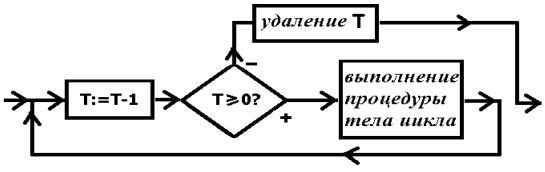


Рис. 2. Блок-схема команды цикла со счётчиком **DO-**

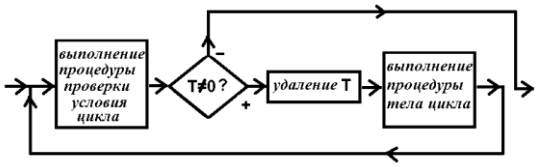


Рис. 3. Блок-схема команды цикла условием **DW**

Представленные здесь управляющие конструкции считаются базовыми, поскольку соответствующие им слова уже занесены в первоначальный словарь ДССП-ТВМ. В прежних версиях ДССП все эти управляющие команды приходилось реализовывать отдельными процедурами на языке ассемблера базовой машины. Но в версии ДССП-ТВМ почти все они теперь реализуются напрямую командами самой троичной машины (ТВМ). В настоящее время лишь команды **BR** и **ELSE** реализованы в виде процедур ассемблерного ядра ДССП.

Отметим важную особенность всех управляющих конструкций ДССП: тела ветвлений и циклов должны обязательно быть оформлены в виде отдельных слов, что стимулирует дробление программы на более мелкие процедуры. Поэтому ДССП характеризуется как система, поддерживающая процедурный вариант структурированного программирования.

Заметим, что впервые такой вариант структурированного программирования был обеспечен системой команд экспериментальной ЦМ «Сетунь-70» [7]. Можно образно сказать, что в «Сетуни-70» он был, таким образом, поддержан на уровне машинного кода. А в ДССП он поддерживается уже на уровне так называемого сшитого кода, который используется для внутреннего представления ДССП-программ.

2. Процедурный сшитый код для тел ДССП-процедур

Для реализации процедур, исполняющих слова-команды управляющих конструкций, необходимо ознакомиться, как в ДССП-ТВМ устроено внутреннее представление откомпилированных программ.

Обычно для такого представления в системах ДССП и ФОРТ используется особый вид кода, состоящий, в основном, из адресных ссылок на вызываемые при его исполнении процедуры. Такой вид кода получил давно устоявшееся название сшитый код (threaded code).

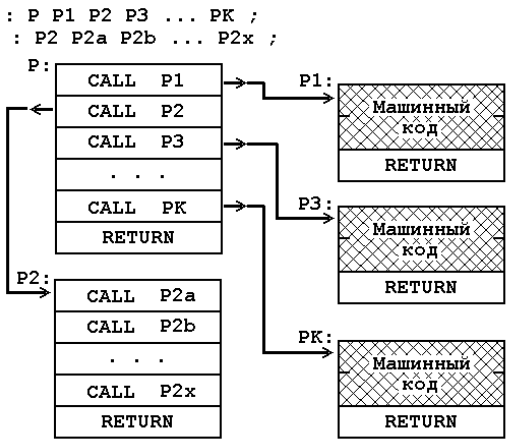


Рис. 4. Строение тел ДССП-процедур

В различных версиях ФОРТа и ДССП применялись разнообразные варианты сшитого кода: прямой; косвенный, знаково-косвенный, двойной косвенности [17]. В ДССП-ТВМ применяется вариант процедурного сшитого кода [18], который может напрямую исполняться троичной (виртуальной) машиной. Главная особенность такого кода заключается в том, что адреса процедур, содержащиеся в нём, воспринимаются троичной машиной как команды вызова процедур, расположенных по этим адресам в памяти (см. Рис. 4).

В ДССП при компоновке кода программы соблюдается так называемый принцип «1 в 1», согласно которому одному слову компилируемого предложения исходного текста должно соответствовать одно машинное слово формируемого машинного кода. Так что в ДССП-ТВМ каждому слову, помещаемому в словарь, назначается одно троичное (27-тритное) слово машинного кода, которое компилятор помещает в позицию формируемого тела компилируемой процедуры.

Если действие, вызываемое этим словом, может быть исполнено одной командой ТВМ, ему назначается слово с кодом этой команды. Иначе для исполнения слова формируется тело процедуры, состоящей уже из последовательности машинных команд, завершающейся командой возврата из процедуры. И в таком случае определяемому слову в словаре приписывается адрес расположения создаваемой для него процедуры, который при исполнении будет расшифровываться машиной как команда вызова процедуры с заданного адреса.

В виде исключения ДССП-компилятор особым образом обрабатывает слова, представляющие числовые и символьные константы (`'X'`), а также строки символов (`"x..x"`) и строки печатаемого текста (`."x..x"`). И при этом принцип “1 в 1” в настоящее время удаётся соблюдать лишь для констант, значение которых укладывается в два трайта (18 тритов). И для слова, изображающего такую константу **V**, формируется 27-тритный код команды засылки её значения в стек (DPushI_**V**), где само значение **V** записано в двух младших трайтах.

2.1. Средства доступа к телу сшитого кода через стек возвратов

Процедуры, реализующие действия слов, в ДССП не имеют явных параметров. Параметры для обработки вызванной процедуре передаются через арифметический стек. Входными параметрами для процедуры (слова) могут служить величины, помещённые в стек до её вызова, а выходными параметрами могут считаться величины, оставленные в стеке в результате её исполнения.

Но кроме этих величин процедуре может потребоваться доступ к тем позициям тела сшитого кода, которые располагаются вслед за командой, откуда эта процедура была вызвана. Именно такой доступ как раз и необходим тем особым процедурам, которые создаются для исполнения управляющих команд, и приёмы реализации которых являются предметом нашего дальнейшего обсуждения.

Схему доступа к этим позициям (см. Рис. 5) рассмотрим на примере такой команды, процедура исполнения которой принимает **m** параметров X1,X2 .., X**m** из арифметического стека и которой для обработки доступны также **n** слов в теле сшитого кода, стоящих вслед за командой её вызова.

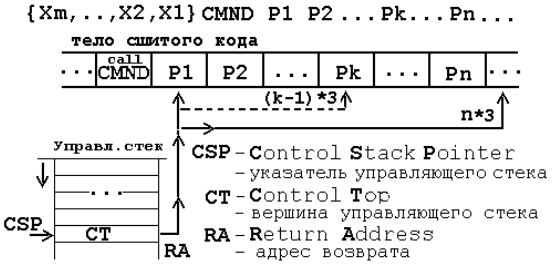


Рис. 5. Схема доступа к телу сшитого кода

Адрес возврата (RA) вызванной процедуры запоминается в вершине (CT) управляющего стека (CSP). Он указывает на позицию первого параметра в теле (P1), расположенного сразу вслед за позицией самой команды вызова процедуры (call CMND). Для взятия k-параметра (Pk) из тела необходимо сдвинуться по телу ещё на (k-1) позицию, для чего прибавить к адресу RA значение (k-1)*3, т.к. адресация ведётся с точностью до трайта, а одна позиция кода занимает троичное слово из 3-х трайтов.

После исполнения процедуры необходимо изменить адрес возврата в вершине управляющего стека так, чтобы вернуться из

процедуры на продолжение программы в позицию, расположенную за последним параметром (P**n**) всей управляющей конструкции, охватываемой этой командой. И для этого адрес возврата надо увеличить на значение **n***3.

Для доступа к вершине управляющего стека в ДССП-ТВМ предусмотрены две команды:

@R	взять значение вершины из стека возвратов и заслать его в стек данных
!R	взять значение вершины из стека данных и заслать его в стек возвратов

С помощью этих двух команд можно реализовать все те действия, которые схематично были представлены на Рис. 5 и которые могут потребоваться нам для разработки самых разнообразных управляющих команд в дальнейшем.

Чаще всего для взятия очередного параметра из тела вызова с продвижением указателя возврата на следующую позицию тела используется последовательность команд вида:

```
@R C 3+ !R @W
```

Поскольку такой ряд действий приходится исполнять достаточно часто, в ДССП для его исполнения выделено специальное слово **GTP** (которым мы далее и будем заменять этот ряд операций).

2.2. Приёмы передачи параметров процедуре из тела её вызова

Для начала рассмотрим пример реализации процедуры, которой из тела её вызова передаётся всего один параметр. Эта процедура должна печатать таблицу заданной двуместной троичной логической функции (операции):

{StrName,Len} **PrintT3LogFunc TFunc** { }

Для неё в стеке данных задаётся строка названия функции (адрес и длина), а в позиции следом по телу располагается команда или адрес процедуры, предназначенной для вычисления значения логической функции:

В теле определения процедуры

```
: PrintT3LogFunc {StrName,Len}
    .xy {x,y} @R C 3+ !R @W {GTP}
  {x,y,'TFunc} EXEC {TFunc(x,y)}
  ." =" .T9 ." = "
  {StrName,Len} TOS ."(x,y)" CR { };
: .T9 {V} 9 TON3 {V} ;
: .xy .---000+++. ." x=" C .T9 CR
 .-0+-0+-0+. ." y=" C .T9 CR {x,y};
```

с помощью операций **@R C 3+ !R @W** (или **GTP**) выбирается (в стек данных) первый параметр из тела её вызова. По взятому из тела параметру-адресу далее командой **EXEC** вызывается процедура (TFunc), назначенная для вычисления значения троичной логической функции от двух параметров (x,y), заданных в стеке.

После определения такой универсальной процедуры **PrintT3LogFunc** можно воспользоваться ей многократно, чтобы напечатать таблицы троичных операций, исполняемых командами ТВМ:

```
 "TADD"  PrintT3LogFunc TADD
 "TMUL"  PrintT3LogFunc TMUL
 "TMIN"  PrintT3LogFunc TMIN
 "TMAX"  PrintT3LogFunc TMAX
```

А также некоторых троичных логических функций, определённых на их основе:

```
{{ TSUB(x,y) = (x-y) mod 3 = TADD(x,NEG(y))
 : TSUB {x,y} NEG TADD {x-y} ;
{{ Функция следования Брусенцова:
{{ tSLBr(x,y)= x*y( x*y-x+y )
: tSLBr {x,y} C2 C2 TMUL C E4
  {x*y,y,x*y,x} TSUB TADD TMUL
  { x*y(y+x*y-x) } ;
```

```
: Use_PrintT3LogFunc
." Таблицы функций троичной логики:" CR
 CR "TSUB"  PrintT3LogFunc TSUB
 CR "tSLBr" PrintT3LogFunc tSLBr ;
```

В рассмотренном примере процедура (TFunc), взятая из тела в качестве параметра, вызывалась лишь один раз при исполнении процедуры **PrintT3LogFunc**. Приведём теперь пример, где процедуре сортировки одномерного массива (вектора) троичных слов в качестве параметра из тела вызова передаётся процедура (или команда) сравнения двух чисел, которая вызывается многократно, т.е. всякий раз для сравнения двух элементов вектора. С помощью такой процедуры сравнения можно задать порядок, в каком надлежит отсортировать числа массива (по возрастанию, по убыванию или ещё по какому-то критерию). Место её вызова выделено здесь подчёркиванием:

```
{{ {AdrV,Len} SortVctr CmpFunc { }
: SortVctr {AdrV,Len} GTP E3 E2
  {'CmpFunc,A,L} DO- SortVI DD { };
: SortVI {'F,A,i} MaxVI {i=L-1,..,0}
  {'F,A,i,k} SwapVIk D {'F,A,i};
: MaxVI {'F,A,i}
 {опр.индекс макс.элемента среди A[0]..A[i] }
  C {i,k} C2 {i} DO- MaxVj {i,k};
: MaxVj{'F,A,i,k,j} C4 C C4 SHL + @W
  {..,A,A[k]}E2 C3 SHL + @W {..,A[k],A[j]}
  {'F,A,i,k,j,A[k],A[j]} 7 CT EXEC {!!}
  IF+ {if A[k]<A[j]} MaxVk=j {'F,A,i,k,j} ;
: MaxVk=j {..,k,j} E2 D C {..,k=j,j} ;
: SwapVIk {переставляет эл-ты A[k]<=>A[i] }
  {..,A,i,k} C3 C C3 SHL + {'A[k]}
  E2 C4 SHL + {..,A,i,k,'A[k],'A[i]}
```

```
  {--,'A[k],'A[i]} C2 @W C2 @W
  {--,'A[k],'A[i],A[k],A[i]} E3 !W {A[i]<=A[k]}
  {--,'A[k],A[i]} E2 !W{A[k]<=A[i]} {..,A,i,k};
```

Используя процедуру такой универсальной сортировки, можно выстроить вектор чисел в нужном порядке такими простыми вызовами:

```
    9 VCTR Z   : 'Z 0 ' Z ;
: <mod10{x,y} E2 mod10 E2 mod10
{xmod10,ymod10}<{(xmod10)<(ymod10)?};
: mod10{z}10 /DivMod E2D {zmod10};
```

```
'Z 10 SortVctr < {по возрастанию}
'Z 10 SortVctr > {по убыванию}
{и в порядке возрастания младшей цифры:}
'Z 10 SortVctr <mod10
```

2.3. Разработка итераторов

Следующим рассмотрим пример передачи процедуры в качестве параметра другой процедуре для того, чтобы вызывать её многократно для выполнения обработки каждого элемента перебираемой структуры данных. Такие универсальные процедуры, предназначенные для обхода всех элементов имеющейся структуры данных с выполнением над каждым из них процедуры (или функции), заданной ей в качестве параметра, с некоторых пор стали называть **итераторами** (после их применения в широко известной библиотеке Turbo Vision [9]).

Сначала в качестве простейшего примера создания итератора рассмотрим универсальную процедуру обработки элементов произвольно заданного вектора двухтрайтных значений в порядке их обхода от (n-1)-го элемента до 0-го:

```
{..,A,n}ForEachTTVctr Operation{} ...
{{   {..,A,i,Xi} Operation {..,A,i,newXi}
: ForEachTTVctr{..,A,n}
   DO- DoEachTTVItem {..,A} D{..}
   @R 3+ !R{RA:=RA+3} ;
: DoEachTTVItem {..,A,i} C2 C2 C + +
   {..,A,i,'Ai} @TT {..,A,i,Xi}
   @R @R C !R E2 !R @W
   {'Operation} EXEC {..,A,i,newXi}
   {..,A,i,newXi} C3 C3 C + +
   {..,A,i,newXi,'Ai} !TT {..,A,i} ;
```

С помощью этой универсальной процедуры можно умножить каждый i-ый элемент вектора на 10, увеличить на i, распечатать через запятую как десятичное или девятеричное число:

```
 9 DTRYTE VCTR Z   : 'Z 0 ' Z ;
: *10 {X} 10 * {newX=X*10} ;
: +i {i,X} C2 + {i,newX=X+i} ;
: .[i], {i,X} . ',' TOB {i,X} ;
: .9[i], {i,X} C 0 TON9 ',' TOB {i,X};
```

```
'Z 10 ForEachTTVctr *10
'Z 10 ForEachTTVctr +i
'Z 10 ForEachTTVctr .[i], CR
'Z 10 ForEachTTVctr .9[i], CR
```

В качестве более сложного примера реализации итератора рассмотрим универсальную процедуру обхода всех элементов (узлов) линейного списка, на основе которого построен словарь ДССП:

```
: ForEachWord GTP {адрес ProcNode}
SLOVAR NextNode? DW HndlNode DD{};
: NextNode? @W C ;
: HndlNode {ProcNode,NodeAdr}
   C2 EXEC {ProcNode,NodeAdr} ;
```

{ProcNode – процедура из тела вызова, назначаемая для исполнения над каждым узлом словаря}

Пример применения итератора для распечатки и подсчёта количества тех слов словаря, в имени которых содержится заданная строка:

```
: .WORDS? {pWord,Len}
 0 {Cnt=0} ForEachWord .=Word? CR
 ." count of words= " . DDD { } ;
: .=Word? {pWord,Len,Cnt,PN,NodeAdr}
 5 CT 5 CT C3 _WAdr C4 _WLen
 str_pos? 0 >= IF+ .+Word ;
: .+Word {Cnt,PN,NA} E3 1+ E3
 {Cnt+1,PN,NA} .Word {Cnt+1,PN,NA} ;
: .Word {печать имени слова с пробелом}
{Adr} C _WAdr C2 _WLen TOS SP{Adr};
```

```
{pStr1,L1,pStr2,L2} str_pos? {pos}
определяет позицию pos, где содержится строка
(pStr1,L1) в строке (pStr2,L2)
```

Другой вид итераторов **FirstWordThat**, предусмотренный в ДССП, позволяет вести поиск в словаре требуемого слова и в качестве результата получить указатель на найденный в нём узел этого слова.

2.4. Команда вызова процедуры из заданной в теле таблицы

Список параметров P1,..,P**n**, задаваемый в теле вслед за вызовом команды **call_CMND** (см. Рис. 5), удобен для представления в этом месте таблицы процедур (или команд), из которых предстоит выбирать на исполнение одну по заданному в вершине стека номеру (k). Продемонстрируем, как можно реализовать команду такого многовариантного выбора для случая, когда длина такой таблицы процедур уже известна (9) заранее:

```
{{... {k} ExecPTable9 P1 P2 ... P9 ...
: ExecPTable9 {k} 5 - #4 TMUL 4+
 {скорректируем k=(k-5)&(.++.)+4 }
  SHL  @R C 27 + !R {RA=RA+9*3}
  {3*k',RA} + @W {Pk} EXEC ;
```

или задаётся (**n**) перед началом таблицы:

```
{...{k} ExecPTable n P1 P2 ... Pn ...}
{{   if k<1 then call_P1 else
{{   if k>n then call_Pn else
{{ { if k in [1,n] then } call_Pk
: ExecPTable {k} @R C @TT {k,RA,n}
   C 1+ SHL C3 + !R {RA=RA+(n+1)*3}
 {k,RA,n} E2 E3 {RA,n,k} crctK
  E2D SHL + @W {Pk} EXEC { };
: crctK {if k>n then k:=n; if k<1 then k:=1;}
  {n,k} C C3 > IF+ k:=n {n,k}
        C 1 < IF+ T1 {n,k'} ;
 : k:=n {n,k} D C {n,k'=n} ;
```

3. Разработка новых команд для организации ветвлений

Для реализации процедур, предназначенных для исполнения команд ветвлений, необходимо извлекать из вызываемого тела адрес процедуры (или команду), выбранной(-ую) на исполнение, а также передвигать адрес возврата по телу сшитого кода, чтобы он стал указывать на позицию, расположенную следом после всей управляющей конструкции, подлежащей исполнению.

Сначала рассмотрим простейшие приёмы реализации команд для организации одиночного, двоичного и троичного ветвлений, а затем - команды, обеспечивающие сложные конструкции многовариантного выбора (ветвления).

3.1. Реализация команд пропуска

Команды условного вызова **IF- IF0 IF+**, используемые для одиночного ветвления, можно трактовать как команды пропуска последующего слова при соблюдении обратного условия. Продемонстрируем приёмы разработки подобных команд на примерах определения процедур для пропуска по заданному условию одной, трёх или серии из **n** команд, схемы которых представлены на Рис. 6, 7, 8.

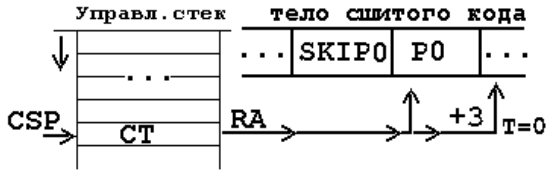


Рис. 6. Схема команды пропуска **SKIP0**

```
...{T} SKIP3- P1 P2 P3 ...
```

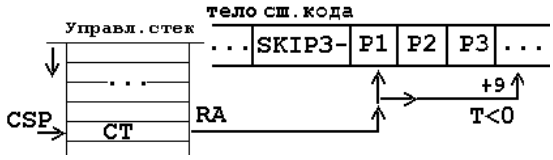


Рис. 7. Схема команды пропуска **SKIP3-**

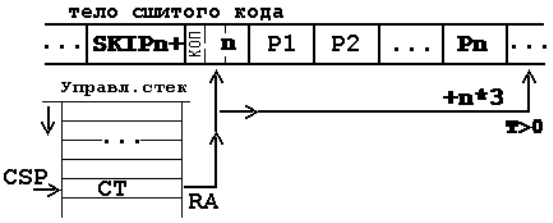


Рис. 8. Схема команды пропуска **SKIPn+**

```
{{проскок следующей команды (P0),если
{{вершина стека нулевая: ... SKIP0 P0 ...
: SKIP0 {x} @R E2 {RA,x} IF0 3+ !R
{ if x=0 then RA:=RA+3 } { } ;
```

```
{{проскок следующих 3-х команд (P1,P2,P3),
{{если вершина <0 ... SKIP3- P1 P2 P3 ...
: SKIP3- {x} @R E2 {RA,x} IF- 9+
  !R{if x<0 then RA:=RA+9 }; : 9+ 9 + ;
```

```
{{проскок следующих n команд P1,P2,P3 .. Pn,
{{если вершина стека положительна:
{{ ... SKIPn+ n P1 P2 P3 ... Pn ...
: SKIPn+ {x} @R C @TT
  {@W #444444444 TMUL }
  {x,RA,n} SHL E2 E3 {RA,3*n,x}
  {RA,3*n,x} BR+ + D 3+ !R { } ;
{if x>0 then RA:=RA+(n+1)*3 else RA:=RA+3;}
```

Заметим, что для извлечения из тела числа **n** в процедуре **SKIPn+** применялась операция **@TT**, чтобы сразу взять из памяти двутрайтное значение (хотя в комментарии далее показан и другой возможный вариант выделения числа **n** из 27-тритного кода команды DPushi_**n**, сформированного компилятором для слова **n**).

3.2. Реализация команд двоичного выбора (ветвления)

При реализации команд двоичного ветвления требуется выбрать на исполнение одно из двух слов, расположенных следом в теле:

```
{{Команда 2-го ветвления ... BR0N P1 P2 ...
{{ if T<>0 Then P1 else P2
: BR0N {T} @R C E3 {RA,RA=^P1,T}
  IF0 3+{RA,^Pi}E2 6 + !R{RA:=RA+6}
  {^Pi} @W EXEC { } ;
```

```
{{ Команда 2-го ветвления ... BRLE P1 P2 ...
{{ if T<=0 Then P1 else P2
: BRLE {T} @R C E3 {RA,RA=^P1,T}
  IF+ 3+{RA,^Pi}E2 6 + !R{RA:=RA+6}
   {^Pi} @W EXEC { } ;
```

3.3. Реализация команд троичного выбора (ветвления)

При реализации команд троичного ветвления предстоит в зависимости от условия выбрать на исполнение одно слово из расположенной следом по телу тройки слов и

обеспечить дальнейшее продолжение программы с позиции, стоящей за этой тройкой.

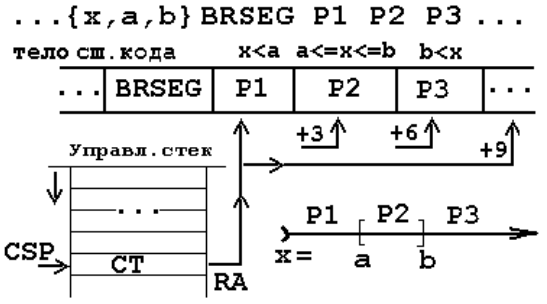


Рис. 9. Схема команды ветвления **BRSEG**

Приёмы реализации команд троичного ветвления рассмотрим на примере команды **BRSEG** (см. Рис. 9), которая вызывает одну из трёх предусмотренных процедур (или слов-команд) в зависимости от расположения оцениваемого значения **x** на числовой оси слева, внутри или справа от заданного сегмента [**a,b**]:

```
{{ Команда троичного ветвления BRSEG:
{{ ... {x,a,b} BRSEG P1 P2 P3 ...
{call P1,if x<a ; call P2,if x in[a,b]; call P3,if b<x}
: BRSEG {x,a,b} inSeg? {T=-1/0/+1}
   @R C E3 {RA,RA=^P1,T}
   1+ SHL + {RA,^Pi=^P1+3*(T+1)}
   E2 {^Pi,RA} 9 + !R {RA:=RA+9}
   {^Pi} @W EXEC { } ;
: inSeg? {x,a,b} C3 E2 CMP E3
   {x cmp b, a, x} E2 CMP
   {x cmp b, x cmp a } tCarry {-1/0/+1}
   {= -1,if x<a ; 0,if x in[a,b]; +1,if b<x } ;
: tCarry{x,y} C2 C2 TADD E3
   {x+y,y,x}TMUL TMUL NEG {-x*y*(x+y)};
```

3.4. Реализация команд многовариантного выбора (ветвления)

В многовариантных управляющих конструкциях ветвления предлагается выбрать действие из предусмотренного заранее списка процедур (или команд). В ДССП подобный список располагается следом за командой **BR** (см. Рис. 1), где перед каждой процедурой (Pi) помещается ещё и слово, выдающее значение (Vi), при совпадении с которым оцениваемого значения (V) и должна выбираться данная ветка продолжения программы.

Раннее в одной из версий ДССП предусматривалась ещё одна команда многовариантного выбора **BRINT**, которая позволяла выбрать одну из процедур в зависимости от того, в какой из нескольких полуинтервалов, заданных числами V1,..,Vk, попадает оцениваемое значение V. Продемонстрируем, как можно реализовать в ДССП-ТВМ такую новую команду для многовариантного ветвления.

Если заменить в блок-схеме исполнения команды **BR** на Рис. 1 все условия вида **T=V?** на **T>=V?**, то получившаяся блок-схема станет задавать порядок исполнения команды **BRINT**:

```
{x} BRINT V1 P1 V2 P2 ... Vk Pk _ELSE P0
  if x<=V1 then P1 else
  if x<=V2 then P2 else ...
  if x<=Vk then Pk else P0
```

Фрагмент тела сшитого кода, который сформирует компилятор для управляющей конструкции **BRINT ... _ELSE**, схематично представлен на Рис. 10. На нём стрелками изображено, как пройти по этому телу и найти правильный вариант процедуры для выбора, а также найти позицию продолжения программы после её исполнения.

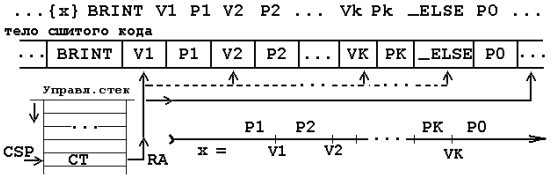


Рис. 10. Схема команды выбора **BRINT .. _ELSE**

Руководствуясь этой пояснительной схемой, можно предложить такую реализацию процедур исполнения команд **BRINT** и **_ELSE**:

```
: BRINT {поиск такого Pi, что Vi>=V }
  {V} BRI_Vi? DW BRI_nextViPi
  {V} D @R C !R {RA} 3+ @W {Pi}
   {поиск конца _ELSE для BRINT}
   BRI_ELSE?_++ DW NOP
  @R 6 + !R {RA=>на поз.после _ELSE P0}
  {Pi} EXEC {исполняем Pi} ;
: BRI_Vi?{V} @R @R C !R E2 !R @W
  {V,'Vi}EXEC{V,Vi} C2 < {V,Vi<V?};
: BRI_nextViPi {V} {RA':=RA'+6}
   @R @R 6 + !R !R {V} ;
: BRI_ELSE?_++ {[RA,RA!DW]} @R @R
 {[]RA!DW,RA}6 + C !R{RA:=RA+6}
 {[RA]RA!DW,RA}E2 !R{[RA,RA!DW]RA}
 {[RA,RA!DW]RA} @W '' _ELSE <>
 {[RA,RA!DW] Mem[RA]<>'_ELSE?} ;
```

```
: _ELSE{V}{вызвали вместо очередного Vi}
  @R D{удалим адрвозврата из вызова EXEC}
  @R D{удалим адрвозврата из BRI_Vi?}
  @R {адрес возврата из BRINT --> _ELSE}
  3+ !R {передвинем его дальше по телу}
 {V}D {};{и сразу же вернёмся по нему на
 продолжение исполнения тела после _ELSE}
```

Заметим, что такую управляющую конструкцию **BRINT .. _ELSE** можно считать расширенным вариантом команды **BRSEG**, так как

```
X a b BRSEG P1 P2 P3
```

почти что эквивалентно

```
x BRINT a P1 b P2 ELSE P3
```

4. Создание новых команд для организации циклов

Команды повторения **LOOP DO- DW**, предусмотренные в ДССП-ТВМ для организации циклов, в настоящее время исполняются напрямую троичной [виртуальной] машиной. Но первоначально они были реализованы в ДССП-ТВМ сначала на языке самой системы, а затем (для эффективности) переписаны на языке ассемблера ТВМ и помещены в ассемблерное ядро, которое подключается при компоновке ДССП-программы.

Продемонстрируем здесь их возможные реализации в виде процедур на языке ДССП.

4.1. Команда бесконечного цикла и команда выхода из него

Команду **LOOP** реализовать проще всего: надо взять из тела вызова процедуру **PBody** (или команду), представляющую тело цикла, и перед её вызовом скорректировать адрес возврата (RA) так, чтобы после исполнения тела цикла осуществить возврат на позицию, откуда будет снова взята на исполнение эта же команда **LOOP** (см. Рис. 11):

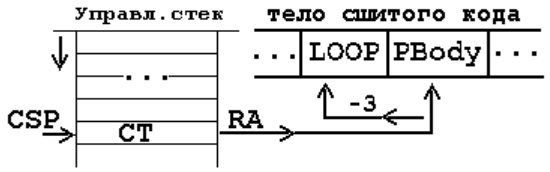


Рис. 11. Схема команды цикла **LOOP**

```
{{ команда организации бесконечного цикла
{{ ... LOOP_ PBody ...
: LOOP_ @R C 3- !R {RA:=RA-3}
   {^PBody} @W {PBody} EXEC { } ;
```

При применении команды **LOOP** обычно предполагается, что предусматриваются также и некоторые команды, с помощью которых можно будет осуществить экстренный выход из организуемого ею бесконечного цикла. Рассмотрим пример реализации одной из них:

```
{{ Команда EXIT_LOOP, чтобы прекратить
{{ цикл, исполняемый командой LOOP_
: EXIT_LOOP @R { адрес возврата (AB) }
   D {удалили AB из процедуры EXIT_LOOP}
@R D{удалим AB из EXEC в теле проц. LOOP}
  @R {AB из процедуры, реализующей LOOP }
6 + {изменим, чтобы продолжить исполнение}
  !R {программы после команды LOOP} { } ;
```

Заметим, что такой командой можно осуществить выход из цикла **LOOP**, только если употребить её в теле самой процедуры цикла (**PBody**). А в общем случае для прекращения нескольких вложенных (друг в друга) процедур или циклов следует применять предусмотренный в ДССП-ТВМ механизм обработки исключительных ситуаций [14, п. 4.1].

4.2. Команда цикла со счётчиком

При исполнении команды цикла **DO-** (см. Рис. 2) всякий раз перед вызовом процедуры (команды) тела цикла необходимо уменьшать на 1 значение счётчика–параметра цикла, которое сохраняется в вершине стека, и проверять, что оно остаётся неотрицательным. При продолжении цикла адрес возврата требуется вернуть в позицию самой команды цикла **DO-**, а для завершения цикла передвинуть его вперёд на позицию слова, стоящего за всей конструкцией этого цикла (см. Рис. 12):

```
{{команда организации цикла со счётчиком
{{ ... {n} DO- PBody ...   {i} PBody {i}
: DO- {n} 1- @R C 3- !R {RA:=RA-3 }
  {n-1,RA=^PBody} @W C2 {n-1,'PBody,n-1}
     BR- DOmEnd EXEC {n-1} ;
  : DOmEnd {n-1,'PBody} @R DDD { }
    @R 6 + !R { RA:=RA+6 } ;
```

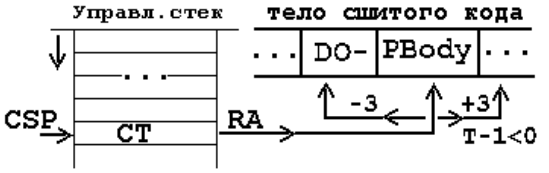


Рис. 12. Схема команды **DO-** цикла со счётчиком

4.3. Команда цикла с условием

Для реализации команды цикла с условием (см. Рис. 3) необходимо передвигать адрес возврата не на одну, а уже на две позиции назад (см. Рис. 13), чтобы после исполнения тела цикла снова вызывать процедуру, вычисляющую условие цикла:

```
{{ Цикл с условием: ... Cond DW Body ...
{{ Cond вычисляет условие: Да (<>0) | Нет (0)
: DW{U=0/1} @R E2 {RA,U} IF0 DWEnd
  {RA} C @W {'Body} E2 6 -
  {'Body,RA-6} !R {RA:=RA-6}
  {'Body} EXEC { } ;
: DWEnd{RA} @R D 3 + !R {RA:=RA+3};
{ сразу с выходом из команды DW !! }
```

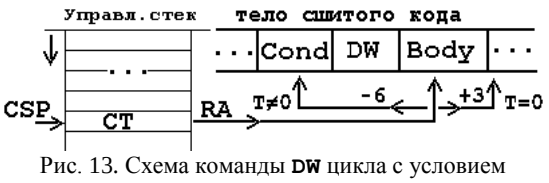


Рис. 13. Схема команды **DW** цикла с условием

4.4. Цикл с троичным условием

В ДССП для троичной машины появилась новая команда цикла с троичным условием. Порядок исполнения такой команды **DW-+** изображён в виде блок-схемы на Рис. 14.

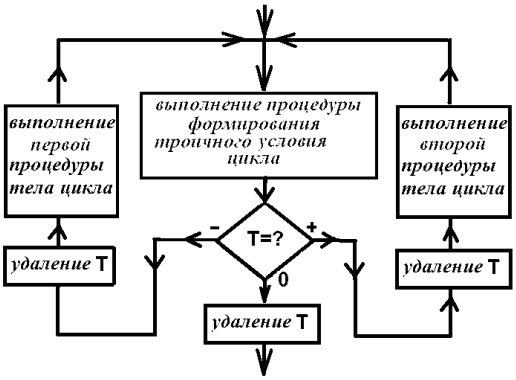


Рис. 14. Блок-схема исполнения команды цикла с троичным условием

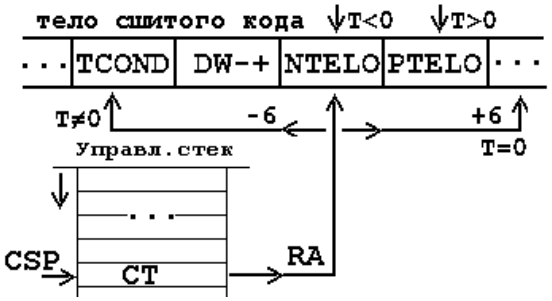


Рис. 15. Схема команды цикла с троичным условием

А схема корректировки адреса возврата в процедуре реализации этой команды — на Рис. 15:

```
{{ Команда цикла с троичным условием:
{{ ... TCOND DW-+ NTELO PTELO ...
{{TCOND вычисляет троичное условие: -1,0,+1
{{если вершина<0,то выполняется тело NTELO
{{если вершина>0,то выполняется тело PTELO
{{если вершина =0, то цикл прекращается
: DW-+ {U} @R E2 {RA,U} SGN
  {-|0|+} BRS DWTN DWTEnd DWTP
  {NTELO|PTELO,RA-6} !R {RA:=RA-6}
  {NTELO|PTELO} EXEC { } ;
: DWTEnd{RA} @R D 6 + !R{RA:=RA+6};
: DWTN{RA} C @W E2 6 -{NTELO,RA-6};
: DWTP{RA}C 3+ @W E2 6 -{PTELO,RA-6};
```

Такая команда цикла с троичным условием **DW-+** была реализована сначала на языке ДССП, затем для повышения эффективности — на ассемблере ТВМ. А после того, как стала очевидной полезность её применения (см. далее п. 5.1 и [16]) на троичной машине, аналогичная команда цикла (с троичным условием) была добавлена уже в систему команд самой троичной машины, а также была реализована в её имитаторе (ТВМ).

4.5. Реализация конструкции цикла с многими условиями и телами

И в качестве последнего примера реализации собственной довольно сложной управляющей конструкции покажем, как в ДССП-ТВМ можно создать аналог конструкции цикла **do**…**od**, предложенной Э. Дейкстрой в [8, с.59-60] и упомянутой нами во введении.

Для обеспечения функционирования такого цикла с множественными условиями и телами предложим добавить в ДССП-словарь две команды: **WHILE** и **ENDW.** Порядок исполнения цикла, образуемого этими двумя командами, изображён на Рис. 16.

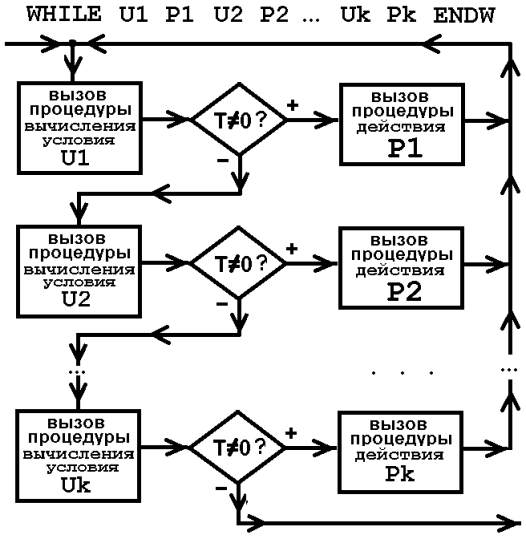


Рис. 16. Блок-схема исполнения конструкции цикла с множественными условиями и телами

Для реализации этих двух команд можно определить такие процедуры на языке ДССП:

```
{{цикл с множественными условиями и телами
{{ WHILE U1 P1 U2 P2 ... Uk Pk ENDW Px
: WHILE {[RA=^U1]} @R C 3- !R !R
  {[RAW=^While,RAi=^U1]}
  {[RAW,RAi=^Ui]} ~Ui? DW RAi+3
  {[RAW,RAi=^Pi]} @R @W
  {[RAW] Pi} EXEC {[RAW] } ;
: ~Ui?{[RAW,RAi=^Ui,RA]}
    @R @R C @W E2 3+ !R E2 !R
    {[RAW,RAi=^Pi,RA] cmnd_Ui }
   { Ui } EXEC {U} NOT {~U} ;
: RAi+3{[RAW,RAi=^Pi],RA} @R @R 3+
     !R !R {[RAW,RAi=^Ui,RA]} ;
: ENDW {[RAW,RAi=^Px,RA,RA']}
  @R D {{выбросили АВ вызова call_ENDW
  @R D {{выбросили АВ в команду DW
  @R @R D !R {[RAi=^Px]} ;
  {{теперь АдресВозврата ==> ^Px
```

Действия с адресами возврата, указывающими на отдельные позиции этой конструкции цикла, отображены схематично на Рис. 17. Выполняя последовательно процедуры Ui, вычисляющие условия, надо определить, какая первая из них даёт в качестве ответа значение истины (T≠0), и вызвать на исполнение в качестве тела цикла процедуру (Pi), соответствующую этому условию. Перед вызовом этой процедуры надлежит установить адрес возврата на позицию, где расположена команда **WHILE** этого цикла, чтобы после вызова процедуры тела цикла Pi продолжить исполнение цикла. Если же на очередном шаге цикла ни одна из процедур вычисления условий Ui не выдаст значения истины, то последует вызов команды **ENDW**, которая должна будет прекратить исполнение цикла и обеспечить продолжение программы с позиции, стоящей после команды **ENDW**.

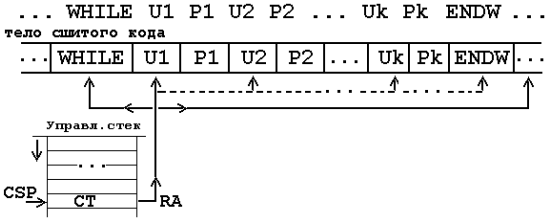


Рис. 17. Схема конструкции цикла **WHILE** … **ENDW**

5. Примеры применения новых управляющих конструкций

Приведём примеры применения некоторых новых управляющих конструкций, объяснению приёмов реализации которых (на языке ДССП) была посвящена основная часть изложенного ранее материала.

5.1. Применение команды DW-+

Сначала сравним два варианта ДССП-процедур вычисления наибольшего общего делителя двух натуральных чисел НОД(X,Y) алгоритмом Евклида: 1) с применением команды цикла **DW** с двоичным условием и команды BR- условного выбора; 2) с применением команды цикла **DW-+** с троичным условием:

```
: NOD1 {x,y} x-y? DW y-x|x-y D ;
: y-x|x-y{x,y} x-y? BR- y-x x-y ;
 : x-y?{x,y} C2 C2 - ;
 : y-x{x,y} C2 - {x,y-x};
 : x-y{x,y} E2 C2 - E2 {x-y,y};
```

```
: NOD2 x?y DW-+ y-x x-y  D ;
 : x?y{x,y} C2 C2 CMP ;
 : y-x{x,y} C2 - {x,y-x} ;
 : x-y{x,y} E2 C2 - E2 {x-y,y};
```

Обе этих процедуры вычисления НОД запускались на программном имитаторе троичной машины (ТВМ) с замером количества исполненных команд. В результате проведённых испытаний было подтверждено, что алгоритм процедуры **NOD2**, применяющий цикл с троичным условием, не только выглядит проще, но ещё и работает быстрее (более, чем в полтора раза), так как исполняется за меньшее число команд ТВМ (см. таблицу 1).

Таблица 1. Сравнение ДССП-процедур NOD1 и NOD2 по числу исполненных команд

аргументы		результат	кол-во исп.команд		
X	**Y**	**Z=NOD(X,Y)**	**NOD1**	**NOD2**	**КВ**
1010	15100	10	958	542	1.77
19683	45	9	8814	5294	1.66
1001	29	1	970	578	1.68
2000	155	5	492	300	1.64
9999	199	1	1446	870	1.66
6000	2250	750	96	64	1.48
КВ – коэффициент выигрыша, который процедура NOD2 обеспечивает по отношению к NOD1					

5.2. Применение команды BRINT

Допустим, требуется решить такую задачу. Дана последовательность чисел, каждое из которых (в диапазоне от 0 до 100) задаёт число баллов, полученных студентами группы на экзамене. Пусть признаком конца последовательности будет число -1. Требуется составить программу, которая должна подвести итог экзамену и подсчитать, сколько получено студентами группы оценок «отлично», «хорошо», «удовлетворительно» и «неудовлетворительно», полагая, что критерии выставления оценок таковы: 0-59 баллов — 2 (неуд), 60-74 — 3 (удовл), 75-89 — 4 (хорошо) и 90-100 — 5 (отлично).

Сначала приведём вариант решения этой задачи с применением конструкции множественного ветвления **BRINT .. _ELSE** :

```
  VAR Neud { кол-во неудов }
  VAR Udov { кол-во удовл. }
  VAR Hor  { кол-во хорошо }
  VAR Otl  { кол-во отлично}
: ExamResults {..,x} 0 ! Otl
   0 ! Hor 0 ! Udov 0 ! Neud
   {..,x} >=0? DW Look D { }
  ." Otl="  Otl . ." Hor="  Hor .
 ." Udov=" Udov . ." Neud=" Neud .
   DDDD { } ;
: >=0?{x} C 0 >= {x,x>=0?};
: Look {x} BRINT
        59 Neud++ 74 Udov++
        89 Hor++ _ELSE Otl++ {} ;
: Otl++ Otl 1+ ! Otl {};
: Hor++ Hor 1+ ! Hor {};
: Udov++ Udov 1+ ! Udov {};
: Neud++ Neud 1+ ! Neud {};
```

```
: Task CR
```

```
 ."Задача распределения оценок экзамена:"
  -1  60  75  90 100
  45  61  67  74  75  79  80  85  36  89  90
   0  55  70  87  77  99  68  93  88
   ExamResults  ;
```

5.3. Применение цикла WHILE...ENDW

А теперь для решения той же задачи приведём пример программы, в которой применяется цикл **WHILE** … **ENDW**:

```
: ExamResults {..,x} 0 ! Otl
   0 ! Hor 0 ! Udov 0 ! Neud
   WHILE >=90? `Otl+  >=75? `Hor+
         >=60? `Udov+ >=0?  `Neud+
   ENDW D
  ." Otl="  Otl . ." Hor="  Hor .
 ." Udov=" Udov . ." Neud=" Neud .
   DDDD { } ;
: >=90?{x} C 90 >= ;
: `Otl+{x} D Otl 1+ ! Otl {};
: >=75?{x} C 75 >= ;
: `Hor+{x} D Hor 1+ ! Hor {};
: >=60?{x} C 60 >= ;
: `Udov+{x} D Udov 1+ ! Udov {};
: >=0?{x} C 0 >= ;
: `Neud+{x} D Neud 1+ ! Neud {};
```

Заключение

В статье рассмотрены основные приёмы разработки в ДССП-ТВМ собственных управляющих команд, который каждый пользователь-программист может создать сам, чтобы добавить в словарь ДССП.

Применение этих новых управляющих команд позволяет не только выразить в сокращённом виде исходный текст созидаемой ДССП-программы, но также и уменьшить в ряде случаев количество исполняемых ею команд, что позволяет в итоге повысить её производительность.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «Исследование и разработка моделей микропроцессоров, ориентированных на задачу горения для создания отечественной супер-ЭВМ» (0065-2014-0060).

.

Development of own control constructions in DSSP for the ternary computer

A.A.Burtsev

Abstract. Programming systems with the dictionary organization (FORTH and DSSP) give to the user programmer a unique opportunity by building of the dictionary to create in fact an own programming language. As new words it is possible to add to the dictionary also such words which are intended to control the commands flow of program execution. In the article methods of development in the DSSP-TVM (the version of DSSP for the ternary computer) special procedures for such words (commands) which provide functioning of the own control constructions to be added to the DSSP language are considered in detail.

Keywords: structured programming, dictionary organization, FORTH, DSSP, ternary computer, threaded code, control constructions.

Литература

1. С.Н.Баранов, Н.Р.Ноздрунов. Язык ФОРТ и его реализации. - Л.: Машиностроение, 1988.- 157 с.

2. А.А.Бурцев, С.А.Сидоров. История создания и развития ДССП: от «Сетуни-70» до троичной виртуальной машины // Труды 2-ой международной конференции «Развитие вычислительной техники и её программного обеспечения в России и странах бывшего СССР» SORUCOM-2011, г. Великий Новгород, Россия, 12–16 сентября 2011г.— В.Новгород: Изд-во НовГУ, 2011.— С. 83–88.

3. А.А.Бурцев, М.А.Бурцев. ДССП для троичной виртуальной машины // Труды НИИСИ РАН, т. 2, № 1.— М.: Изд-во НИИСИ РАН, 2012.— С. 73–82.

4. У.Дал, Э. Дейкстра, К.Хоор. Структурное программирование. — М.: Мир, 1975. –247 с.

5. С.А.Дапкунас, К.Н.Раннев. Простые средства обеспечения работы с подпрограммами и структурированного программирования на языке Бейсик. // Анализ и синтез дискретных устройств (межвуз. сб. статей под ред.М.К. Чиркова, С.П. Маслова). – Л.: Изд-во ЛГУ, 1984. — с. 45-49.

6. А.А.Бурцев. Об одной возможности расширения языка программирования средствами обработки исключительных ситуаций. // Дискретные модели. Анализ, синтез, и оптимизация (сб. статей под ред.М.К. Чиркова, С.П. Маслова). – Спб.: СпбГУ, 1998. — с. 171-181.

7. Г.В.Златкус. Ассемблер малой ЦМ «Сетунь-70» // Вычислительная техника и вопросы кибернетики. Вып №17. (сб. статей под ред. Брусенцова Н.П., Шаумана А.М.). М.: Изд-во МГУ, 1981, с. 26-33.

8. Э.Дейкстра. Дисциплина программирования. — М.: Мир, 1978. — 275 с.

9. В.В.Фаронов. Турбо Паскаль 7.0. Начальный курс. Учебное пособие. — М.: Нолидж, 1997.

10. Н.Вирт. Программирование на языке Модула-2. — М.: Мир, 1987. – с. 179-182.

11. С.Янг. Алгоритмические языки реального времени: конструирование и разработка. — М.: Мир, 1985. – с. 146-177.

12. Н.Джехани. Язык Ада. — М.: Мир, 1988. – с. 125-180.

13. А.В.Борисов. Схемы взаимодействия параллельных процессов. // Программное оснащение микрокомпьютеров (сб. статей под ред. Брусенцова Н.П. и Маслова С.П.). - М.: Изд-во МГУ, 1982. - с. 88-99.

14. А.А.Бурцев, С.А.Сидоров. Троичная виртуальная машина и троичная ДССП. // Программные системы: теория и приложения. — 2015. — Т. 6, №:4 (27). — с. 29–97. [электронный ресурс] URL: http://psta.psiras.ru/read/psta2015_4_29-97.pdf.

15. А.А.Бурцев, С.А.Сидоров. Программный комплекс ДССП-ТВМ для структурированного программирования троичной [виртуальной] машины. // Современные информационные технологии и ИТ-образование. 2015. — Т. 11, № 2. — с. 371–379.

16. А.А.Бурцев. Особенности программирования в ДССП для троичной машины. // Современные информационные технологии и ИТ-образование, 2017. —Т.13, № 4, 2017. с. 188-196.

17. А.А.Бурцев, Д.В. Франтов, М.Н.Шумаков. Разработка интерпретатора сшитого кода на языке Си. // Вопросы кибернетики (сб. статей под ред. В.Б.Бетелина) — М.: Изд-во НИИСИ, 1999, с. 64-77.

18. А.А.Бурцев. Процедурный сшитый код для обеспечения нисходящей разработки структурированных программ в ДССП для троичной машины. // Труды НИИСИ РАН, т. 5, № 1.— М.: Изд-во НИИСИ РАН, 2015.— с. 122–129.

Некоторые вопросы эффективности детерминированных алгоритмов распознавания образов с помощью библиотеки OpenCV

Н.О.Бесшапошников[1], М.А.Кузьменко[2], А.Г.Леонов[3], М.А.Матюшин[4]

[1,2,3,4] ФГУ ФНЦ НИИСИ РАН, [2,3,4] ФГБОУ ВО «Московский государственный университет имени М.В.Ломоносова», Механико-математический факультет и [3] ФГБОУ ВО «Московский педагогический государственный университет» (МПГУ), Москва, Россия,

E-mail's: [1]nikita.beshaposhnikov@gmail.com ,[2] gmk@infomir.ru , [3]dr.l@math.msu.ru , [4]itsaprrank@yandex.ru

Аннотация. В настоящее время технология распознавания образов с использованием нейронных сетей занимает существенную часть в программных системах, находящие применения в широких областях науки и техники. Однако, ограниченные компьютерные мощности не позволяют сделать эти решения массовыми. Для обеспечения эффективности алгоритмов распознавания и совмещения реальных и виртуальных объектов возможно использовать и детерминированные методы. В статье рассматривается исследование эффективности применения таких методов с использованием библиотеки OpenCV.

Ключевые слова: OpenCV, распознавание образов, нейронные сети.

В последние годы технологии по распознаванию образов с помощью нейронных сетей все чаще используются в различных областях науки и техники, начиная от робототехники и автомобилестроения и, заканчивая торговлей и образованием. Возросшая вычислительная мощь современных персональных компьютеров, смартфонов и планшетов подталкивает разработчиков программного обеспечения к созданию систем, объединяющих виртуальную и дополненную реальность для индустрии компьютерных игр, промышленности, учебных организаций и т.п. При этом, от мобильных устройств требуется не только возможность обрабатывать потоковое видео, но и обеспечить разумную временную реакцию на происходящие события, например, при движении объектов, конструируя элементы дополненной реальности, перенося настоящие объекты в виртуальный мир.

Для обеспечения эффективности алгоритмов распознавания и совмещения реальных и виртуальных объектов авторами предлагается объединить при использовании детерминированные (точные, основанные на аналитической геометрии) и эвристические методы.

Для исследовательских экспериментов была выбрана кроссплатформенная библиотека компьютерного зрения библиотека OpenCV, содержащая в себе множество оптимизированных алгоритмов, которые можно использовать для нахождения и распознавания различных объектов в изображениях или видео-потоках [1].

Библиотека OpenCV была создана корпорацией Intel (США) в 1999 году как часть другого большого проекта по стереодисплеям. С 2006 года по настоящее время разработано 3 поколения библиотеки. В современном варианте OpenCV включает в себя более 500 функций компьютерного зрения и обработки изображений, а также некоторые полезные алгоритмы общего назначения. Разработчики библиотеки утверждают, что созданный ими система обладает высокой мобильностью и эффективностью [2].

Одной из интересных проблем является задача распознавания автомобильных (российских) номеров с использованием мобильных устройств. Съемка производится камерой смартфона или планшета и распознавание необходимо производить в реальном времени.

Для решения поставленной задачи были применены методы библиотеки OpenCV. В процессе работы алгоритма распознавания использовался обученный каскадный детектор Хаара и библиотека TesseractOCR для распознавания букв и цифр на номере. Схема работы алгоритма выглядит так:

1. Производится поиск всех подходящих областей (в которых содержится номер) изображения с помощью каскада Хаара;

2. В каждой области с помощью детектора границ Канни находятся все контуры, отсекая кандидатов на шум (небольшие области, по сравнению с размером номера, которые детектируются как шум на изображении, чтобы отличить от области номера или буквы);

3. Контуры сортируются по размерам площади, и самый большой (или два из них, так как номер может распасться на две области из-за линии внутри него) определяется как граница самого номера;

4. Область номера преобразуется поворотом так, чтобы границы контура стали параллельны границам изображения;

5. Повторно применяется детектор границ Канни для нахождения областей, где детектируются цифры и буквы. Из них отбираются те, которые соответствуют примерным размерам прямоугольника номера (в основном занимают некоторую фиксированную площадь относительно площади номера);

6. К каждой найденной области используется алгоритм библиотеки TesseractOCR для конечного определения цифр и букв.

Сокращенная логика описанного алгоритма на языке C++ представлена ниже:

```
findNumbers()
{
  cv::UMat sourseImage;
  cv::UMat gray;
  cv::cvtColor(sourseImage, gray, CV_BGR2GRAY);
  equalizeHist(gray, gray);
  bilateralFilter(gray, gray, 3, 45, 45);

  std::vector<cv::Rect> plates;
 //Нахождение областей номеров с помощью каскада Хаара
  cascadePlate.detectMultiScale(gray, plates,1.1,10,CV_HAAR_DO_CANNY_PRUNING,cv::Size(50, 15), cv::Size(500, 150));

for(auto& p : plates)
  {
    //выделение части изображения с номером
     …
     findLetters(imagePart);
  }
}
findLetters(imagePart)
{
//применение фильтров для улучшения качества изображения
  cv::UMat cannyOutput, smothed, srcThreshold;
  auto clahe = cv::createCLAHE(2.0, cv::Size(8, 8));
  clahe->apply(imagePart, imagePart);
thresh-
old(imagePart,srcThreshold,0,255,CV_THRESH_BINARY,CV_THRESH_OTSU);
  medianBlur(srcThreshold, srcThreshold, 3);
  cv::blur(srcThreshold, srcThreshold, cv::Size(3,3));
  cv::Canny(srcThreshold, cannyOutput, 66, 133, 3);
   std::vector<std::vector<cv::Point>> rectContours;

  cv::findContours(cannyOutput, rectContours,CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);
//сортировка контуров по размеру
std::sort(rectContours.begin(), rectContours.end(), [](std::vector<cv::Point> &first, std::vector<cv::Point> &second)
{
    return cv::contourArea(first) > cv::contourArea(second);
 });
// вычисление угла angle на который повернута область номера (rectContours[0])
…
rotateImage(srcThreshold, angle);
cv::Canny(srcThreshold, cannyOutput, 66, 133, 3);
cv::findContours(cannyOutput, rectContours, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);
for(auto& c : contours)
{
//если размер области соответствует размеру номера, то применить распознавание символа в области с помощью TesseractOCR
recognizeChar(charPart);
}
}
```

Представленный алгоритм устойчив при различных уровнях освещения и частичном загрязнении номера.

К сожалению, в процессе исследовательского эксперимента не удалось побороть ошибки распознавания, связанные с неверным детектированием при геометрических искажениях, связанных с нефронтальным расположением устройства съемки относительно объекта фиксации. Существенное время работы фильтров, применяемых для улучшения качества изображения для распознавания и коррекции изображения, привело к тому, что распознавание происходит с существенным запаздыванием (до десятка секунд) от реального времени. Также было замечено, что работа алгоритмов распознавания сильно загружает процессор мобильного устройства, разряжая батарею, что делает сложным и дорогостоящим применение исследованных метода на смартфонах и планшетах.

Следующий исследовательский эксперимент был поставлен в системе ПиктоМир [3], предназначенной для обучения основам алгоритмики детей дошкольного и младшего школьного возраста. Дети, которые еще не умеют читать или не очень любящие читать, собирают алгоритм задачи из картинок-

пиктограмм на магнитной доске и, затем, фотографируют на смартфон или планшет, «отправляя» свою программу на компьютер. То есть нужно было научить смартфон распознавать фиксированный набор пиктограмм в кадре.

Несмотря на кажущуюся простоту задачи, детектирование ограниченного набора известных изображений, в задаче имелись свои сложности. Многие объекты имели несколько компонент связности, тени и другие результаты дизайна картинок также затрудняли детектирование. При использовании библиотеки OpenCV были протестированы два алгоритма детектирования объектов.

Использование базового алгоритма [4] не привело к существенному проценту распознавания, в силу описанной выше сложности изображений. Кроме того, попадание в кадр посторонних объектов с резко контрастирующими с фоном цветами, приводил к массовым ошибкам, когда инородные изображения распознавались как целевые.

Следующий алгоритм [5] позволил существенно улучшить выход успешных распознаваний - все объекты детекции имели один и тот же основной цвет — синий. Однако, в процессе применения этого алгоритма возникли проблемы с шумовой детекцией. В силу координатной близости синего и зеленого цветов в пространстве HSV, при детектировании на зеленом фоне результаты были хуже, чем на контрастных черном или белом фонах.

Модифицирования последнего алгоритма фактически привели к созданию собственного алгоритма - детектора объектов. Варьировав нижние и верхние границы отсечения в алгоритме с их последующим маскированием, позволило более точно отделить зеленый цвет от синего в трёхмерном HSV-пространстве. В результате подбора параметров были получены результаты максимальной точности на тестовых 10 классах изображений.

Однако, при дальнейшем тестировании системы выяснилось, что при изменении освещенности или появлении теней на объектах детектирования, алгоритм снижает число распознанных пиктограмм.

На основании описанных проблем алгоритма распознавания объектов на базе OpenCV было принято решение использовать технологии машинного обучения отделения базовых цветов от фоновых. Из изображений, полученных в ходе тестирования, был собран датасет, состоящий из 86000 точек в трехмерном пространстве — координат цветов, разбитый на два класса: цвет фона и цвет целевого объекта. В результате визуализации точек в трехмерном пространстве была получена картина, на которой отчетливо видна граница двух классов.

При решении данной задачи классификации были использованы методы knn с подбором параметров кроссвалидацией, и линейная модель lasso, оба метода реализованы в библиотеке sklearn. Наиболее удовлетворительные результаты на тренировочной и тестовой выборках показал метод lasso с порогом разделения, подобранным кроссвалидацией. Встраивание решения в систему детектирования объектов дало результат в точности определения объектов порядка 95%: была успешно выполнена детекция всех объектов на 1384 тестовых изображениях из 1451 (Рис.1).

Схема алгоритма приведена ниже (Рис.2).

Представленный алгоритм, тем не менее, имеет невысокий выход детектированных пиктограмм на зеленом фоне, и требует тюниговки при детектировании целевых объектов, если на изображении присутствуют посторонние детали синего цвета.

Несмотря на мощь и простоту библиотеки OpenCV использование только детерминированных алгоритмов не оправдывает себя. Дальнейшее развитие авторы видят в комбинации библиотек OpenCV и нейросетевых технологий.

Работа проводилась в рамках гранта РФФИ 18-07-00901 - Исследование и разработка системы распознавания элементов рукотворного интерьера на базе нейронных сетей для построения дополненной реальности и выработки алгоритмов взаимодействия управляемых объектов с реально-виртуальным окружением.

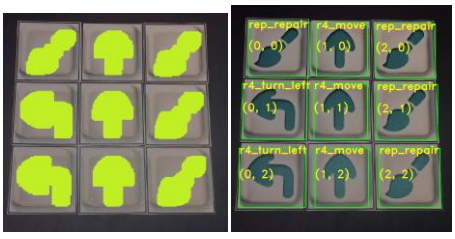


Рис.1. Результат работы алгоритма распознавания и полученные маски

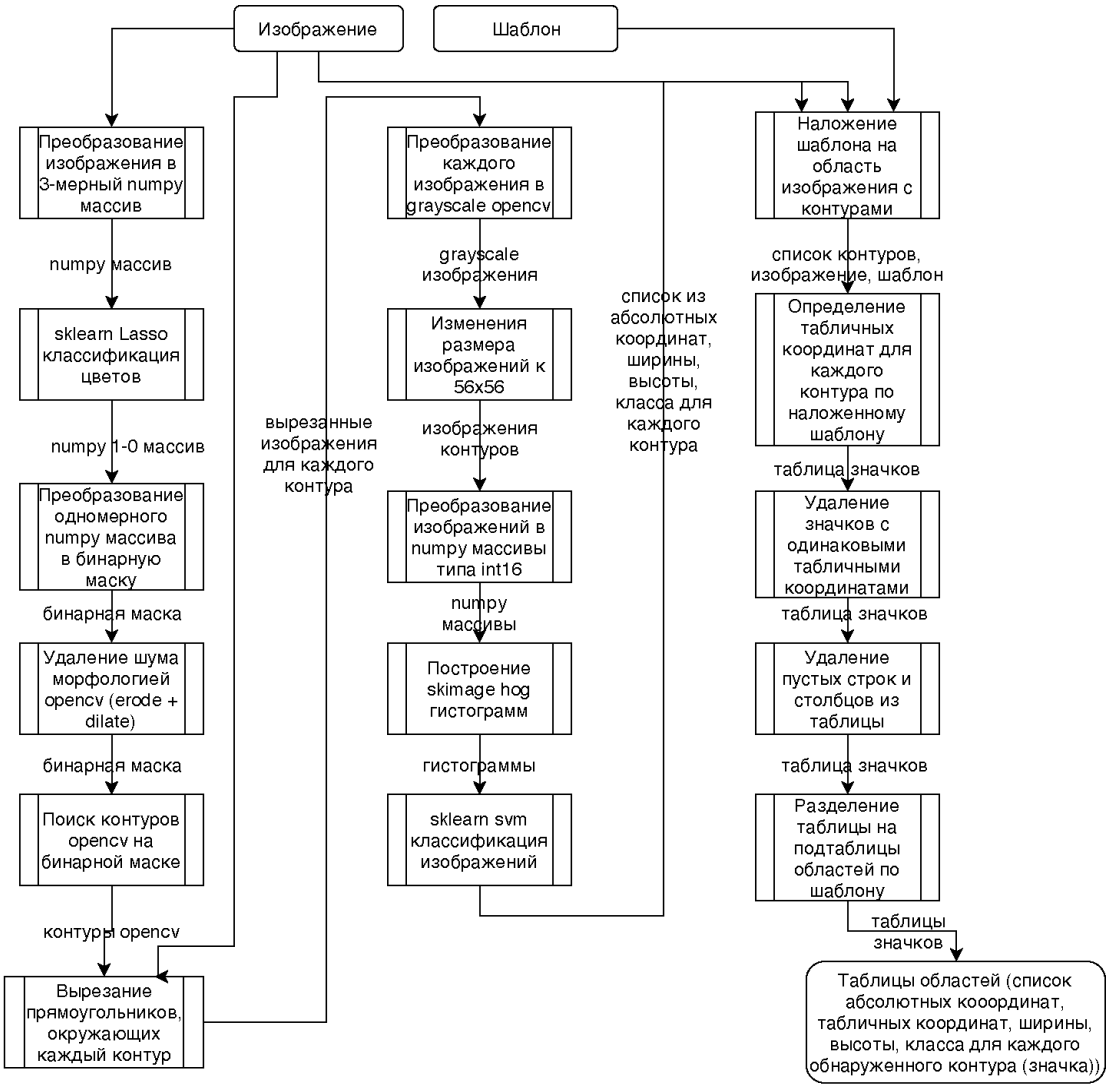


Рис.2. Схема алгоритма распознавания.

Some questions about the effectiveness of deterministic image recognition algorithms using the OpenCV library

N.O.Beshaposhnikov, M.A.Kuzmenko, A.G.Leonov, M.A.Matushin

Abstract. At present, technology of image recognition using neural networks occupies an essential part in software systems that find applications in briad areas of science and technology. However, limited computer power does not allow making these decisions massive. It is possible to use deterministic methods to ensure the effectiveness of algorithms for recognizing and combining real and virtual objects. The article examines the effectiveness of using such methods with the OpenCV library.

Keywords: OpenCV, image recognition, neural networks.

Литература

1. Open Source Computer Vision Library. [Электронный ресурс] / Режим доступа: https://opencv.org/about.html свободный.
2. Joe Minichino, Joseph Howse, Learning OpenCV 3 Computer Vision with Python - Second Edition // Packt Publishing, 2015.
3. А.Г.Кушниренко, А.Г.Леонов. Программирование для дошкольников и младших школьников. — // Информатика. — М.: Первое сент., 2011, N15. — стр.20–23
4. Image Segmentation with Watershed Algorithm. Электронный ресурс] / Режим доступа: https://docs.opencv.org/3.3.1/d3/db4/tutorial_py_watershed.html свободный.
5. Adrian Rosebrock. K Open CV and Python Color Detection // Image Processing, Tutorials: August 4, 2014 / Режим доступа: https://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/ свободный.

Моделирование датчиков в системах виртуального окружения

М.В. Михайлюк[1], Е.В. Страшнов[2], Д.М. Логинов[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1,3] mix@niisi.ras.ru, [2]strashnov_evg@mail.ru

Аннотация: В работе предлагаются методы реализации датчиков в системах виртуального окружения и имитационно-тренажерных комплексах управления сложными динамическими системами. Методы включают способы создания датчиков в конструкторе виртуальной среды, генерации показаний датчиков в подсистемах динамики или визуализации, использование показаний в функциональных схемах вычисления управляющих сигналов и передачу сигналов в подсистемы динамики и визуализации. Таким образом, реализуется обратная связь между виртуальной средой и подсистемой управления, которая повышает качество управления и адекватность системы виртуального окружения моделируемой действительности.

Ключевые слова: система виртуального окружения, датчики, обратная связь, система управления

Введение

В системах управления сложными динамическими объектами имеется обратная связь, которая реализуется с помощью датчиков. Например, используются датчики положения, угловых и линейных ускорений, магнитометры, дальномеры, датчик высоты и другие. Каждый датчик с определенной частотой передает в систему управления свои показания, которые используются частично как информационные, а частично для формирования управляющих воздействий. Реальные датчики требуют предварительной калибровки, а в процессе работы – фильтрования и усреднения информационного сигнала с целью избавления от шумов и случайных всплесков.

В соответствующих имитационно-тренажерных комплексах также необходимо моделировать датчики. При этом имеется два подхода к такому моделированию. При первом подходе моделируются все особенности и процессы, происходящие в реальных датчиках (калибровка, фильтрование, информационные протоколы и т.д., см. [1]). Это важно для получения большей реалистичности в имитационном комплексе. Однако со временем датчики совершенствуются, и актуальность имевшихся ранее проблем уменьшается. Кроме того, в имитационных тренажерах часто можно отвлечься от качества датчиков и сосредоточиться на других аспектах. В этом случае используется второй подход, при котором создаются виртуальные датчики, показания которых вычисляются моделирующим комплексом. Этот подход и рассматривается в настоящей работе. Сначала мы рассмотрим общую структуру имитационно-тренажерного комплекса, а затем опишем реализацию датчиков в нем.

1. Структура имитационно-тренажерного комплекса

Имитационно-тренажерный комплекс состоит из нескольких подсистем (см. Рис. 1):

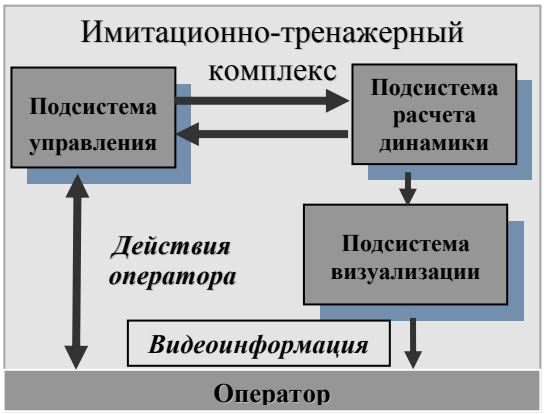


Рис 1. Структура имитационно-тренажерного комплекса

моделирования виртуальной сцены, управления, динамики и визуализации. Виртуальная сцена создается в системе трехмерного моделирования 3DS MAX из стандартных примитивов (прямоугольных

параллелепипедов, сфер, конусов и т.д.), доступных в системе и специальных объектов, добавленных в систему (колес, двигателей, системы частиц и т.д.) и содержащих динамические параметры (массы, тензоры инерции и т.д.). В это множество специальных объектов мы добавили виртуальные датчики. Система моделирования с добавленными объектами образует конструктор виртуальной сцены. Подсистема управления состоит из виртуальных пультов управления и функциональных схем для расчета управляющих сигналов. Она получает на вход оцифрованные воздействия оператора на виртуальные элементы управления и формирует на выходе управляющие сигналы (величину напряжения, подаваемого на двигатель, включение или выключение источников освещения и т.д.), которые передаются в подсистему динамики. Подсистема динамики загружает виртуальную модель и на основе динамических параметров ее объектов рассчитывает новые положения и ориентации этих объектов через промежуток времени Δt моделирования. Эти данные передаются в подсистему визуализации, которая производит синтез виртуальной сцены на мониторе оператора. Весь цикл занимает не более 40 мсек, что обеспечивает работу имитационно-тренажерного комплекса в режиме реального времени. В эту структуру мы добавляем датчики в подсистемы динамики и визуализации. На каждом шаге моделирования каждый датчик передает в систему управления значения величин, которые он измеряет. Эти значения будут использоваться в функциональных схемах для расчетов управляющих воздействий.

2. Виртуальные датчики комплекса

В этой работе мы рассматриваем следующие датчики: положения, ориентации, угловой скорости, акселерометр, датчик касания, сило-моментного очувствления, дальномер и датчик интенсивности излучения. Датчик положения определяет координаты x, y, z положения центра объекта в мировой системе координат. Датчик ориентации вычисляет углы поворота Эйлера вокруг осей системы координат. Датчик угловой скорости определяет угловые скорости $\omega_x, \omega_y, \omega_z$ вращения объекта в его локальной системе координат. Акселерометр вычисляет ускорение объекта вдоль его локальных осей координат, т.е. определяет вектор ускорения объекта в его локальной системе координат. Датчик касания определяет столкновение его рабочего органа с другими объектами виртуального окружения. Датчик сило-моментного очувствления измеряет силу и момент, которые приложены к нему каким-либо объектом внешней среды. Дальномер определяет расстояние до ближайшего виртуального объекта вдоль своей линии визира. Датчик интенсивности излучения определяет уровень интенсивности измеряемого излучения в месте нахождения датчика.

В конструкторе, добавленном в систему 3D моделирования 3DS MAX, для каждого датчика имеется соответствующий объект с параметрами или без параметров. С помощью этого объекта создается конкретный экземпляр датчика, который устанавливается на виртуальный объект сцены и привязываются к нему по иерархии. В узле иерархического дерева сцены, соответствующем датчику, хранится модельная матрица, т.е. матрица перехода от локальной системы координат датчика в мировую систему координат.

В функциональной схеме системы управления датчику соответствует блок с одним входом и соответствующим числом выходов. На вход датчика подается значение флага его работы. Если на вход подается 0, то датчик отключается, если 1 – то включается. Это позволяет временно отключать ненужные датчики. Если датчик включен, то на каждом шаге моделирования подсистема динамики (а иногда подсистема визуализации) с помощью callback функции формирует на выходах этого блока текущие значения показаний датчика.

Рассмотрим эти датчики более подробно.

Датчик положения. Этот датчик определяет координаты x, y, z своего положения в мировой системе координат. В реальности датчик положения может реализоваться с помощью системы GPS или ГЛОНАСС. В конструкторе виртуальной сцены имеется объект типа датчика положения, который не имеет параметров. В функциональной схеме ему соответствует блок с одним входом и тремя выходами, соответствующими x, y и z. Значения на этих выходах формирует подсистема динамики. Для этого используется модельная матрица (матрица перехода из локальной системы координат датчика в мировую систему координат).

Датчик ориентации. Этот датчик определяет углы поворота Эйлера, совмещающие локальную систему координат объекта с мировой системой координат. Последовательность применения углов Эйлера следующая: сначала поворот вокруг оси Y, потом вокруг оси Z и затем – вокруг оси X. Это соответствует самолетным углам, при которых первый поворот означает рысканье,

второй - тангаж, а третий - крен. В системах виртуального окружения углы Эйлера можно вычислять из модельной матрицы, однако в этом случае может возникать так называемая "блокировка кардана". Когда угол тангажа равен $\pm 90^o$, остальные углы определить однозначно невозможно [2]. Для решения этой проблемы в [3] предложено использовать контролируемый четвертый угол. Если блокировки кардана нет, этот угол равен нулю. В случае приближения угла тангажа к $\pm 90^o$ четвертый угол увеличивается до тех пор, пока блокировка исчезает. Поэтому в функциональной схеме датчику ориентации соответствует блок с одним входом и четырьмя выходами, соответствующим углам поворота. В реальных системах датчик ориентации обычно реализуется с помощью гироскопов. Ориентацию можно также вычислить из показаний датчиков угловых скоростей путем их интегрирования.

Датчик угловой скорости. Датчик определяет угловые скорости $\omega_x, \omega_y, \omega_z$ вращения вокруг осей его локальной системы координат. Реальные датчики угловой скорости (ДУС) или микромеханические гироскопы реализуются с помощью трех микромеханических датчиков угловых скоростей и вычислительного устройства, производящего интегрирование скоростей для получения углов ориентации. В системе виртуальной реальности датчик будет вычислять только угловые скорости, а для углов ориентации используется отдельный датчик. В конструкторе 3DS MAX с помощью объекта «Датчик угловой скорости» (см. Рис. 2) создается экземпляр датчика и устанавливается с нужной ориентацией на некоторый виртуальный объект. Датчик не имеет параметров, а в функциональной схеме системы управления он задается блоком с одним входом и тремя выходами, на которые выводятся значения угловых скоростей ω_x, ω_y и ω_z. Подсистема динамики на каждом шаге моделирования вычисляет вектор угловой скорости $\vec{\omega}_W = \omega_{x,W}, \omega_{y,W}, \omega_{z,W}$ датчика в

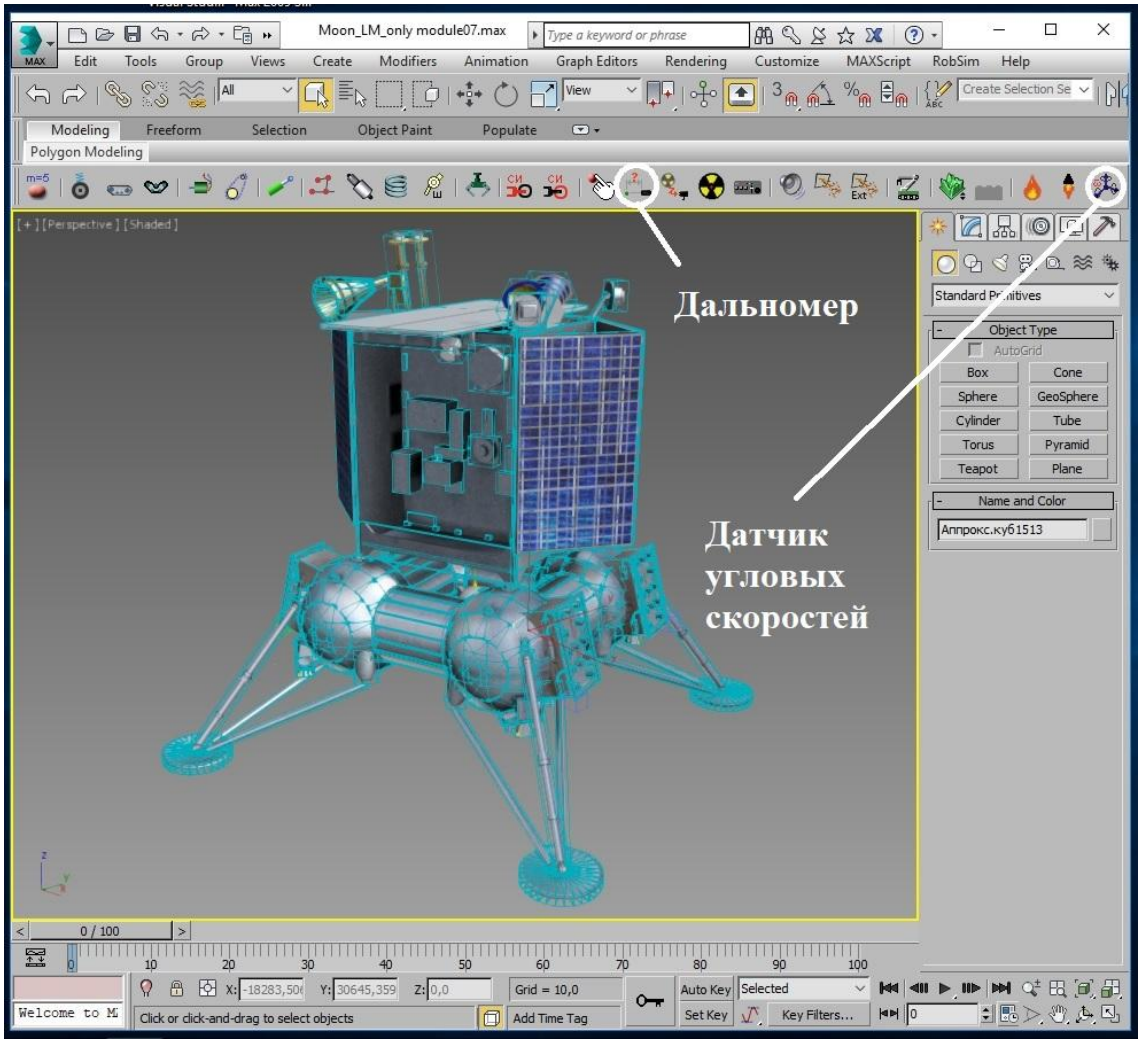


Рис. 2. Элементы датчиков на панели инструментов

мировой системе координат. Умножив этот вектор на матрицу M^{-1}, обратную к модельной, получим координаты $\omega_x, \omega_y, \omega_z$ этого вектора в локальной системе координат датчика. Эти координаты и передаются в систему управления.

Датчик ускорения (акселерометр). Мы будем рассматривать трехстепенной акселерометр, который определяет ускорение объекта вдоль его локальных осей координат, т.е. определяет вектор ускорения объекта в его локальной системе координат. При измерениях акселерометр учитывает (включает) ускорение свободного падения g=9,8 м/с^2. Он измеряет ускорение в единицах g, т.е. показание (-1) означает (-g). Параметр d задает диапазон измерений (т.е. акселерометр будет измерять ускорения в диапазоне $[-d \cdot g, d \cdot g]$).

Акселерометр создается в конструкторе и устанавливается на виртуальном объекте в центре его локальной системы координат так, чтобы оси акселерометра совпадали с соответствующими осями этой системы координат. Параметр d задается в конструкторе и передается в систему динамики один раз при загрузке сцены.

В системе управления акселерометру соответствует функциональный блок с тремя выходами (показания акселерометра). На каждом шаге моделирования на выходах формируются вычисленные акселерометром ускорения, скорректированные с учетом диапазона измерений.

В системе динамики на каждом шаге моделирования сначала вычисляется вектор ускорения в мировой системе координат в точке P расположения датчика:

$$\vec{a}_{PW} = \vec{a}_C + \vec{\varepsilon} \times \vec{r} + \vec{\omega} \times (\vec{\omega} \times \vec{r}),$$

где $\vec{a}_C$ – ускорение центра масс объекта с датчиком, $\vec{\varepsilon}$ – угловое ускорение этого объекта, $\vec{\omega}$ – его угловая скорость, $\vec{r}$ – вектор, соединяющий центр масс объекта с точкой P.

Так как для моделирования динамики тел необходимы только их скорости, то для вычисления ускорений $\vec{a}_C$ и $\vec{\varepsilon}$ воспользуемся разностными производными:

$$\vec{a}_C(t) = \frac{\vec{\mathrm{V}}_C(t) - \vec{\mathrm{V}}_C(t - \Delta t)}{\Delta t};$$

$$\vec{\varepsilon}(t) = \frac{\vec{\omega}(t) - \vec{\omega}(t - \Delta t)}{\Delta t},$$

где $\vec{\mathrm{V}}_C$ – скорость центра масс объекта, Δt – шаг моделирования.

Умножив вектор $\vec{a}_{PW}$ на матрицу M^{-1}, обратную к модельной, получим координаты $\vec{a} = (a_x, a_y, a_z)$ вектора ускорения объекта в его локальной системе координат, которые далее изменяются исходя из диапазона измерений акселерометра.

Проинтегрировав полученные показания датчика ускорения, можно получить новые значения скорости и положения объекта:

$$\vec{\mathrm{V}} = \vec{\mathrm{V}} + \vec{a} \cdot \Delta t;$$

$$P = P + \vec{\mathrm{V}} \cdot \Delta t.$$

Датчик касания. Этот датчик размещается на внешней поверхности объекта. Датчик касания реагирует только на прикосновение и не предназначен для вычисления сил. В системах виртуального окружения этот датчик представляет собой точку. Если точка попадает внутрь объекта (его ограничивающего контейнера), то датчик возвращает 1, в противном случае – 0. В функциональной схеме он представляет собой блок с одним входом и одним выходом, на котором формируется булевское значение.

Датчик силомоментного очувствления. Этот датчик измеряет силы и моменты, действующие на рабочий орган управляемого объекта (например, манипуляционного робота) при выполнении им технологических операций. Здесь мы рассмотрим три варианта размещения таких датчиков на роботах [4-6]: на схвате рабочего органа, рядом с рабочим органом (запястный датчик, который устанавливается между последним звеном и рабочим органом) и в сочленениях с электроприводами (датчики моментов в шарнирах). В реальных датчиках силы и моменты измеряются через создаваемые ими деформации упругих элементов датчика. Датчик силы схвата устанавливается непосредственно на поверхности схвата и определяет величину контактного усилия рабочего органа робота. Запястный датчик измеряет три компоненты силы и три компоненты момента и располагается между последним звеном и рабочим органом робота. В свою очередь датчик момента в шарнире измеряет момент усилия, приложенный к выходному валу привода сочленения.

Датчик силомоментного очувствления каждого из типов создается в конструкторе виртуальной сцены и устанавливается на виртуальном объекте в нужном месте. В системе управления силомоментному датчику соответствует функциональный блок с шестью выходами (три компонента силы и три компонента момента).

В системе динамики датчик силы, установленный на схвате робота, вычисляет величину силы контактного взаимодействия. Для этого сначала с помощью метода последовательных импульсов [7] в точках контакта вычисляются проекции p_z импуль-

сов сил реакции опоры на ось z, перпендикулярную поверхности датчика. Затем искомая сила F_z определяется через импульс следующим образом:

$$F_z = \frac{p_z}{\Delta t},$$

где Δt – шаг моделирования.

Для запястного силомоментного датчика искомыми величинами являются сила и момент, передаваемые через сочленение, соединяющее последнее звено с рабочим органом. Шарнир описывается [8] набором голономных связей и в общем случае ограничивает некоторые поступательные и вращательные степени свободы. Выполнение этих ограничений в системе динамики обеспечивается с помощью вычисляемых компонентов p_x, p_y, p_z импульса силы и компонентов n_x, n_y, n_z импульса момента. Переход от импульсов к силам f_x, f_y, f_z и моментам τ_x, τ_y, τ_z осуществляется аналогично датчику силы на схвате робота делением компонентов импульсов на Δt.

Датчик момента в шарнире реализован в системе динамики путем вычисления момента, передаваемого от двигателя к звену управляемого объекта. Для этого применена математическая модель динамики электропривода [9], в которой учитываются крутящий момент двигателя, момент трения на валу двигателя и моменты, приложенные к звену. С помощью этой модели и ограничения, связывающего координаты звеньев с углом поворота двигателя, вычисляется импульс момента p, с помощью которого вычисляется искомый момент τ следующим образом:

$$\tau = \frac{p}{\Delta t}.$$

Датчик расстояния (дальномер). Этот датчик измеряет расстояние до ближайшего объекта по линии своего визира. Реальные дальномеры подразделяются, по принципу действия, на ультразвуковые, инфракрасные и лазерные. Ультразвуковой датчик издает ультразвуковой импульс на определенной частоте и измеряет время его возврата. Зная скорость распространения звука в данной среде, он вычисляет расстояние до препятствия. Инфракрасный дальномер излучает узкий пучок света, который, отражаясь от препятствия, попадает на ПЗС матрицу (прибор с зарядовой связью, приемник). Зная точку на матрице, в которую попадает отраженный луч, датчик рассчитывает угол наклона принятого луча и расстояние до препятствия. Лазерный дальномер работает аналогично ультразвуковому, только вместо ультразвука используется лазер.

В конструкторе виртуальной сцены имеется объект типа дальномер, который не имеет параметров. Его локальная система координат расположена так, что измерение расстояния производится вдоль линии визирования, которой является его локальная ось Y. Экземпляр дальномера создается в конструкторе и устанавливается на нужный виртуальный объект. В схеме управления ему соответствует функциональный блок с одним выходом, на котором в каждом шаге моделирования формируется расстояние до ближайшего объекта сцены вдоль линии визирования датчика. Реализовать виртуальный дальномер проще всего в подсистеме

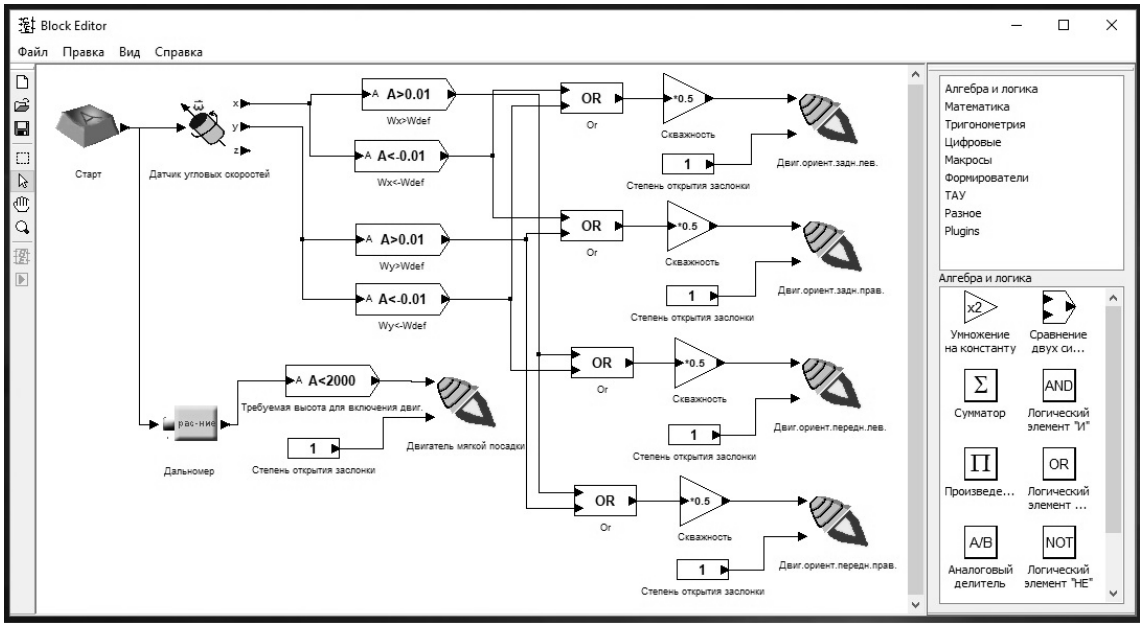


Рис. 3. Функциональная схема управления реактивными двигателями

визуализации, используя виртуальную камеру и буфер глубины.

Датчик интенсивности излучения. Датчик определяет уровень интенсивности измеряемого излучения в точке своего расположения. Можно рассматривать датчики инфракрасного, ультрафиолетового, рентгеновского и других излучений. В реальности детекторы основаны на изменении электрических или других физических характеристик чувствительной области. В системе виртуального окружения интенсивность излучения в точке расположения датчика определяется как сумма интенсивностей от всех источников данного типа в этой точке с учетом их затухания в зависимости от расстояния.

В конструкторе имеются объекты датчиков различного типа излучений, которым в функциональной схеме соответствуют блоки с одним входом и одним выходом, на котором подсистема динамики формирует величину интенсивности измеряемого излучения.

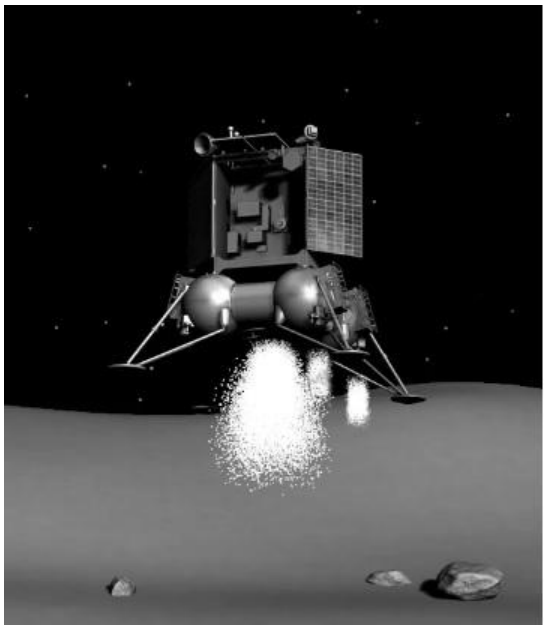


Рис. 4. Моделирование посадки лунного модуля

Заключение

В результате данного исследования предложен метод реализации датчиков различных типов в системах виртуального окружения. Эти датчики обеспечивают обратную связь между состоянием окружающей среды и системой управления, что позволяет существенно улучшить управление виртуальными объектами и приблизить его к реальной теории автоматического управления. Предложенный метод и алгоритмы реализованы и прошли апробацию в комплексе VirSim, разработанном в ФГУ ФНЦ НИИСИ РАН.

Эксперименты показали адекватность их реализации задачам и требованиям имитационно-тренажерных комплексов и систем виртуального окружения. На Рис. 3 приведен пример функциональной схемы управления реактивными двигателями с применением виртуальных датчиков (датчика угловых скоростей и дальномера).

С помощью этой функциональной схемы выполняется моделирование (см. Рис. 4) посадки космического модуля на поверхность Луны.

Для поддержания вертикального положения использовались датчики ориентации и угловых скоростей.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «34.9. Системы виртуального окружения: технологии, методы и алгоритмы математического моделирования и визуализации». (0065-2018-0020).

Sensors Simulation in Virtual Environment Systems

M.V. Mikhaylyuk, E.V. Strashnov, D.M. Loginov

Abstract: The paper proposes the methods for sensors implementation in virtual environment systems and simulators for control of complex dynamic objects. The methods include the means of sensors creation in a virtual environment constructor, generation of sensors indications in dynamics or visualization subsystem, use the indications in functional schemes for control signals calculation and signals transmission to dynamics or visualization subsystem. Thus, the feedback between virtual environment and control subsystem is realized, that improves control quality and the adequacy of virtual environment system to simulated reality.

Keywords: virtual environment system, sensors, feedback, control system.

Литература

1. Д.В. Евстигнеев. Системы управления интеллектуальны мобильных роботов в среде Dyn-Soft RobSim 5: Учеб. пособие, 2014, 190 с.
2. J. Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors, Technical Report, Stanford University, 2006.
3. М.В. Михайлюк, Е.В. Страшнов. Ориентация объектов в системах виртуального окружения. // Труды НИИСИ РАН, т.8, № 1, 2018, стр. 48-51.
4. М. Шахипнур. Курс робототехники. Пер. с англ. – М.: Мир, 1990. – 527 с.
5. Д.М. Гориневский, А.М. Формальский, А.Ю. Шнейдер. Управление манипуляционными системами на основе информации об усилиях/ Под ред. В.С. Гурфинкеля и Е.А. Девянина. – М.: Физматлит, 1994. – 368 с.
6. Е.И. Юревич. Сенсорные системы в робототехнике: учеб. пособие / Е.И. Юревич. – СПб.: Изд-во Политехн. ун-та, 2013. – 100 с.
7. А.М. Трушин. Обработка коллизий виртуальных объектов с помощью метода последовательных импульсов // Труды НИИСИ РАН, 2014, том 4, № 2, стр. 95-105
8. М.В. Михайлюк, Е.В. Страшнов. Моделирование системы связанных тел методом последовательных импульсов // Труды НИИСИ РАН, 2014, том 4 № 2, стр. 52-60.
9. Е.В. Страшнов, М.А. Торгашев. Моделирование динамики электроприводов виртуальных роботов в имитационно-тренажерных комплексах // Издательство “Новые технологии”, Мехатроника, автоматизация, управление, 2016, том 17, № 11, стр. 762-768.

Синтез сценариев управления чрезвычайными ситуациями

Д.А. Кононов[1], Р.О. Пономарев[2]

ИПУ РАН и РГГУ, Москва, Россия, E-mail's: [1]dmitrykon52@gmail.com, [2]roman3457@yandex.ru

Аннотация: Предложен метод сценарного синтеза эффективного сценария для предотвращения чрезвычайных ситуаций в сложных организационно-технических системах на основе исследования базисных режимов их функционирования. Приведен пример сценарного синтеза в модели «городская инфраструктура». Разработаны вычислительные алгоритмы реализации метода. Обсуждаются результаты расчетов.

Ключевые слова: сценарный синтез, уязвимость, городская инфраструктура, вычислительный эксперимент.

Введение

Чрезвычайные ситуации (ЧС) – это незапланированные и неожиданные события, которые требуют срочного применения определенных компетенций, ресурсов или процессов для предотвращения или снижения их фактических или потенциальных последствий. Чрезвычайные ситуации могут приводить к негативным экологическим последствиям или иным последствиям для общества. При определении возможных чрезвычайных ситуаций (например, пожар, разлив химикатов, экстремальные метеоусловия) следует принять во внимание:
– характер имеющихся на местах опасностей (например, легковоспламеняющиеся жидкости, резервуары, баллоны со сжатым газом);
– наиболее вероятный вид и масштаб чрезвычайной ситуации;
– возможность того, что чрезвычайная ситуация затронет близлежащие объекты (например, предприятия, дороги, железнодорожные пути).

Общетеоретическая модель сценарного исследования уязвимости и поведения сложной системы в условиях ЧС для прикладного применения требует подробного описания основных элементов проблемной ситуации. Для реализации этой задачи следует провести комплекс работ:
– сформулировать проблему исследования;
– построить модель объекта;
– определить и формализовать экспертно-значимые события;
– определить и формализовать перечень угроз;
– формализовать сценарные характеристики объекта исследования;
– разработать программу сценарного анализа (вычислительного эксперимента);
– разработать алгоритмы и программы расчета характеристик;
– провести необходимые расчеты;
– сформулировать выводы анализа.

В [0] рассмотрены возможности применения общей методологии уязвимости для модели поведения сложной системы «операторные графы» на динамической модели «Городская инфраструктура» [0, 0].

Модель городской инфраструктуры имеет следующую графовую структуру, вершинами которой являются взаимосвязанные объекты:
1. Населённый пункт; 2. Лесной массив; 3. Промышленное предприятие; 4. Экология; 5. Транспортный узел; 6. Склад; 7. Энергосеть; 8. Водохранилище.

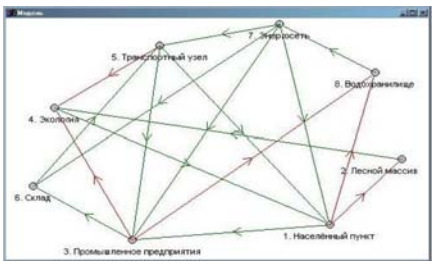


Рис 1. Граф городской инфраструктуры.

Исследование уязвимости графовой модели заключается в поиске «мест уязвимости», при этом каждая вершина может являться носителем потенциальной угрозы. Степень уязвимости каждой компоненты модели можно описать на основе характеристики «*T*-шаговая угроза». В работе [0] рассматривается задача определения наиболее уязвимых элементов системы от источников угроз.

Моделирование поведения объекта, связанного с возникновением и развитием новой ситуации, позволяет прогнозировать момент времени (шаг сценария) первого выхода значения в вершине за границы целевого режима функционирования (ЦРФ) в отсутствие управляющих воздействий.

Для построения матрицы *T*-шаговых угроз начальный импульс, представляющий угрозу, реализуется последовательно в каждой вершине. Строятся сценарии С2-1…С8-1.и составляется матрица МТ-1 *T*-шаговых угроз [0]. Для исследуемой модели она такова:

Таблица 1. Матрица *T*-шаговых угроз (МТ-1).

Вер.	С1	С2	С3	С4	С5	С6	С7	С8
1	19	22	19	23	25	20	23	21
2	13	17	14	12	14	14	13	11
3	30	29	31	29	32	32	30	28
4	7	9	6	10	7	7	11	9
5	18	16	19	17	19	19	17	15
6	19	23	20	18	21	21	19	17
7	19	23	20	18	21	20	19	17
8	20	18	21	18	21	21	19	17

По матрице *T*-шаговых угроз определяется, когда возможно наступление ЧС, а также к какому классу уязвимости элементов сложной системы относится данный ее элемент.

Следующая задача управления – синтез оптимального сценария управления с тем, чтобы либо избежать ЧС, либо отложить их реализацию во времени. Цель управляющего воздействия заключается в том, чтобы увеличить стойкость целевых элементов таким образом, чтобы отклонение от ЦРФ этих элементов происходило на более позднем шаге сценария, чем шаг сценария соответствующего элемента, показанный в матрицы Т-шаговых угроз.

1. Модели угроз

Угрозой $\gamma\in\Omega^{(tr)}$ назовем фактор (явление), реализация которого может привести к гипотетической возможности реализации нежелательных явлений (ситуаций).

При описании сложной системы (СС) в виде формального системного объекта под угрозой будем понимать совокупность факторов системы и внешней среды, могущих привести с точки зрения исследователя или лица, принимающего решения (ЛПР), к существенному ухудшению состояния ее параметров, в том числе ЧС.

Пусть $\Omega^{(tri)}$ – область недопустимых событий, т.е. условие $\Im\in\Omega^{(tri)}$ определяет чрезвычайную ситуацию.

Рассмотрим два типа моделей угроз.

Угроза-возмущение.

В модели «Угроза-возмущение» зададим множество возможных угроз $\Omega^{(tr)}$, т.е. возможных несанкционированных изменений расширенного состояния $\Im$ системы *S*. В данном случае возникает вопрос об отклонении системы от ЦРФ, попадании в область $\Omega^{(tri)}$ при их реализации (прямая задача управления).

Угроза-ЧС.

В модели «Угроза-ЧС» угрозой назовем возможность наступления нежелательных событий, т.е. возможность появления пути событий $W(\Im_0, \Im_N, N)$, при котором $\Im_N\in\Omega^{(tri)}$ – области недопустимых событий. Таким образом, в модели «Угроза-ЧС» априори задано множество недопустимых событий $\Omega^{(tri)}$, и возникает вопрос о поиске нежелательных возмущений, которые их генерируют, т.е. угроз в смысле модели «Угроза-возмущение» (обратная задача управления).

2. Сценарный синтез на основе исследования базисных режмов

Множественность сценариев поведения на операторном орграфе определяется стратегиями управления.

В [0] предложена методика анализа поведения СС на основе исследования импульсных управлений.

В предлагаемой работе рассмотрим построение эффективных управлений на основе анализа «внутренней структуры» СС.

Рассмотрим матрицу смежности *A* орграфа как линейный оператор, действующий в евклидовом пространстве E^n. Выберем канонический базис *B* в этом пространстве, в котором матрица *A* представлена верхней жордановой формой. Тогда определены собственные значения в виде:

$\lambda_\xi = \mu_\xi \exp(i\varphi_\xi) = \mu_\xi(\cos\varphi_\xi + i\sin\varphi_\xi)$ для всех $\xi=1,\ldots,s$;

где i – мнимая единица, μ_ξ – длина, а φ_ξ – фаза комплексного числа λ_ξ.

Любой сценарий в модели является линейной комбинацией базисных сценариев развития, определяемых жордановой формой матрицы *A*. Конечное число типов базисных сценариев определяет теорема 4 в [0]:

– если $\varphi_\xi=0$, то соответствующий сценарий при $\mu_\xi > 1$ является сценарием резонансного развития (РР-сценарием), при $\mu_\xi < 1$ является сценарием сжимающего развития (СР-сценарием);

– если $\varphi_\xi \neq 0$, то соответствующий сценарий при $\mu_\xi = 1$ является сценарий колеблющегося развития (КР-сценарием), при $\mu_\xi > 1$ является сценарий резонансного колеблющегося (РК-сценарием) развития, при $\mu_\xi < 1$ является сценарий сжимающего колеблющегося (СК-сценарием) развития;
– если $\lambda_\xi = 1$, то сценарий является стационарным (СтР), а при $\lambda_\xi = -1$ сценарий является сценарием отражения.

В приложениях существенно важными являются предельные случаи:
– стационарный сценарий (СС-сценарий): $\mu_\xi = 0$;
– сценарий отражения (СО-сценарий): $\varphi_\xi = \pi$;
– сценарий циклического развития (Ц-сценарий): $\varphi_\xi = \pi/m$, m – период развития.

В то же время по критерию *характерного вида внутреннего (скрытого) устройства структуры системы* могут быть выделены следующие типы базисных структур динамической системы:
– простые структуры (ПС): $p(\xi)=1$;
– действительные структуры (РС): $\varphi_\xi = 0$;
– дублированные структуры (ДС): $p(\xi)>1$;
– комплексные структуры (КС): $\varphi_\xi \neq 0$;
– многократно дублированные структуры (МДС): существуют ξ и ζ, что $\xi \neq \zeta$, $\lambda_\xi = \lambda_\zeta$. и $\min[p(\xi), p(\zeta)]>1$.

Сценарный синтез на основе исследования базисных режимов решает следующую задачу: найти управляющие воздействия, которые компенсируют воздействие поражающего фактора на систему таким образом, чтобы функционирование системы осуществлялось в режиме безопасности.

Исследование жордановой формы матрицы смежности позволяет локализовать поиск управляющих воздействий путем расчета базисных режимов.

Расчёт базисных режимов позволяет предоставить ЛПР необходимую информацию. Зная возможный спектр функционирования системы, можно выбрать такое управляющее воздействие, которое позволит перейти к желаемому базисному режиму.

При качественной оценке характеристик наблюдаемых параметров можно перейти к решению обратной задачи – поиску эффективных управленческих решений предотвращения и ликвидации ЧС.

Исследование базисных режимов позволяет определить такое управление сложной системой, которое обеспечивает безопасный режим функционирования, соответствующий выбранной комбинации безопасных режимов.

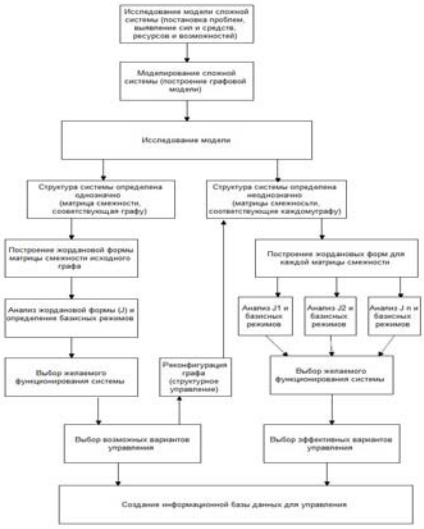


Рис.2. Схема исследования поведения сложной системы на основе исследования базисных режимов.

Исследование жордановой формы позволяет:
– определить количество и виды базисных сценариев;
– выбрать базисный сценарий, который будет соответствовать желаемому развитию системы;
– осуществить управление таким образом, чтобы получить желаемый сценарий;
– определить, что следует изменить в структуре графа с тем, чтобы получить новую динамическую модель, в которой будут только желательные базисные режимы.

3. Вычислительный эксперимент

Цели и задачи эксперимента.

Цель исследования заключается в определении такого управления, чтобы в случае воздействия на систему различного типа угроз обеспечить безопасный режим её функционирования с учётом заданных ограничений на основе анализа базисных режимов.

При надлежащем управлении, смягчающем воздействие поражающего фактора, влияющего на систему, можно обеспечить увеличение живучести системы в течение некоторого периода времени.

Задачи вычислительного эксперимента.

Пусть известны базисные режимы, в которых функционирует сложная система, следует:
– проанализировать жорданову форму;
– определить нежелательные базисные режимы (жордановы клетки);
– определить желательные базисные режимы;

– найти эффективное управление, которое уменьшает возможность возникновения или ликвидирует ЧС.

Вычисления проводятся по следующему алгоритму.

Алгоритм поиска эффективного управления:

Этап 1. Ввод данных:
– Матрица смежности A;
– Жорданова форма J матрицы смежности A;
– Матрица T-шаговых угроз (MT).

Этап 2. Вычисление базисных режимов по жордановой форме матрицы смежности, анализ базисных режимов, определение желательных и нежелательных.

Этап 3. Анализ матрицы T-шаговых угроз выбор наиболее опасной угрозы и наиболее уязвимого элемента.

Этап 4. Определение ограничений для решения обратной задачи управления.

Этап 5. Выбор критерия эффективности.

Этап 6. Разработка программы решения обратной задачи управления.

Этап 7. Проведение расчёта эффективного управления с выбранными данными и критериями.

Этап 8. Вывод данных управляющих воздействий в виде таблицы.

Этап 9. Анализ результатов.

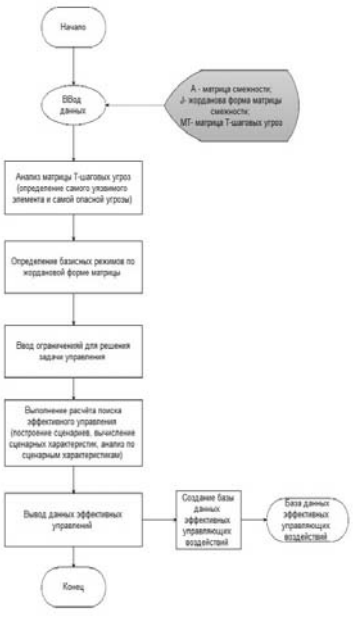


Рис. 3. Алгоритм поиска управляющих воздействий.

Результаты эксперимента

По заданному графу строим матрицу смежности.

Таблица 2. Матрица смежности модели «Городская инфраструктура».

Вер.	1	2	3	4	5	6	7	8
1	0	-1	1	0	0	0	0	-1
2	0	0	0	1	0	0	0	0
3	0	0	0	-1	0	1	0	-1
4	1	0	0	0	0	0	0	0
5	1	0	1	-1	0	0	0	0
6	0	0	0	0	1	0	0	0
7	1	0	1	0	1	1	0	0
8	0	0	0	0	0	0	1	0

Представим матрицу смежности в жордановой форме.

Таблица 3. Жорданова форма матрицы смежности модели «Городская инфраструктура».

-1,3	0	0	0	0	0	0	0
0	-0,8	0	0	0	0	0	0
0	0	0,4	0	0	0	0	0
0	0	0	0,5	0	0	0	0
0	0	0	0	-0,4	0,9	0	0
0	0	0	0	-0,9	-0,4	0	0
0	0	0	0	0	0	0,9	-1,3
0	0	0	0	0	0	1,3	0,9

Анализ базисных режимов.

Проведён анализ модели «Городская инфраструктура» (Рис 1). В модели 6 жордановых клеток. В Таблице 4. приведены базисные режимы развития для исследуемой модели.

Таблица 4. Анализ модели «Городская инфраструктура» по жордановым клеткам.

№ Жордановой клетки	Размер	ρ_i	φ_i, рад	Режим
1	1x1	-1,37	0	РР
2	1x1	-0,85	0	СР
3	1x1	0,46	0	СР
4	1x1	0,58	0	СР
5	2x2	1,08	1,95	РК
6	2x2	1,64	0,92	РК

Сценарий, обладающий самым большим значением по модулю, является определяющим базисным режимом для данного графа. Определяющим является сценарий РР (жорданова клетка 1), который указывает на неограниченное изменение значения параметров в вершине. Сценарий сжимающего развития (СР-сценарием) указывает на тенденцию стабилизации или поглощения значения параметра в вершине [0]. Такой сценарий соответствуют жордановым

клеткам 2, 3 и 4. Комплексно сопряжённые собственные значения объединяются в жорданову клетку порядка два. Такая клетка порождает (РК-сценарий) сценарий резонансного колеблющегося развития (жордановы клетки 5 и 6).Сценарный анализ на основе базисных режимов решает следующую задачу:

Задача. Найти такие управляющие воздействия, которые компенсируют воздействия поражающего фактора на систему таким образом, чтобы функционирование системы осуществляется в одном из желаемых базисных режимах (или комбинации базисных режимов), обеспечивающих безопасное функционирование системы.

В соответствии с алгоритмом проводятся расчёты для определения возможного управления, так чтобы обеспечить смягчение воздействия соответствующей угрозы и увеличения безопасного горизонта сценария. По матрице *T*- шаговых угроз выбираются самые опасные угрозы: С1-1, С3-1, С5-1, С6-1. Полученные эффективные управляющие воздействия, позволяют определить, каким образом распределить ресурсы между вершинами графа, чтобы уменьшить вероятность возникновения ЧС в исследуемой сложной системе. Полученные результаты показывают, что при распространении по системе поражающего фактора от различных источников угроз (вершина 1. Населённый пункт – социальная нестабильность; вершина 3. Промышленное предприятие – загрезнение окружающей среды промышленными отходами; вершина 5. Транспортный узел – сбой графика движения транспотных грузов) складывается кризисная ситуация.

Для улучшения сложившейся ситуации и обеспечения отклонения параметров вершин от ЦРФ на более познем шаге сценария найдены эффективные управления.

При распространении по системе поражающего фактора от источника опасной угрозы (вершина 6. Склад – нарушение правильного хранения химикатов) складывается неконтролируемая ситуация, для которой не найдены эффективные управления, которые улучшают ситуацию.

Заключение

Расчёты показывают, что не всегда можно определить эффективные управления, обеспечивающие желаемое состояние системы. Следующий этап исследования для решения поставленной задачи – рассмотреть возможность реконфигурации структуры системы таким образом, чтобы найти спектр управляющих воздействий, компенсирующих поражающий фактор, и функционирование системы на заданном горизонте сценария не приводило к неконтролируемой ситуации, за которой может последовать ЧС.

Synthesis of emergency control scenarios

D.A. Kononov, R.O. Ponomarev

Abstract: A method for scenario synthesis of an effective scenario for preventing emergencies in complex organizational and technical systems based on research of the basic modes of their functioning is proposed. An example of scenario synthesis in the model "urban infrastructure" is given. Computational algorithms for the implementation of the method have been developed. The results of calculations are discussed.

Keywords: scenario synthesis, vulnerability, urban infrastructure, simulation experiment.

Литература

1. Р.О. Пономарев, Д.А. Кононов, Н.О. Пономарев Сценарное исследование сложных систем: уязвимость и чрезвычайные ситуации /Труды НИИСИ РАН «Математическое и компьютерное моделирование сложных систем: теоретические и прикладные аспекты» – М.: НИИСИ РАН, Т. 6, № 2, 2016, с.21–25 .

2. Д.А. Кононов, А.А. Кочкаров, Н.О. Пономарев. Стойкость сложных технических систем: сценарный взгляд на проблему /Труды НИИСИ РАН. – М.: НИИСИ РАН, 2012, С. 80-86.

3. Модели и методы анализа и синтеза сценариев развития социально-экономических систем. Кн. 2 /Под редакцией чл.-корр. РАН Шульца В.Л., д.т.н., проф. Кульбы В.В. – М., Наука, 2012. (коллективная монография). 358 с.

4. Р.О. Пономарев, Д.А. Кононов, И.М. Хасанов. Сценарный анализ уязвимости сложной системы: вычислительный эксперимент /Труды НИИСИ РАН. – М.: ФГУ ФНЦ НИИСИ РАН, 2016. Т. 6, № 2. С. 26-33.

Новые функции управления чрезвычайными ситуациями в сложных организационно-технических системах

Р.О. Пономарев[1], **Д.А. Кононов**[2]

ИПУ РАН и РГГУ, Москва, Россия, E-mail's: [1] *roman3457@yandex.ru,* [2] *dmitrykon52@gmail.com*

Аннотация. Рассмотрены основные режимы управления чрезвычайными ситуациями, а также факторы, которые влияют на принятие управленческого решения в зависимости от сложившейся ситуации. Предложены новые функции управления, осуществляемые в системах управления чрезвычайными ситуациями и основанные на ситуационном анализе при сценарном исследовании поведения изучаемого объекта. Рассмотрены задачи управления, обеспечивающие реализацию предлагаемых функций.

Ключевые слова: ситуационное моделирование, чрезвычайные ситуации, управление.

Введение

Управление чрезвычайными ситуациями (ЧС) обеспечивается единой государственной (Российской) системой предупреждения и ликвидации ЧС (РСЧС), а также Федеральным Законом о предупреждении и ликвидации ЧС [1]. РСЧС объединяет органы государственного управления РФ всех уровней, различные общественные организации, в компетенцию которых входят функции, связанные с обеспечением безопасности и защиты населения, предупреждением, реагированием и действиями в ЧС. РСЧС обеспечивает координацию сил и средств этих органов управления, материальных и культурных ценностей при возникновении аварий, катастроф, стихийных бедствий и применение возможным противником современных средств поражения. РСЧС включает территориальные и функциональные подсистемы и имеет пять уровней: объектовый, местный, территориальный, региональный, федеральный.

Проблема разработки эффективных систем управления чрезвычайными ситуациями (АСУ ЧС) представляет собой важную, но сложную задачу. С одной стороны, ее важность определяется все более усложняющимися устройствами, механизмами, предметами и средствами труда, с другой – применяемыми технологиями и условиями их функционирования [2]. Все это требует новых средств исследования, в том числе ситуационных методов моделирования, основанных на сценарном исследовании сложных систем. На основе введенного понятия «целевой режим функционирования» (ЦРФ) в работе [3] предложена вербальная классификация ситуаций, которые могут возникнуть с точки зрения лица, принимающего решения (ЛПР), при функционировании и развитии сложной организационно-технической системы ОТС [4]: безопасная, нештатная, кризисная, критическая, неконтролируемая, чрезвычайная.

При ситуационном моделировании каждая из них возникает в результате реализации угрозы $\gamma \in \Gamma^{(У,\alpha)}$ ($\Gamma^{(У,\alpha)}$ – множество угроз), при этом наносится ущерб элементам системы. Выявление источников угроз и возможностей их отражения является главными целями создания АСУ ЧС.

Крайне нежелательной ситуацией реализации угрозы является возникновение чрезвычайной ситуации. Поэтому первая задача системы управления – обнаружение указанных состояний при превентивном исследовании ее поведения. Вторая задача – прогнозирование последствий реализации угроз. Третья задача – исследование возможности перевода менее желательной (с точки зрения ЛПР) ситуации в более желательную в соответствии с указанной иерархией. Четвертая задача – поиск оптимальных способов (управляющих воздействий) такого перевода.

Защищённость заданного системного элемента (СЭ) сложной системы (СС) – это способность системы частично или полностью восстанавливать ЦРФ при реализации угрозы по отношению к указанной компоненте [5].

В настоящей работе рассмотрены стандартные функции, осуществляемые в АСУ ЧС, а также предложены новые, основанные на проведении ситуационного анализа при сценарном исследовании поведения изучаемого объекта.

Разработанная методология позволит более эффективно решать широкий спектр задач управления процессами предупреждения и ликвидации последствий возникновения ЧС на стратегическом, тактическом и оперативном уровнях.

Предлагаемые методы обеспечивают возможность комплексного многоаспектного исследования альтернативных вариантов развития ЧС при заданных целевых и критериальных установках в условиях неопределённости.

Основным преимуществом такого подхода является возможность прогнозирования тенденций развития ЧС путём формирования альтернативных сценариев [6, 2].

1. Режимы управления управления чрезвычайными ситуациями

Обеспечение необходимого уровня безопасности функционирования и развития объекта осуществляется выбором эффективных управляющих воздействий и превентивных мероприятий, направленных на снижение тяжести последствий от реализации угроз (например, снижение интенсивности воздействий). Эффективность планирования и проведения комплекса работ по нивелированию последствий ЧС различного типа существенно зависит от качества исследования поведения объекта, на основе которых формируются и реализуются превентивные мероприятия и оперативные решения.

Стандартные режимы управления чрезвычайными ситуациями.

Выделяют 4 основных режима управления чрезвычайными ситуациями [7].

1. **Повседневной деятельности** – характеризуется отсутствием информации о явных признаках угрозы возникновения ЧС.

2. **Повышенной готовности** – характеризуется наличием информации о признаках потенциальной угрозы возникновения ЧС.

3. **Чрезвычайный** – характеризуется обстоятельствами, совокупность которых в соответствии с существующими нормативами определяется как чрезвычайная ситуация.

4. **Ликвидация последствий ЧС** – характеризуется отсутствием активных поражающих факторов ЧС.

Основные функции управления чрезвычайными, сгруппированные порежимно, представлены на Рис. 1 – Рис. 4.

Реализация рассмотренных функций управления базируется на исследовании поведения объекта на основе построения его динамической модели, которая описывает динамику изменения обстановки.

Модель управления системой позволяет в каждый момент времени задавать ограничения на ее состояния, выход за которые может привести к заданным нежелательным состояниям, в том числе ЧС.

Проводя анализ перехода из одного состояния в другие, следует отследить типичные характеристики динамики, что позволяет сформировать типовые или базисные сценарии изменения обстановки.

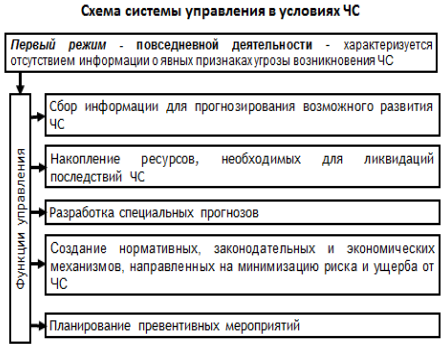


Рис. 1. Стандартные функции управления в режиме повседневной деятельности.

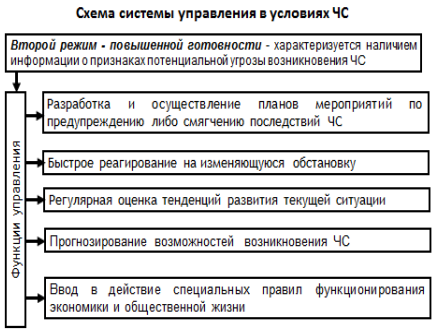


Рис. 2. Стандартные функции управления в режиме повышенной готовности.

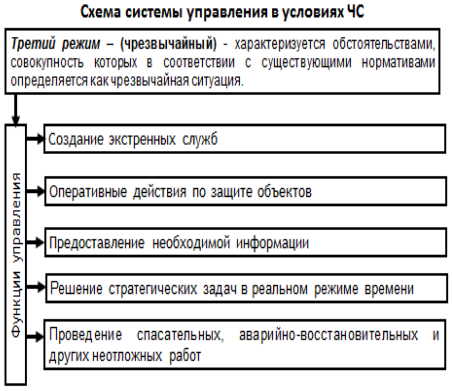


Рис. 3. Стандартные функции управления в чрезвычайном режиме.

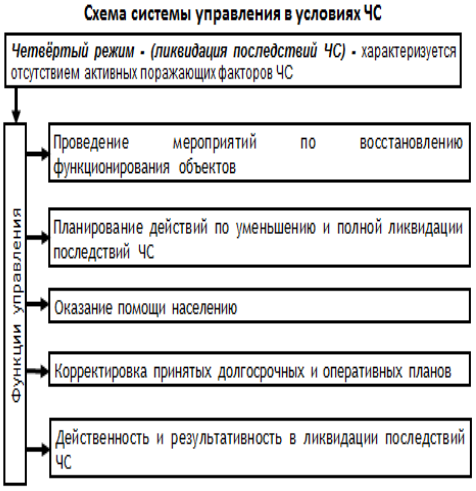


Рис. 4. Стандартные функции управления в режиме ликвидации последствий.

2. Модификация функции управления чрезвычайными ситуациями

В прикладных задачах исследования развития ЧС требуется изучать поведение сложных систем, в том числе техногенных, экологических, социально-экономических и т.п. Элементы необходимой информации, а также способы ее выделения, получения и обработки зависят от целей и аппарата исследования. В любом случае изучается модельное представление системы, а в силу дискретности измерения получают сведения о выделенных дискретных ее состояниях.

Следовательно, любой прогноз поведения фактически является некоторым сценарием развития системы, определяемым теми предпосылками, на которые опирается исследователь.

Генерацию сценариев различного назначения в автоматизированных системах управления чрезвычайными ситуациями (АСУ ЧС) можно осуществить в рамках формализованного сценарного анализа, опирающегося на элементы сценарного исчисления [4].

Каждый сценарий развития объекта – синтез моделей изменения обстановки.

Они отображают возникновение и развитие процесса преобразования его параметров и условий функционирования.

Проводится ситуационное описание, фиксирующее принципиальные с точки зрения исследователя моменты перехода в новое качественное состояние [4].

Таким образом, предлагается дополнить функции и задачи управления в рассмотренных режимах следующими компонентами (Рис. 5–Рис. 8).

Новые задачи управления в режиме повседневной деятельности.

Мониторинг критически важных и опасных объектов – процесс инструментального автоматизированного круглосуточного наблюдения за заданными параметрами объекта.

Цель мониторинга – слежение за динамикой поведения системы.

Мониторинг в режиме повседневной деятельности – отслеживание текущих значений параметров и фиксация их отклонений от заданных, т.е. установление рассогласования функционирования заданных системных элементов объекта от ЦРФ.

Мониторинг следует использовать в процессе принятия решения как инструмент обратной связи в целях осуществления эффективного управления и предотвращения наступления негативного события.

Для повышения эффективности реализации функций управления в режиме повседневной деятельности предлагается ввести функции прогнозирования сценарной стойкости, живучести и уязвимости системы, выполняемые на основе методов сценарного анализа.

В соответствии со сгенерированными сценариями функционирования объекта при анализе заданных поражающих факторов следует рассчитать характеристики прогнозируемых состояний и сформировать специальные базы данных сценарных

характеристик указанных системных параметров.

Это позволит создать оперативную базу данных базовых обстановок, нежелательное развитие которых может привести к ЧС.

Рис. 5. Предлагаемые функции управления в режиме повседневной деятельности.

Для ее создания в соответствии с [3] будем следовать следующим определениям.

Стойкость – системный параметр, характеризующий способность противостоять возмущениям, в том числе непосредственным воздействиям, и функционировать в ЦРФ в условиях реализации угроз. Основной характеристикой стойкости системы служит время достижения системой предельного состояния безопасности.

Под возмущением понимается реализация внешней или внутренней угрозы из множества предполагаемых угроз. Источником внутренней угрозы может являться любой потенциально опасный ее элемент. Внешняя угроза представляет собой угрозу, исходящую от объектов внешнего окружения системы.

Живучесть – системный параметр, свойство системы, характеризующее ее способность функционировать под влиянием возмущений, возникающих в процессе ее функционирования с учетом возможности восстановления ЦРФ. Живучесть учитывает возможность самовосстановления при реализации угрозы или управляемую восстанавливаемость повреждённых элементов во время кризисной ситуации. Живучесть системы не допускает перехода критической ситуации в ЧС, если обеспечено надлежащее управление.

Характеристики стойкости и живучести определяются значениями и структурой показателей, а также динамикой изменения системных элементов окружающей среды.

Уязвимость – системный параметр, характеризующий возможность нанесения описываемой системе повреждений любой природы, нарушающих ЦРФ. Характерной особенностью уязвимости являются компоненты системы, которые могут быть подвергнуты атаке и реализованы угрозы («окна уязвимости») [8].

Характеристикой уязвимости является коэффициент $C^{(\alpha)}(\gamma, \Delta)$ уязвимости системного элемента $S^{(\alpha)}$ при реализации угрозы $\gamma \in \Gamma^{(\alpha)}$ на горизонте управления Δ, который представляет собой величину ущерба $W^{(\alpha)}(\gamma, \Delta)$.

Таким образом могут быть вычислены

– минимальный коэффициент уязвимости для множества угроз $\Gamma^{(\alpha)}$

$$C^{(\alpha,\,\min)}(\gamma, \Delta) = \min_{\gamma \in \Gamma^{(\alpha)}} C_{\alpha}(\gamma, \Delta);$$

– максимальный коэффициент уязвимости для множества угроз $\Gamma^{(\alpha)}$

$$C^{(\alpha,\,\max)}(\gamma, \Delta) = \max_{\gamma \in \Gamma^{(\alpha)}} C^{(\alpha)}(\gamma, \Delta);$$

– минимальный коэффициент уязвимости для элементов системы

$$C^{(\min)}(\gamma, \Delta) = \min_{\alpha \in \bar{A}} C_{\alpha}^{(\min)}(\gamma, \Delta);$$

– максимальный коэффициент уязвимости для элементов системы

$$C^{(\max)}(\gamma, \Delta) = \max_{\alpha \in \bar{A}} C^{(\alpha)}(\gamma, \Delta)$$

Таким образом, предлагается ввести в информационное, математическое и методическое обеспечение АСУ ЧС вербальную и формальную постановки следующих задач.

Задача 1. (Задача определения границ стойкости заданных системных элементов).

Для заданной динамической модели управления определить границы стойкости целевых системных элементов $S^{(a)}$ при реализации $\gamma \in \Gamma^{(\alpha)}$ на горизонте управления Δ.

Задача 2. (Задача определения силы поражающего воздействия).

Пусть ЛПР получена информация, что в момент времени t_0 потенциально опасный системный элемент $S^{(\alpha)}$ может стать объектом несанкционированного воздействия.

Для заданного множества целевых системных элементов $MS^{(a)}$ определить последствие реализации угрозы в форме значений характеристик:

– момент $t^{(\beta)}$ выхода из строя системного элемента $S^{(\beta)} \in MS^{(a)}$;

– ущерб $W(S^{(\beta)})$ при выходе из строя элемента $S^{(\beta)} \in MS^{(a)}$;

– степень уязвимости системы;

– наиболее опасный по времени целевой элемент;

– наиболее уязвимый по ущербу целевой элемент;
– кумулятивный ущерб при выходе из строя подмножества $S^{(a\sigma)} \subseteq MS^{(a)}$ целевых элементов.

Задача 3. (Задача определения характера поражающего воздействия).

Для заданного множества целевых системных элементов $MS^{(a)}$ определить последствие реализации угрозы по форме и характеру повреждений в соответствии с классификацией:
– по характеру повреждений при наступлении ЧС:
— элемента системы $x_i \in X$,
— конструктивного решения (структуры системы) $R(X)$,
— концепта системы, т.е. заданных (проектируемых) свойств, в том числе ЦРФ $P(X)$,
— системных элементов, в том числе подсистем рассматриваемой системы $S^{(\beta)} \in MS^{(a)}$,
– внутреннего состояния системы, т.е. набора системных элементов и связей между ними,
— комплексные повреждения;
– по степени повреждений:
— сильные повреждения;
— средние повреждения;
— несущественные повреждения [5].

Задача 4. (Задача определения источника поражающего фактора).

Пусть при мониторинге ЛПР получена информация, что в момент времени t_0 значение параметра v_i потенциально опасного системного элемента $S^{(\alpha)}$ отклоняется от ЦРФ на величину $\delta^{(\alpha)}_i$.

Определить следующие характеристики ситуации:
– набор системных элементов $S^{(\sigma)}=\{S^{(\sigma)}_i\ i=1,\ldots,s^{(\sigma)}\}$, значения параметров $\mathbf{r}^{(\sigma)}$ которых отклоняются от ЦРФ;
– величины отклонения $\delta\mathbf{r}^{(\sigma)}_i$;
– системные элементы $S^{(\sigma)}=\{S^{(\sigma)}_i\ i=1,\ldots,s^{\sigma}\}$, которые являются источником возникновения возмущения;
– определить необходимые для анализа моменты времени мониторинга $\mathbf{r}^{(\sigma)}$ параметров системных элементов $S^{(\sigma)}=\{S^{(\sigma)}_i\ i=1,\ldots,s^{\sigma}\}$ (оптимальный мониторинг);
– возможные последствия распространения возмущений.

Значения характеристик, определенные при решении этих задач, а также условия безопасности позволяют определить базовые (стандартные), предполагаемые моменты $t^{(sf)}$ наступления экспертно-значимого события $\Im$ (ЭЗС), которые являются реперными точками развития обстановки [4]. Следовательно, будет создана база данных экспертно-значимых событий (БДЭЗС).

Решение указанных задач позволит создать базу данных для формирования базисных сценариев развития обстановки, варианты неопределенности которой зафиксированы в квазиинформационной гипотезе (КИГ) [4]. Для принятия эффективных решений целесообразно дать подробную структуризацию: описание возможных вариантов информированности, способов организации, распределения функций управления и т.п., т.е. четко сформулировать эти данные в качестве составляющих частей КИГ, которым будет следовать ЛПР при решении задачи анализа поведения и управления ЧС.

Новые задачи управления в режиме повышенной готовности.

Важнейшей задачей прогнозирования в режиме повышенной готовности является оперативное слежение за развитием обстановки, связанной с функционированием критически важных и опасных объектов. При этом учитываются сформированные в режиме повседневной деятельности обстоятельства и динамика в виде построения альтернативных сценариев его поведения. Основная цель построения сценариев в режиме повышенной готовности – дать ЛПР представление о качественно различных по своим характеристикам вариантах развития обстановок в условиях неопределенности.

Это позволяет, во-первых, максимально возможное ее снижение; во-вторых, описание не устраненной части неопределенности с помощью нескольких сценарных вариантов, что может существенно снизить влияние неопределенности на точность процессов прогнозирования, планирования и управления.

Рис. 6. Предлагаемые функции управления в режиме повышенной готовности.

Если в режиме повседневной деятельности сформированы базовые модели динамики и соответствующие им сценарии развития событий, то в режиме повышенной готовности, на основе оперативного мониторинга обстановки, ЛПР
– фиксирует наступление экспертно-значимых событий;
– выбирает варианты неопределенности, которые реализованы в текущих условиях при функционировании управляемых объектов;
– формирует квазиинформационные гипотезы дальнейшего развития обстановки;
– осуществляет ветвление синергического сценария соответственно текущим условиям;
– осуществляет прогнозирование изменения обстановки на основе решения задачи управления.

Таким образом, предлагается ввести в информационное, математическое и методическое обеспечение АСУ ЧС вербальную и формальную постановки следующих задач.

Задача 5. (Задача определения очередного ЭЗС).

Пусть ЛПР получена информация, что в момент времени t_0 опасный системный элемент $S^{(\alpha)}$ находится в ситуации $S(t_0)$. Определить момент наступления очередного ЭЗС, контрольные показатели которого зафиксированы в БДЭЗС.

Задача 6. (Задача оптимального мониторинга).

Пусть ЛПР получена информация, что в момент времени t_0 наблюдалось ЭЗС $\Im(t_0)$. Для заданного множества потенциально опасных системных элементов $MS^{(\alpha)}$ определить максимальное время $t^{(\max\alpha)}$ измерения их параметров, после которого будет потеряна их управляемость.

Задача 7. (Задача классификации состояний).

В момент времени t_0 дать классификацию состояний заданного множества потенциально опасных системных элементов $MS^{(\alpha)}$: безопасная, нештатная, кризисная, критическая, неконтролируемая, чрезвычайная.

Задача 8. (Задача определения возможности ухудшения состояний).

В соответствии классификацией состояний заданного множества потенциально опасных системных элементов $MS^{(\alpha)}$ в момент времени t_0 определить возможности ухудшения текущего состояния в менее желательное.

Задача 9. (Задача определения возможности управления состояниями).

В соответствии с классификацией состояний заданного множества потенциально опасных системных элементов $MS^{(\alpha)}$ в момент времени t_0 определить возможности управляемых переходов из текущего состояния в более желательное.

Новые задачи управления в чрезвычайном режиме.

Эффективность управления состоянием потенциально опасных системных элементов существенно зависит от следующих факторов: уровень неопределенности модели управления; уровень классической устойчивости объекта управления; уровень структурной устойчивости объекта управления; вероятностные характеристики реализуемости сценария; стоимостные характеристики реализуемости сценария; информационная полнота, характеризуемая оценкой оперирующей стороны степени информированности, заложенной в квазиинформационной гипотезе; организационная обеспеченность, характеризующая степень организационной подготовки при реализации сценария; ресурсная обеспеченность и реализуемость выбранного сценария.

Различные способы представления информации об обстановке и определение ответных действий в процессе принятия решений имеют важное значение для ЛПР с точки зрения оперативности и других критериев эффективности управления, при этом автоматизация указанного процесса требует четкой структуризации и формализации выбранных вариантов действий. Для построения формального аппарата разработки сценария требуется формализовать системные элементы проблемы, решаемой оперирующей стороной.

Важнейшей задачей управления в чрезвычайном режиме является обеспечение скорейшего выхода из ЧС. Для этого проводят ряд технических, технологических, организационных и экономических мероприятий, предусмотренных в нормативных документах и представленных на Рис. 3.

Для эффективного обеспечения управления в условиях уязвимости элементов системы следует решить вопросы об ее оптимальной реконфигурации, реорганизации и реструктуризации с целью максимальной

управляемой восстановимости элементов системы.

Существует много определений этих понятий. Так, под реконфигурацией сложного объекта понимаются процесс изменения его структуры (структур) в целях сохранения и последующего восстановления (повышения) уровня работоспособности объекта либо в целях обеспечения минимального снижения уровня эффективности системы при деградации ее функций [8]. Здесь будем использовать следующие соображения.

Реконфигурация – изменение отношений между системными элементами в технических и технологических системах.

Реорганизация – изменение организационной структуры организационной системы.

Реструктуризация – изменение отношений между системными элементами (изменение структуры) в любой сложной системе.

В качестве одного из подходов следует рассмотреть системные элементы (СЭ) системы оперативного управления многофункциональные унифицированные (однородные) организационные и ресурсно-функциональные средства.

Для реализации указанных мероприятий следует:

– построить модели управления для решения задач реконфигурации – установления новых отношений между функционирующими СЭ;

– модели реорганизации организационных структур: от линейных до дивизионных и штабных с тем, чтобы обеспечить необходимую оперативность принятия решений и решать вопросы их реализации;

– общую модель реструктуризации системы, включающую две предыдущие, а также учитывающие связи СЭ с внешним окружением.

Реконфигурация ресурсно-функционального комплекса в целях увеличения оперативности управления может проводиться по следующей методике. Множество решаемых задач разбивается на группы задач с близкими (одинаковыми) характеристиками. Каждая группа задач решается на наиболее продуктивном ресурсно-функциональном элементе (РФЭ) комплекса. При отказе РФЭ выполняемая им группа задач передается на РФЭ, где решаются задачи с наиболее низкими приоритетами. Если не удается решить задачи объединенных групп, то задачи группы с низкими приоритетами снимаются с автоматического решения и вводится режим ручного управления [9].

Для начального состояния осуществляется построение плана распределения задач и информационных потоков с учетом технологических, технических, стоимостных, временных, ресурсных и т.п. ограничений. В случае перехода системы (вызванного отказом некоторой совокупности РФЭ) в менее желательную ситуацию осуществляется перераспределение решаемых ею задач между работоспособными РФЭ. Обеспечение требуемого уровня функциональности осуществляется путем итогового размещения в различных РФЭ резервных копий алгоритмов решаемых задач.

В то же время анализ показывает, что при решении задачи реконфигурации системы в чрезвычайном режиме в рамках указанных процедур, как правило, требуется сформировать множество промежуточных состояний, переход в которые приводит к потере управляемости рассматриваемыми объектами.

При указанных предпосылках задачу реконфигурации для повышения качества моделирования в условиях действия случайных возмущений целесообразно использовать понятие марковского процесса, когда переход из состояния в состояние определяют вероятности перехода, значения которых не менее заданной величины.

Более эффективно задача реконфигурации, по-видимому, может быть решена на основе применения модели распространения возмущений, предложенной в работе [3].

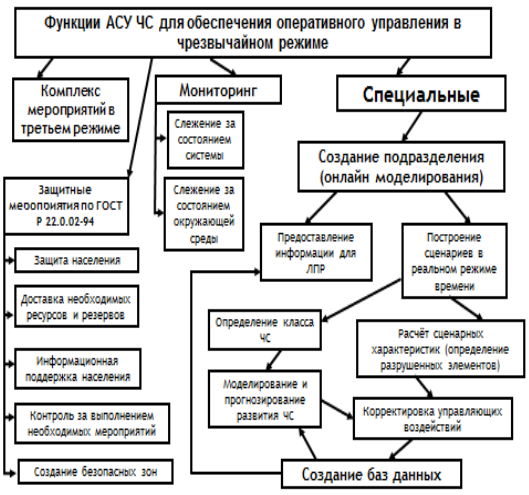


Рис. 7. Предлагаемые функции управления в чрезвычайном режиме.

Таким образом, предлагается ввести в информационное, математическое и методическое обеспечение АСУ ЧС вербальную и формальную постановки следующих задач.

Задача 10. (Задача определения ущерба поражающего воздействия).

Пусть ЛПР получена информация, что в момент времени t_0 потенциально опасный системный элемент $S^{(\alpha)}$ стал объектом несанкционированного воздействия.

Для заданного множества целевых системных элементов $S^{(a)}$ определить последствие реализации угрозы в форме значений характеристик:

– момент $t^{(\beta)}$ выхода из строя системного элемента $S^{(\beta)} \in S^{(a)}$;

– ущерб $W(S^{(\beta)})$ при выходе из строя элемента $S^{(\beta)} \in S^{(a)}$;

– степень уязвимости системы;

– наиболее опасный по времени целевой элемент;

– наиболее уязвимый по ущербу целевой элемент;

– кумулятивный ущерб при выходе из строя подмножества $S^{(a\sigma)} \subseteq S^{(a)}$ целевых элементов.

Задача 11. (Задача определения характера поражающего воздействия).

Для заданного множества целевых системных элементов $S^{(a)}$ определить последствие реализации угрозы по форме и характеру повреждений в соответствии с классификацией:

По характеру повреждений при наступлении ЧС:

– элемента системы $x_i \in X$;

– конструктивного решения (структуры системы) $R(X)$;

– концепта системы, т.е. заданных (проектируемых) свойств, в том числе ЦРФ $P(X)$;

– системных элементов, в том числе подсистем рассматриваемой системы $S^{(\beta)} \in S^{(a)}$;

– внутреннего состояния системы, т.е. набора системных элементов и связей между ними;

– комплексные повреждения.

По степени повреждений:

– сильные повреждения;

– средние повреждения;

– несущественные повреждения.

Задача 12. (Задача определения источника поражающего фактора).

Пусть при мониторинге ЛПР получена информация, что в момент времени t_0 значение параметра v_i потенциально опасного системного элемента $S^{(\alpha)}$ отклоняется от ЦРФ на величину $\delta^{(\alpha)}{}_i$.

Определить следующие характеристики возникшей ситуации:

– набор системных элементов

$$S^{(\sigma)}=\{S^{(\sigma)}{}_i \; i=1,\ldots,s^{(\sigma)}\},$$

значения параметров $\mathbf{r}^{(\sigma)}$ которых отклоняются от ЦРФ;

– величины отклонения $\delta\mathbf{r}^{(\sigma)}{}_i$;

– системные элементы

$$S^{(\sigma)}=\{S^{(\sigma)}{}_i \; i=1,\ldots,s^{\sigma}\},$$

которые являются источником возникновения возмущения;

– определить необходимые для анализа моменты времени мониторинга $\mathbf{r}^{(\sigma)}$ параметров системных элементов

$$S^{(\sigma)}=\{S^{(\sigma)}{}_i \; i=1,\ldots,s^{\sigma}\}$$

(оптимальный мониторинг);

– возможные последствия распространения возмущений.

Задача 13. (Задача классификации ЧС).

Пусть при мониторинге ЛПР получена информация, что в момент времени t_0 наступила ЧС.

Дать классификацию ЧС по следующим критериям:

– по длительности:

– мгновенные (соответствующие одному шагу сценария),

– средней продолжительности (в пределах горизонта сценария),

– длительные (превышающие горизонт сценария);

– по направленности ЧС:

– ЧС, увеличивающая силу воздействия на систему (величина ущерба увеличивается со временем),

– ЧС с ослабевающей силой воздействия на систему (величина ущерба уменьшается со временем),

– ЧС постоянной силы (величина ущерба во времени не изменяется или изменяются незначительно на выбранном промежутке сценария);

– по воздействию ЧС на элементы системы:

– влияющие напрямую (реализация ЧС в элементе является источником ЧС смежного элемента),

– влияющие косвенно (реализация ЧС в элементе является источником ЧС для элементов, находящихся в пределах достижимости для источника ЧС на заданном горизонте сценария),

– не влияющие (реализация ЧС в элементе при распространении возмущений по системе не

реализует ЧС на заданном горизонте сценария для экспертно-значимых элементов).

Новые задачи управления в режиме ликвидации последствий.

Задачами системы управления ЧС в этом режиме являются разработка и осуществление детальных планов мероприятий по предупреждению либо смягчению последствий ЧС на основе заранее подготовленных сценариев ее развития и ответных действий; мониторинг и корректировка основных стратегических задач и непрерывное слежение за появлением ЧС.

Изменение сложившейся обстановки при ЧС – изменение состояния потенциально опасного объекта (событие), в результате которого объект стал источником поражающего фактора распространяющегося по системе и нарушающее правильное функционирование элементов системы.

Рис. 8. Предлагаемые функции управления в режиме ликвидации последствий.

Таким образом, предлагается ввести в информационное, математическое и методическое обеспечение АСУ ЧС вербальную и формальную постановки следующих задач.

Задача 14. (Задача определения наличных ресурсов ликвидации поражающего фактора).

Защитные мероприятия осуществляются после предсказания момента наступления ЧС: развертывание систем наблюдения и разведки для уточнения прогноза, ввод в действие специальных правил функционирования экономики и общественной жизни, нейтрализация источников (объектов) повышенной опасности, готовность спасательных служб, частичная эвакуация населения.

Выбор оптимального сценария может проводиться методами теории статистических решений или теории игр.

Задача 15. (С учётом риска) Определить оптимальную стратегию осуществления управленческих мероприятий, которые наилучшим образом компенсируют действие поражающих факторов с заданным риском.

Задача 16. (Критерий max) Определить оптимальную стратегию управления, которая обеспечивает необходимую живучесть сложной системы в сложившейся ситуации, учитывая воздействия реализованных угроз.

Задача 17. (Критерий min) Определить оптимальную стратегию управления, которая обеспечивает допустимое угловое отклонение функционирования сложной системы от целевого направления при воздействии возможных угроз.

Решение поставленных задач позволяет ЛПР сделать анализ имеющихся стратегий по сценарным характеристикам и выбрать те из них, которые будут наиболее приемлемыми в сложившейся ситуации в зависимости от временных ограничений и имеющихся ресурсов.

Управляющие воздействия выбираются исходя из необходимых ресурсов, сил и средств, достаточных для противодействия силе угрозы и способствующих уменьшению масштаба предполагаемого ущерба. Оперативное управление может осуществляться на основе выбор эффективных управляющих воздействий с учётом баз данных сценарных характеристик, учитывающих динамику поведения сложной системы в сценариях в зависимости от возможного вида комплексных угроз. Оно также основано на расчёте характеристик по текущей информации в онлайн режиме.

Заключение

Можно предложить следующие направления исследований по данной теме:
– формализация предлагаемых методов;
– разработка алгоритмов для расчёта необходимых сценарных характеристик;
– разработка и классификация баз данных;
– определение области управляющих воздействий для перевода ситуации из более опасной в менее опасную;
– формализация методов выбора оптимальных стратегий оперативного управления при ЧС;
– определение оптимальных сценариев ликвидации последствий

New emergency control functions in complex organizational and technical systems

R.O. Ponomarev, D.A. Kononov

Abstract: The main emergency control regimes, as well as the factors which influence for decision-making depending on the current situation are considered. New control functions in indicated regimes which are implemented for system emergency are proposed. It based on situational analysis under scenario research of the studied object behavior. Control tasks providing the implementation of the proposed functions are formulated.

Keywords: situational modeling, emergency, control.

Литература

1. Федеральный закон от 21.12.1994 № 68-ФЗ (ред. От 23.06.2-16) «О защите населения и территорий от чрезвычайных ситуаций природного и техногенного характера».
2. И.М. Хасанов, Р.О. Пономарев. Управление в чрезвычайных ситуациях: существующие и перспективные подходы к исследованию /Труды 25-й Международной научной конференции «Проблемы управления безопасностью сложных систем» (Москва, 2017). – М.: Издательский Центр РГГУ, 2017. С. 521-526.
3. Д.А. Кононов, Н.Л. Лепе, Р.О. Пономарев. Управление чрезвычайными ситуациями в региональных системах методами ситуационного анализа //Вестник РГГУ. Серия «Управление». 2016. № 4 (6). С. 58-70.
4. Модели и методы анализа и синтеза сценариев развития социально-экономических систем. Кн. 1, 2 /Под редакцией чл.-корр. РАН Шульца В.Л., д.т.н., проф. Кульбы В.В. – М.: Наука, 2012.
5. Р.О. Пономарев, Д.А. Кононов, Н.О. Пономарев. Сценарное исследование сложных систем: уязвимость и чрезвычайные ситуации /Труды НИИСИ РАН. – М.: ФГУ ФНЦ НИИСИ РАН, 2016. Т. 6, № 2. С. 21-25.
6. Д.А. Кононов. Эффективные стратегии формирования сценариев в АСУ ЧС //АиТ. № 2. 2001. С. 170-181.
7. Н.И. Архипова, В.В. Кульба. Управление в чрезвычайных ситуациях. 3-е изд. Перераб. и доп. – М.: РГГУ. 2008.
8. Р.О. Пономарев, Д.А. Кононов, И.М. Хасанов. Сценарный анализ уязвимости сложной системы: вычислительный эксперимент /Труды НИИСИ РАН. – М.: ФГУ ФНЦ НИИСИ РАН, 2016. Т. 6, № 2. С. 26-33.
9. А.Н. Павлов. Методологические основы решения проблемы планирования структурно-функциональной реконфигурации сложных объектов. Изв. вузов. Приборостроение. 2012. Т. 55, № 11.

Определение коллизий аппроксимирующих цилиндров со сферами и плоскостью

Е.В. Страшнов[1], Д.В. Омельченко[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1]strashnov_evg@mail.ru, [2]omelchenko_dv@mail.ru

Аннотация: Определение коллизий (столкновений) трехмерных виртуальных объектов является важной задачей в системах виртуального окружения и видеотренажерах, в частности космических. В настоящей работе предлагаются алгоритмы определения коллизий аппроксимирующих цилиндров со сферами и плоскостью.

Ключевые слова: определение коллизий, аппроксимирующие контейнеры, точки контакта, системы виртуального окружения

Введение

Определение коллизий (столкновений) трехмерных виртуальных объектов является неотъемлемой частью моделирования динамики твердых тел в системах виртуального окружения и компьютерного моделирования [1]. Примером служит захват объекта антропоморфным роботом, где требуется определять коллизии между рабочим органом и захватываемым объектом. Определение коллизий для тел произвольной формы, заданных в виде полигональной сетки, является трудной вычислительной задачей. Поэтому одним из распространенных подходов [2] является применение аппроксимирующих контейнеров (сфера, цилиндр, параллелепипед, капсула и т.д.), окружающих геометрии объектов. В общем случае объект может окружаться несколькими такими контейнерами. Тогда определение коллизий двух объектов сводится к определению коллизий между парами аппроксимирующих контейнеров, окружающих эти объекты.

Алгоритмы определения коллизий виртуальных объектов можно условно разделить на две фазы: широкая и узкая. На широкой фазе определяется потенциальный список пересекающихся пар контейнеров. Примером служит алгоритм сортированных списков AABB [3] (axis-aligned bounding boxes). В узкой фазе [4] на основе полученного из широкой фазы списка потенциальных пар контейнеров определяется сам факт пересечения, и для пересекающейся пары контейнеров вычисляется необходимая контактная информация (точки контакта, глубины проникновения этих точек и нормаль расталкивания). Вычисленная контактная информация в дальнейшем используется на стадии разрешения коллизий [5, 6], в которой осуществляется моделирование удара, контакта и различных типов трения (скольжения, качения, верчения).

В настоящей работе мы рассматриваем апостериорные алгоритмы определения коллизий, то есть обработка коллизий происходит после установления факта их пересечения. В работах [7-9] были предложены алгоритмы определения коллизий для аппроксимирующих параллелепипедов, сфер и капсул. Здесь мы рассмотрим алгоритмы определения коллизий аппроксимирующего цилиндра со сферой и плоскостью, позволяющей в некоторых случаях удобным образом задать ровную поверхность земли.

1. Определение коллизии цилиндра и сферы

Рассмотрим цилиндр высотой l и радиусом r_1, локальная система координат которого расположена в центре цилиндра O_1, как показано на Рис. 1. Обозначим центры оснований цилиндра через P_1 и P_2, а сам цилиндр – через $C(O_1, r_1, l)$.

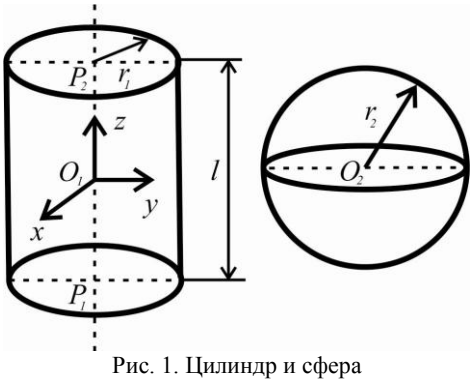


Рис. 1. Цилиндр и сфера

Пусть аппроксимирующая сфера имеет радиус r_2 и расположена в точке O_2. Обозначим сферу как $S(O_2, r_2)$.

Определение коллизии аппроксимирующего цилиндра со сферой состоит в установлении факта их пересечения и, в случае их пересечения в определении точки их контакта, глубины проникновения и нормали расталкивания. Точкой коллизии Q будем считать точку на поверхности цилиндра, ближайшую к центру сферы. Нормаль коллизии $\bar{n}$ – это единичный вектор, проходящий через точку Q и направленный к центру сферы. Глубина проникновения d – это расстояние, на которое проникла точка Q цилиндра внутрь сферы вдоль нормали $\bar{n}$. При этом мы предполагаем, что скорости объектов невелики, поэтому центр сферы не может сильно проникнуть внутрь цилиндра. Мы будем считать, что глубина проникновения $d < r_1 + r_2$.

Предлагаемый алгоритм определения коллизии цилиндра $C(O_1, r_1, l)$ и сферы $S(O_2, r_2)$ основан на определении положения центра сферы O_2 относительно цилиндра. Длина проекции вектора $\overline{P_1O_2}$ на ось цилиндра равна

$$s = \overline{P_1O_2} \cdot \bar{v},$$

где $\bar{v}$ – единичный вектор, направленный вдоль оси цилиндра.

Отклонение центра сферы O_2 от оси цилиндра равно

$$t = \left| \overline{O_2P_1} + s\bar{v} \right|.$$

Возможны три случая взаимного расположения сферы и цилиндра: точка цилиндра, ближайшая к центру сферы, лежит на боковой поверхности, на одном из его оснований и на одной из окружностей оснований. Рассмотрим эти случаи более подробно.

Случай 1. Если выполнены условия $0 \le s \le l$, $r_1 - t \le s$ и $r_1 - t \le l - s$, то ближайшая точка цилиндра находится на его боковой поверхности (см. Рис. 2). Действительно, первое условие означает, что центр сферы лежит между плоскостями оснований цилиндра. Последние два условия означают, что если центр сферы попал внутрь цилиндра, то расстояние $r_1 - t$ от центра сферы до боковой поверхности меньше, чем расстояния s и $l - s$ от центра сферы до оснований цилиндра. Если же центр сферы находится вне цилиндра, то эти два условия будут выполнены автоматически и точка, ближайшая к центру сферы, будет лежать на боковой поверхности цилиндра.

Если выполнено условие $t > r_1 + r_2$, то цилиндр и сфера не пересекаются. В противном случае определяем точку контакта, глубину проникновения и нормаль расталкивания.

На Рис. 2 показан случай, когда центр сферы лежит вне цилиндра. В этом случае верны равенства $AB + d = r_1$ и $AB + r_2 = t$. Тогда глубина проникновения вычисляется как

$$d = r_1 + r_2 - t.$$

Можно показать, что эта формула верна и для случая, когда центр сферы лежит внутри цилиндра.

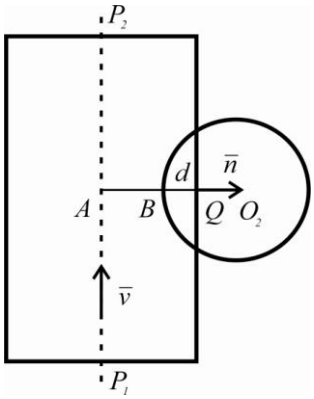


Рис. 2. Пересечение сферы с боковой поверхностью цилиндра

Нормалью коллизии будет вектор $\bar{n} = \dfrac{\overline{AO_2}}{\left|\overline{AO_2}\right|} = \dfrac{\overline{AP_1} + \overline{P_1O_2}}{t} = \dfrac{-s\bar{v} + \overline{P_1O_2}}{t}$, а точкой контакта $Q = O_2 - (t - r_1)\bar{n}$.

Случай 2. Если выполнено условие $t \le r_1$ и одно из условий $s < r_1 - t$ или $l - s < r_1 - t$, то ближайшая точка цилиндра находится на основании цилиндра (см. Рис. 3). Действительно, первое условие говорит о том, что центр сферы находится внутри бесконечно длинного цилиндра. Последние два условия означают, что если центр сферы попал внутрь цилиндра, то расстояния от центра сферы до одного из оснований цилиндра (s или $l - s$) меньше, чем расстояние $r_1 - t$ от центра сферы до боковой поверхности цилиндра.

Если выполнено одно из условий $s < -r_1$ или $s > l + r_1$, то цилиндр и сфера не пересекаются. В противном случае определяем нормаль, глубину и точку контакта.

Рассмотрим случай, когда выполнено условие $s < r_1 - t$, т.е. ближайшая точка к центру сферы лежит на нижнем основании цилиндра (Рис. 3, сфера I). Глубина проникновения тогда вычисляется как

$$d = QA = s + r_2,$$

где A – точка сферы, наиболее глубоко проникшая внутрь цилиндра.

Точка контакта и нормаль вычисляются по формулам:

$$Q = O_2 - s\bar{v}\ ,\ \ \bar{n} = -\bar{v}\ .$$

Другой случай соответствует условию $l - s < r_1 - t$, т.е. ближайшая точка лежит на верхнем основании цилиндра (Рис. 3, сфера II).

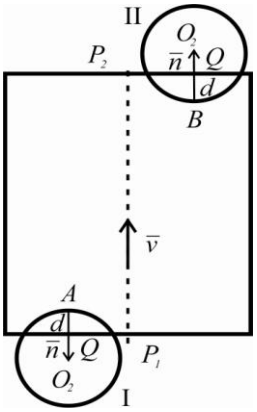


Рис. 3. Пересечение сферы с основанием цилиндра

Глубина проникновения в этом случае вычисляется как

$$d = QB = l - s + r_2,$$

где B – точка сферы, наиболее глубоко проникшая внутрь цилиндра.

Точка контакта и нормаль вычисляются по формулам

$$Q = O_2 + (l - s)\bar{v}\ ,\ \ \bar{n} = \bar{v}\ .$$

Случай 3. Если выполнено условие $t > r_1$ и одно из условий $s < 0$ или $s > l$, то точка цилиндра, ближайшая к центру сферы, находится на окружности нижнего (верхнего) основания цилиндра (см. Рис. 4). Действительно, первое условие означает, что центр сферы лежит снаружи от боковой поверхности цилиндра, а остальные условия – что он находится ниже (выше) основания цилиндра.

Если выполнено одно из условий $t > r_1 + r_2$ или $s < -r_1$ или $s > l + r_1$, то цилиндр и сфера не пересекаются. В противном случае для определения пересечения требуется дополнительная проверка.

Рассмотрим сначала случай, когда $-r_1 \le s < 0$ (см. Рис. 4, сфера I). Глубина проникновения d вычисляется по формуле:

$$d = r_2 - O_2Q = r_2 - \sqrt{s^2 + (t - r_1)^2}\ .$$

Если $d < 0$, то цилиндр и сфера не пересекаются, в противном случае они пересекаются и мы вычисляем точку контакта

$$Q = P_1 - \overline{QP_1} = P_1 - r_1\frac{\overline{O_2A}}{t} = P_1 - r_1\frac{\overline{O_2P_1} + s\bar{v}}{t}\ ,$$

где $\overline{O_2A} = \overline{O_2P_1} + \overline{P_1A} = \overline{O_2P_1} + s\bar{v}$.

Нормаль вычисляется как $\bar{n} = \dfrac{\overline{QO_2}}{r_2 - d}$.

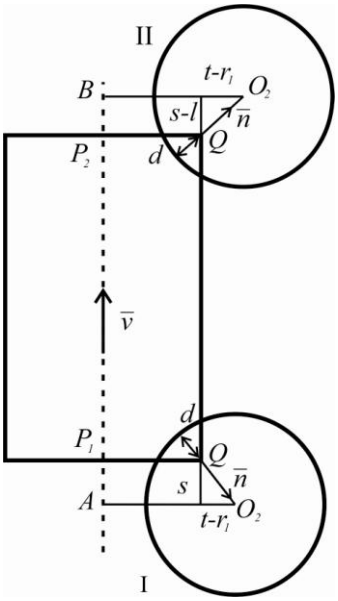


Рис. 4. Пересечение сферы с окружностью основания цилиндра

Аналогично, если $l < s \le l + r_1$ (Рис. 4, сфера II), то глубина проникновения вычисляется по формуле:

$$d = r_2 - O_2Q = r_2 - \sqrt{(s - l)^2 + (t - r_1)^2}\ .$$

При $d < 0$ цилиндр и сфера не пересекаются, в противном случае вычисляем точку контакта

$$Q = P_2 - \overline{QP_2} = P_2 - r_1\frac{\overline{O_2B}}{t} = P_2 - r_1\frac{\overline{O_2P_1} + s\bar{v}}{t}$$ и

нормаль коллизии $\bar{n} = \dfrac{\overline{QO_2}}{r_2 - d}$.

2. Определение коллизии цилиндра с плоскостью

Рассмотрим цилиндр $C(O, r, l)$ и плоскость, заданную в векторной форме

$$\bar{n} \cdot \bar{p} - d = 0\ ,$$

где $\bar{n} = (n_x, n_y, n_z)^{\mathrm{T}}$ – единичный вектор, перпендикулярный к плоскости и направленный от начала мировой системы координат к плоскости, $\bar{p} = (x, y, z)^{\mathrm{T}}$ – координаты любой точки плоскости, $d \ge 0$ – расстояние от начала мировой системы координат до плоскости.

Пусть точки P_1 и P_2 являются центрами оснований цилиндра, а единичный вектор $\bar{v}$ задает направление оси цилиндра.

Плоскость разбивает пространство на две области. Мы будем считать, что цилиндр из-

начально расположен в той области, где для каждой точки $\bar{p}$ выполнено неравенство $d-\bar{n}\cdot\bar{p}<0$. Тогда цилиндр и плоскость будут пересекаться, если найдется такая точка $\bar{q}$ цилиндра, для которой будет выполнено неравенство $d-\bar{n}\cdot\bar{q}\geq 0$. В предположении, что скорости объектов малы, будем полагать, что цилиндр в процессе моделирования не пересечет полностью плоскость.

Определение коллизии аппроксимирующего цилиндра с плоскостью базируется на взаимном расположении оси цилиндра и нормали к плоскости. Возможны три случая: когда ось цилиндра перпендикулярна, параллельна и расположена произвольно относительно плоскости. Нормаль коллизии цилиндра с плоскостью определим, как единичный вектор, совпадающий с нормалью к плоскости.

Ось цилиндра перпендикулярна плоскости. Этот случай соответствует условию $\bar{n}\cdot\bar{v}=\pm 1$, то есть вектор нормали к плоскости и вектор оси цилиндра коллинеарны. В этом случае возможно пересечение основания цилиндра с плоскостью.

Если для точек цилиндра выполнено неравенство

Вычислим расстояния (со знаком) от точек P_1 и P_2 до плоскости по формулам:

$$d_1=d-\bar{n}\cdot P_1,\quad d_2=d-\bar{n}\cdot P_2.$$

Если цилиндр пересекается с плоскостью, то будет выполнено неравенство $d_1\cdot d_2\leq 0$, в противном случае цилиндр и плоскость не пересекаются.

Рассмотрим случай коллизии нижнего основания цилиндра с плоскостью. В этом случае для точки P_1 выполнено неравенство $d_1\geq 0$. Построим два взаимно перпендикулярных вектора $\bar{v}_1$ и $\bar{v}_2$ в основании цилиндра (см. Рис. 5).

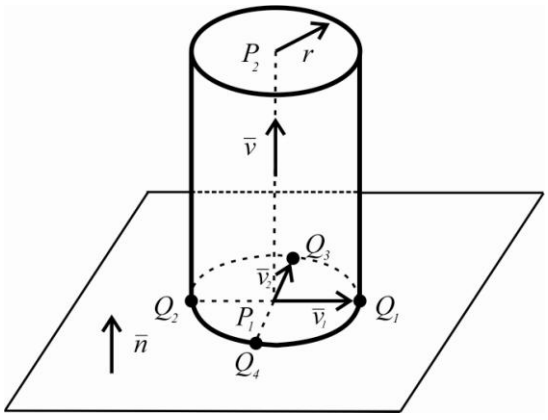


Рис. 5. Пересечение основания цилиндра с плоскостью

Алгоритм построения векторов $\bar{v}_1$ и $\bar{v}_2$ следующий. Если $v_x=0$, то выбираем вектор $\vec{v}_2'=0,-v_z,v_y$, в противном случае $\vec{v}_2'=v_z,0,-v_x$. В итоге получаем, что для вектора $\vec{v}_2'$ выполнены условия $\vec{v}_2'\cdot\bar{v}=0$ и $\|\vec{v}_2'\|\neq 0$. Нормализуем вектор $\vec{v}_2'$: $\bar{v}_2=\vec{v}_2'/\|\vec{v}_2'\|$. Вектор $\bar{v}_1$ определяется через векторное произведение векторов $\bar{v}$ и $\bar{v}_2$, то есть $\bar{v}_1=\bar{v}_2\times\bar{v}$. Полученные вектора масштабируются до границ основания цилиндра: $\bar{v}_1=r\bar{v}_1$, $\bar{v}_2=r\bar{v}_2$.

В рассматриваемом случае для стабильности моделирования достаточно 4-х точек коллизии, которые определяются по следующим формулам:

$$Q_1=P_1+\bar{v}_1,\ \ Q_2=P_1-\bar{v}_1,$$
$$Q_3=P_1+\bar{v}_2,\ \ Q_4=P_1-\bar{v}_2.$$

Их глубины проникновения вычисляются через уравнение плоскости как

$$d_i=d-\bar{n}\cdot Q_i,\ i=\overline{1,4}.$$

Пересечение верхнего основания цилиндра с плоскостью определяется аналогичным образом.

Ось цилиндра параллельна плоскости. Это соответствует случаю, когда ось цилиндра перпендикулярна нормали к плоскости, т.е. выполнено условие $\bar{n}\cdot\bar{v}=0$. Тогда потенциально возможны две точки контакта, как показано на Рис. 6.

Искомые точки коллизии вычисляются по формулам

$$Q_i=P_i-r\bar{n},\ i=1,2.$$

Их глубины проникновения вычисляются из уравнения плоскости следующим образом:

$$d_i=d-\bar{n}\cdot Q_i,\ i=1,2.$$

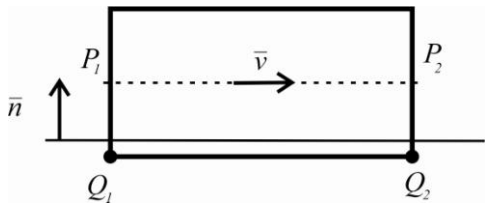


Рис. 6. Пересечение боковой поверхности цилиндра с плоскостью

Если окажется, что $d_i<0$, то цилиндр и плоскость не пересекаются.

Произвольное положение оси цилиндра относительно плоскости. Это соответствует случаю, когда ось цилиндра не параллельна и не перпендикулярна плоскости. Определение коллизии цилиндра с плоскостью в этом случае основывается на поиске точки Q, для которой будет верно

$$d_m = d - \bar{n}Q = \max_{\bar{q}\in C(O,r,l)} (d - \bar{n}\cdot\bar{q}).$$

Тогда, если будет выполнено условие $d_m < 0$, то цилиндр и плоскость не пересекаются. В противном случае определяем все точки коллизии и их глубины.

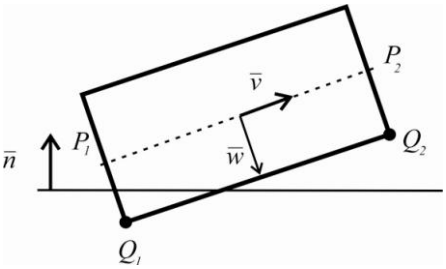


Рис. 7. Пересечение окружности основания цилиндра с плоскостью

В качестве возможных кандидата точки Q являются точки Q_1 и Q_2, как показано на Рис. 7. Положения точек Q_i определяется с помощью вектора $\bar{w}$, перпендикулярного оси цилиндра. Этот вектор вычисляется по формуле

$$\bar{w} = r\frac{(\bar{v}\cdot\bar{n})\bar{v} - \bar{n}}{\left\|(\bar{v}\cdot\bar{n})\bar{v} - \bar{n}\right\|}.$$

Полученный вектор построен таким образом, чтобы были выполнены условия $\bar{w}\cdot\bar{v} = 0$, $\bar{w}\cdot\bar{n} < 0$ и $\|\bar{w}\| = r$.

Точки Q_1 и Q_2 вычисляются как

$$Q_i = P_i + \bar{w},\ i = 1,2.$$

Рассмотрим случай, когда искомая точка Q лежит на нижнем основании, т.е. $Q = Q_1$. Тогда если $d_1 = d - \bar{n}\cdot Q_1 \geq 0$, то цилиндр и сфера пересекаются. В этом случае возможны одна, две и четыре точки коллизии.

Если для точки Q_2 верхнего основания цилиндра выполнено $d_2 = d - \bar{n}\cdot Q_2 \geq 0$, то точка Q_2 является точкой коллизии. Это соответствует случаю пересечения боковой поверхности цилиндра с плоскостью. В итоге получаем две точки коллизии Q_1 и Q_2 с глубинами проникновения d_1 и d_2.

В противном случае (т.е. если для точки Q_2 выполнено $d_2 < 0$), рассмотрим точку $Q_3 = P_1 - \bar{w}$.

Если выполнено неравенство $d_3 = d - \bar{n}\cdot Q_3 \geq 0$, то это соответствует случаю пересечения основания цилиндра с плоскостью. В этом случае необходимо взять четыре точки коллизии, лежащие в нижнем основании цилиндра. Это точки Q_1 и Q_3, рассмотренные ранее, и точки Q_4 и Q_5, которые вычисляются как

$$Q_4 = P_1 + \bar{t},\ Q_5 = P_1 - \bar{t},$$

где $\bar{t} = \dfrac{\bar{v}\times\bar{w}}{\left\|\bar{v}\times\bar{w}\right\|}$. Для этих точек определяем глубины проникновения d_4 и d_5.

Если $d_3 < 0$, то точка Q_1 является единственной точкой коллизии с глубиной проникновения d_1.

Точки коллизии для случая, когда $Q = Q_2$, т.е. искомая точка лежит на верхнем основании, вычисляются аналогичным образом.

Заключение

Предложенные в статье алгоритмы определения коллизий аппроксимирующего цилиндра со сферой и плоскостью не требуют существенных вычислительных затрат, что позволяет моделировать динамику большого (несколько тысяч) количества виртуальных объектов в режиме реального времени.

Эти алгоритмы реализованы в подсистеме динамики имитационного комплекса, разработанного в ФГУ ФНЦ НИИСИ РАН и предназначенного для моделирования динамики виртуальных роботов, в том числе динамики антропоморфных роботов.

Апробация алгоритмов в рамках этой системы показала их применимость для моделирования динамики виртуальных объектов.

Работа выполнена при поддержке РФФИ, грант № 17-07-00137.

Collision detection of bounding cylinders with spheres and a plane

E.V. Strashnov, D.V. Omelchenko

Abstract: Collision detection of 3D virtual objects is an important task in virtual environment systems, in particular space simulators. In this paper, we propose collision detection's algorithms for bounding cylinders with spheres and a plane.

Keywords: collision detection, bounding volumes, contact points, virtual environment systems.

Литература

1. C. Ericson. Real-Time Collision Detection. CRC Press, 2004, 594 p.
2. M. Moore, J. Wilhelms. Collision Detection and Response for Computer Animation. Computer Graphics, 1988, Vol. 22, no. 4, pp. 289–298.
3. J. Cohen, M. Lin, D. Manocha, and M. Ponamgi. I-collide: An interactive and exact collision detection system for large-scale environments. In Proc. of ACM Interactive 3D Graphics Conference, pp. 189-196, 1995.
4. C. Fares, and Y. Hammam. Collision detection for rigid bodies: A state of the art review. In GraphiCon, 2005.
5. L. Kavan. Rigid Body Collision Response, Faculty of Mathematics and Physics, Charles University of Prague, 2003.
6. А.М. Трушин. Обработка коллизий виртуальных объектов с помощью метода последовательных импульсов // Труды НИИСИ РАН, 2014, том 4, № 2, стр. 95-105.
7. А.М. Трушин. Определение коллизий аппроксимирующих сфер и прямоугольных параллелепипедов в системах трехмерного моделирования // Программные продукты и системы, 2015, № 4, стр. 105-109.
8. А.М. Трушин, М.В. Михайлюк. Определение коллизий аппроксимирующих капсул и сфер трехмерных виртуальных динамических объектов // Труды НИИСИ РАН, 2016, т.6, № 2, стр. 42-45.
9. М.В. Михайлюк, П.Ю. Тимохин. Определение коллизий аппроксимирующих капсул и прямоугольных параллелепипедов // Труды НИИСИ РАН, 2017, т. 7, № 1, стр. 25-29.

Модель автономного агента, строящего и использующего когнитивную карту

Т. И. Шарипова[1], В. Г. Редько[2], Г. А. Бесхлебнова[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]SharipovaTamara@gmail.com, [2]vgredko@gmail.com, [3]beskhlebnova@gmail.com

Аннотация. В статье построена модель автономного агента, который формирует когнитивную карту двумерного пространства и использует эту карту при поисковом поведении. Используется метод нейронного газа, в котором нейронная сеть формируется при обучении агента, память каждого нейрона запоминает определенную точку в пространстве. Исследуется поисковое поведение автономного агента в двумерном лабиринте. Показана возможность резкого сокращения размеров нейронной сети, в которой достаточно запоминать только те точки пространства, в которых сильно меняется окружающая ситуация.

Ключевые слова: когнитивная карта, автономный агент, метод нейронного газа, поисковое поведение.

Введение

Ранее нами была построена модель автономного агента, ведущего поиск в одномерном лабиринте [1]. Использовался метод растущего во времени нейронного газа [2]. Построенная модель продемонстрировала самостоятельное нахождение автономным агентом источника пищи. В настоящей работе моделируется аналогичное поисковое поведение агента в двумерном лабиринте. Также используется метод нейронного газа.

Суть этого метода состоит в следующем.

Имеется сеть связанных между собой нейронов, узлов сети. Каждый нейрон-узел запоминает определенную сенсорную информацию $\mathbf{S}_i$ (вектор памяти нейрона), где i – номер нейрона. Нейроны имеют веса, модифицируемые путем обучения. Нейронная сеть управляет поведением агента. Текущая сенсорная информация на входе системы управления агента $\mathbf{S}(t)$ сравнивается со всеми векторами памяти нейронов $\mathbf{S}_i$, и ищется k-й нейрон, для которого расстояние между $\mathbf{S}(t)$ и $\mathbf{S}_k$ минимально (t – дискретное время). Если для этого нейрона расстояние между $\mathbf{S}(t)$ и $\mathbf{S}_k$ меньше определенного порога, то вектор $\mathbf{S}_k$ немного приближается к вектору $\mathbf{S}(t)$ и k-й нейрон становится активным. Если это расстояние больше порога, то создается новый нейрон-узел, который в своей памяти запоминает текущую сенсорную информацию $\mathbf{S}(t)$.

Связи между нейронами сети формируются при переходе между нейронами. Например, когда создается новый нейрон, то формируется связь между нейроном, который был активен в предыдущий такт времени, и новым нейроном.

Если сенсорная информация определяет положение агента в пространстве, то несложно организуется перемещение агента в соответствии с запомненной сенсорной информацией. В этом случае предварительно происходит блуждание агента по пространству. При этом веса нейронов формируются методом обучения с подкреплением [3]. Чем более благоприятно место организма в пространстве, тем больше вес нейрона, вектор $\mathbf{S}_i$ которого определяет данную точку в пространстве. После предварительного блуждания по пространству формируется нейронная сеть со связями между нейронами. Движение агента может управляться такой нейронной сетью следующим образом. Каждый такт времени агент находится в точке пространства, соответствующей вектору $\mathbf{S}_i$ активного нейрона. Далее анализируются все связи этого нейрона, и выбирается нейрон, который связан с данным активным нейроном и имеет наибольший вес среди таких нейронов. Затем происходит переход от данного нейрона к выбранному нейрону с большим весом. При этом агент перемещается в пространстве в точку с координатами, соответствующими новому выбранному нейрону. Новый нейрон становится активным.

В настоящей модели исследуется поисковое поведение агента в двумерном лабиринте. Особое внимание уделяется

процессам формирования растущего нейронного газа в процессе поиска. Строя нейронную сеть, агент формирует когнитивную карту рассматриваемого лабиринта.

1. Описание модели

Описание автономного агента. Автономный агент представляет собой материальную точку в двумерном пространстве. Агент может находиться только в пустых областях лабиринта. Агент имеет сенсоры, с помощью которых может измерять расстояние до стенок, находящихся справа, слева, спереди и сзади. Хотя автономный агент может перемещаться в любом направлении, он не теряет своих представлений о нахождении «сторон света».

Агента имеет полезный ресурс R. Цель агента – пополнение ресурса. Агент строит когнитивную карту лабиринта и осуществляет поиск источников пищи, питаясь, агент пополняет свой ресурс.

Двумерный случай. Модель имеет следующие предположения.

1. Рассматривается агент, который может двигаться в двумерном пространстве x, y.

2. Имеется лабиринт с источником питания. Используемый при моделировании лабиринт показан на Рис. 1.

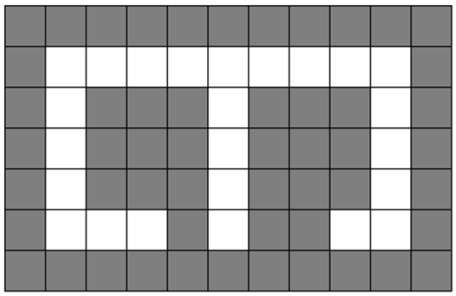


Рис. 1. Схема коридора. Белые квадраты – пустые клетки лабиринта, в которых может находиться агент, серые квадраты -стенки лабиринта

Лабиринт представляет собой прямоугольную область, состоящую из квадратов размером 10x10 единиц. Каждый квадрат может быть либо пустым (где может находиться агент), либо полным (представляет из себя стенку, где агент находиться не может). То есть, лабиринт состоит из нескольких прямоугольных участков, «комнат». Более того, рассматривается частный случай лабиринтов, где внешняя граница окружена стеной (заполненными квадратами), а пустые области представляют собой коридоры шириной не более 1 квадрата. Агент движется внутри лабиринта. Задача агента – исследование лабиринта и поиск источника пищи.

3. Источник пищи имеет зону действия – подобласть двумерного лабиринта.

4. Агент имеет ресурс R, который увеличивается при нахождении источника пищи.

5. Агент функционирует в дискретном времени t. Каждый такт времени агент совершает движение, при этом его координаты x, y изменяются на некоторые величины $\Delta x(t)$, $\Delta y(t)$.

6. Когда координата агента совпадает с зоной действия источника пищи, ресурс агента за один такт времени увеличивается на величину Δr.

7. Агент имеет свою систему управления, на сенсорный вход которой поступают координаты агента $x(t)$, $y(t)$.

8. Система управления агента задается растущей нейронной сетью. На вход активного нейрона подаются текущие координаты агента $x(t)$, $y(t)$.

9. Каждый нейрон имеет память, он запоминает определенные координаты x_i, y_i в данном случае вектор памяти $\mathbf{S}_i$ имеет шесть компонент: четыре расстояния до стенок лабиринта, находящихся спереди, сзади, слева и справа от агента, а также координаты агента x_i, y_i. Кроме того, каждый нейрон имеет свой вес W_i . Чем больше вес нейрона, тем более предпочтительно то место, которое характеризует нейрон.

10. Имеется два режима динамики агента: 1) режим случайного движения и 2) режим детерминированного перемещения, перемещения в соответствии весами узлов-нейронов нейронного газа. Характерное время перехода из случайного режима в детерминированный равно T_N.

11. Каждый такт времени выбирается первый или второй режим. Причем вероятность выбора первого режима, т.е. режима случайного поиска в начале функционирования агента близка к 1, а дальнейшем эта вероятность постепенно уменьшается и происходит переход к детерминированному движению в соответствии с весами нейронов. Таким образом, реализуется метод отжига [4]: при малых временах t агент движется случайно (формирует нейронную сеть), при больших временах – детерминировано, в соответствии с весами нейронов.

12. В режиме случайного поиска после перемещения агента его координаты

становятся равными $x(t)$, $y(t)$. Определяются параметры комнаты (длина, ширина), в которой находится агент. Если параметры комнаты изменились, то формируется новый нейрон, в памяти которого записываются расстояния от агента до стенок лабиринта и координаты агента. Новый нейрон становится активным.

13. При появлении нового нейрона формируется связь от предыдущего активного нейрона к новому.

14. В режиме детерминированного перемещения определяются веса всех «контактных» нейронов, которые связаны с текущим активным, и среди этих контактных нейронов находится предпочтительный, имеющий наибольший вес. Этот нейрон становится активным в следующий такт времени. Координата агента становится равной координате, хранящейся в памяти предпочтительного нейрона.

Описание когнитивной карты. В данной работе под когнитивной картой имеется в виду нейронная сеть, отображающая исходный лабиринт. Каждый нейрон запоминает свои двумерные координаты, соответствующие точке пространства. Связь между нейронами подразумевает наличие коридора между этими точками. Когнитивная карта создается автономным агентом по мере изучения лабиринта. Веса нейронов меняются только после нахождения полезного ресурса. Если не найдена область полезного ресурса, то веса всех нейронов равны 0. Если область полезного ресурса найдена, то нейроны, находящиеся в данной области, имеют вес, равный единице. Веса остальных нейронов рассчитываются по формуле

$$W_i = 0{,}5 \max (W_j),$$

где W_j – веса соседних (контактных) нейронов. Под соседями нейрона подразумевается нейроны, находящиеся через одно ребро от рассматриваемого.

Нейроны появляются только в трёх случаях: 1) там, где появился автономный агент (точка старта); 2) в «тупиковых» случаях: подробнее см. ниже; 3) там, где глобально менялись параметры комнаты (у автономного агента есть четыре сенсора, по которым он измеряет длину и ширину комнаты. Если длина или ширина изменились более чем на заданное количество (в компьютерной программе это количество равно 5) единиц, то считается, что автономный агент попал в перекресток или новую комнату.

При построении нейронной сети (когнитивной карты) лабиринта нейроны сети создаются только в особых точках лабиринта. Первая такая точка – это место появления автономного агента. Кроме того, нейроны создаются на перекрестках (по одной точке на один перекресток) и в коридорах (одна точка на один коридор), а также в тупиках (концах коридора). Предложенный принцип построения когнитивной карты позволит существенно уменьшить размер нейронной сети.

Целью автономного агента является составление когнитивной карты существующего лабиринта и нахождение областей полезного ресурса.

Описание схемы передвижения автономного агента. При исследовательском поведении агента сначала рассматриваются возможные варианты выбора общего направления автономного агента. Рассматриваются четыре стороны (вверх, вниз, слева, справа) на возможность прохождения на 10 и более шагов. Если такая возможность есть, то данное направление считается возможным для дальнейшего исследования. Далее в выбранном направлении с отклонением в ±90° автономный агент пробует произвести шаг. Если произвести шаг нельзя (агент попадает в полный квадрат лабиринта), то выбирается другое отклонение и делается попытка шага в новом направлении.

За каждую единицу времени автономный агент может совершить одно движение или перемещение. Движение представляет собой переход автономного агента на 1 единицу в выбранном направлении. Перемещение представляет собой переход адаптивного агента на 10 единиц в выбранный коридор или перемещение между созданными и запомненными точками-нейронами. Если при выполнении предполагаемого действия агент выходит за пределы лабиринта (натыкается на стенку), то он это действие не совершает.

Описание поискового поведения автономного агента. В первоначальный момент времени автономный агент появляется в пустой области лабиринта. С помощью сенсоров он определяет параметры (длину – a и ширину – b) коридора, в котором находится, а также определяет направления, которые он может исследовать. Случайным образом выбирается одно из возможных направлений для исследований. Сначала делается перемещение на 10 единиц строго в выбранном направлении (чтобы, если

автономный агент находится на перекрёстке, он случайно не ушёл в другой коридор). После этого в выбранном направлении со случайным отклонением автономный агент делает движение. Если в выбранном направлении движение невозможно, то выбирается другое случайное отклонение. Так происходит до тех пор, пока параметры комнаты глобально не изменятся (т.е. длина или ширина, или оба параметра). Тогда в данной точке создаётся новый нейрон. Для него считается исследованным направление, из которого он пришел, и проверяется возможность исследования в других направлениях. При этом в когнитивной карте создается связь между созданным нейроном и тем, из которого пришел автономный агент. Если I_T раз подряд не получается сделать движение в выбранном направлении, то автономный агент считает, что попал в тупик. Тогда в данной точке создаётся нейрон, в котором три направления невозможны для исследования, а оставшееся направления помечается как исследованное. После этого автономный агент возвращается в нейрон, из которого начал движение в данном направлении, помечает коридор, из которого вернулся, как исследованный (создает связь между нейронами) и выбирает другое направление для исследований. Если в данном месте все направления исследованы, то автономный агент возвращается в тот нейрон, из которого пришел к текущему нейрону, пока не найдет нейрон с неисследованными направлениями. Если весь лабиринт исследован, то дальнейшее поведение автономного агента определяется тем, какой мотив доминирует в автономном агенте.

2. Результаты моделирования

Параметры расчетов. Начальное значение ресурса агента R = 10. На каждое движение затрачивается часть ресурса Δr = 0,01. Если автономный агент находится в области полезного ресурса, то его запасы увеличиваются на Δr = 1.

Характерное время перехода с режима случайного поиска в детерминированный режим равно T_N = 1000 тактов времени.

Количество итераций, после повторения которых автономный агент решает, что попал в тупик I_T = 50.

Результаты моделирования динамики агента для двумерного случая представлены на Рис. 2-5. Видно, что, при выбранных параметрах большую часть времени агент находится в области полезного ресурса.Однако, при небольших значениях времен происходит уменьшение полезного ресурса, что соответствует растрате ресурса при совершении движений (Рис. 3).

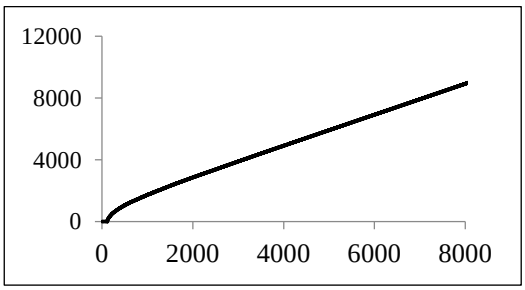


Рис. 2. Зависимость уровня ресурса агента R от времени t

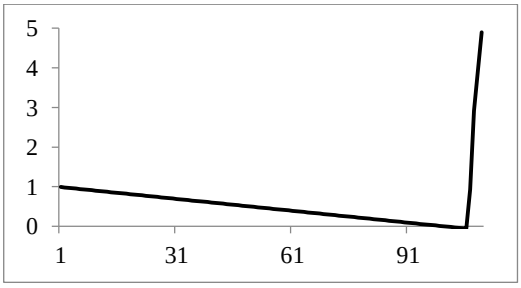


Рис. 3. Зависимость уровня ресурса агента R от времени t (начальный участок графика)

Зависимость числа нейронов от времени представлена на Рис. 4. Показано, что, при достаточно большом времени размер нейронной сети не изменяется. Это связано с принципами построения нейронной сети (когнитивной карты) лабиринта – нейроны создаются лишь в некоторых подпространствах лабиринта, характеризующих текущую ситуацию нахождения агента в лабиринте. На Рис. 5 показана траектория движения агента по двумерному лабиринту. Кружками отмечены создаваемые узлы нейронной сети. Узлы образуются при сильном изменении окружающей среды. Зона полезного ресурса (источник пищи) отмечен на рисунке серым фоном.

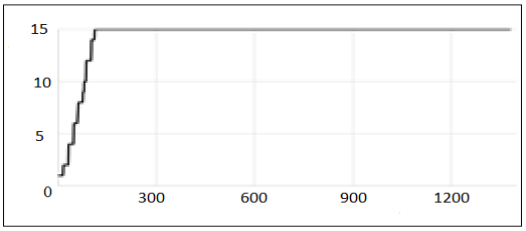


Рис. 4. Зависимость числа нейронов N от времени t

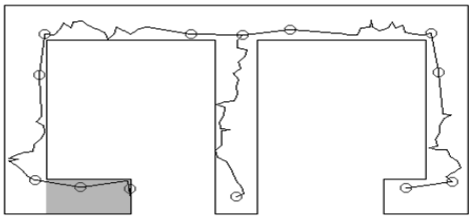


Рис. 5. Движение агента по двумерному лабиринту. Траектория движения агента показана изломанной линией, точки, характеризующие сильное изменение окружающей обстановки, отмечены кружками, источник пищи показан серым фоном

Анализ полученных результатов показал, что агент успешно анализирует лабиринт, строит когнитивную карту и находит источник пищи, после этого ресурс агента растет.

Кроме того, показана возможность резкого сокращения размеров нейронной сети, в которой достаточно запоминать только те точки пространства, в которых сильно меняется окружающая ситуация (в данном случае это соответствует сильному изменению размеров комнаты).

Заключение

Таким образом, построена модель поискового поведения агента, система управления которого формируется на основе метода растущего нейронного газа. Разработан метод обучения с подкреплением для растущей нейронной сети; проанализирован этот метод для двумерного лабиринта. Построен вариант модели растущего нейронного газа, в котором радикально сокращается число узлов-нейронов за счет того, что в нейронах запоминаются не все точки, в которых побывал агент, а только те, в которых радикально меняется окружающая среда. Нейронная сеть характеризует построенную агентом когнитивную карту пространства, в котором агент ведет поиск. Агент использует эту когнитивную карту.

Настоящая работа выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «*35.16. Моделирование процессов формирования когнитивных способностей автономных агентов*» (№ 0065-2018-0002).

The model of an autonomous agent that builds and uses a cognitive map

T.I. Sharipova, V.G. Red'ko, G.A. Beskhlebnova

Abstract. The article describes a model of an autonomous agent that forms a cognitive map of two-dimensional space and uses this map in searching behavior. The method of neuron gas is used, in which the neural network is formed by the agent, the memory of each neuron remembers a certain point in space. The searching behavior of an autonomous agent in a two-dimensional labyrinth has been investigated. The possibility of a sharp reduction in the size of the neural network has been demonstrated. It has been shown that it is enough to remember only those space points that correspond to significant changes of surrounding situations.

Keywords: cognitive map, autonomous agent, neuron gas, searching behavior.

Литература

1. В.Г.Редько, Т.И.Шарипова. Использование метода нейронного газа для моделирования поискового поведения агентов. «Труды НИИСИ РАН», т. 3 (2013), № 2, 22–25.
2. M.V.Butz, E.Shirinov, K.Reif. Self-organizing sensorimotor maps plus internal motivations yield animal-like behavior. "Adaptive Behavior", v. 18 (2010), no. 3–4, 315–337.
3. Р.С.Саттон, Э.Г.Барто. Обучение с подкреплением. М., Бином, 2011.
4. S.Kirkpatrick, C.D.Gelatt, M.P.Vecchi. Optimization by simulated annealing "Science", v. 220 (1983), no. 4598, 671–680.

Исследование процессов самоорганизации в эволюционной модели прозрачной децентрализованной экономики

З. Б. Сохова[1], В. Г. Редько[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail': [1]zarema_s@mail.ru, [2]vgredko@gmail.com

Аннотация. В статье предлагается модель взаимодействия инвесторов и производителей в децентрализованной прозрачной экономической системе. Анализируются процессы *эволюции и самоорганизации* в данном сообществе. Предложен конкретный механизм распределения капитала инвесторов между производителями. Важной особенностью модели является прозрачность и децентрализация. Инвесторы и производители открыто обмениваются информацией о своих капиталах, эффективностях и намерениях с помощью легких агентов-посланников. Модель исследована с помощью компьютерного моделирования, результаты которого демонстрируют эффективность рассматриваемого механизма.

Ключевые слова: децентрализация, многоагентные системы, коллективное поведение, сотрудничество, эволюция, прозрачная экономика.

Введение

В последние годы идеи сотрудничества [1], прозрачности и децентрализации в экономике стали играть значительную роль. В данной работе ставится вопрос − выгодно ли сотрудничество конкурирующим экономическим субъектам? Предлагается конкретная модель взаимодействия инвесторов и производителей в прозрачной децентрализованной экономике. Работа является продолжением работы [2]. В данной работе проводится детальное исследование *эволюционной модели* взаимодействия инвесторов и производителей.

Основные идеи, положенные в основу модели, следующие. Имеются две эволюционирующие популяции агентов, которые взаимодействуют в конкурентной среде: популяция агентов-инвесторов и популяция агентов-производителей. Инвесторы передают свой капитал производителям и получают определенную долю их прибыли. Внутри сообщества производителей и внутри сообщества инвесторов имеется конкуренция.

Эволюция в модели происходит в течение ряда поколений. Продолжительность жизни каждого поколения равна N_T периодам. В конце каждого периода инвесторы принимают решение о вкладах, которые они будут делать в следующий период. Для этого инвесторы оценивают в течение определенного количества итераций, какую прибыль можно получить от того или иного производителя, учитывая при этом, как в процессе итераций меняются ожидаемые капиталы производителей в зависимости от намерений других инвесторов. Вся информация о капиталах производителей и намерениях инвесторов открыта для сообщества. Такая прозрачность является ключевой особенностью данной модели. В конце периода производитель возвращает капитал, вложенный инвестором, и распределяет часть своей прибыли между инвесторами пропорционально их вкладам. В конце каждого поколения происходит отбор производителей и инвесторов в новое поколение в соответствии с их капиталами.

Каждый инвестор имеет параметр, отвечающий за степень доверия к производителю. Степени доверия в начале поколения формируют генотипы сообщества инвесторов. В качестве генотипа для сообщества производителей выступает эффективность производства. Генотип задается в момент рождения агента и не меняется в течение его жизни. Генотипы потомков отличаются от генотипа родителя небольшими мутациями. Результаты компьютерного моделирования демонстрируют, что в результате эволюции средняя эффективность производителей возрастает. В новую популяцию инвесторов отбираются более успешные инвесторы.

В модели инвесторы могут обучаться путем настройки текущих степеней доверия производителям. В частности, степень

доверия инвестора к производителю плавно увеличивается при росте получаемой от данного производителя прибыли. Соответственно, у инвесторов в процессе обучения и взаимодействия с производителями будет формироваться определенное «мнение» о каждом производителе.

Поведение коллектива агентов в условиях децентрализации исследовалось в ряде работ, в частности, в хорошо известной книге В.И. Варшавского, Д.А. Поспелова «Оркестр играет без дирижера» [3]. Формы агрессивной и конструктивной конкуренции между отдельными индивидуумами в рамках агент-ориентированного подхода анализировались в работах [4, 5]. Наша модель основана на подходе работ [6, 7], в которых использовались легкие агенты-посланники для оптимизации работы производственного цеха и маршрутизации движения автомобилей в городе. В нашей работе легкие агенты используются для обмена информацией между инвесторами и производителями.

Путем компьютерного моделирования продемонстрирована эффективность предложенной схемы взаимодействий и выгодность сотрудничества между экономическими агентами в условиях конкуренции.

1. Описание эволюционной модели

В работе рассматриваются две эволюционирующие популяции агентов, которые взаимодействуют в конкурентной среде: *агенты-инвесторы* и *агенты-производители*. Поясним сначала, как в модели используются временные шкалы. Эволюция в модели происходит в течение ряда поколений, $t_g = 1, 2, \ldots T_g$, где T_g – количество поколений. Продолжительность жизни каждого поколения полагаем равной N_T тактов времени, будем называть каждый такт времени *периодом* $T = 1, 2, \ldots, N_T$. Аналогично, модели без эволюции, здесь в конце каждого периода T каждый инвестор принимает решение о вкладах, которые он будет делать в следующий период T+1. Для нахождения этого решения выполняется серия итераций, которые обозначаются $t = 1, 2, \ldots, t_{max}$, где $t_{\max}$ – максимальное количество итераций внутри периода. Таким образом, имеем *три временные шкалы*: 1) поколения агентов (T_g), 2) периоды (N_T), 3) итерации (t_{max}). Каждое поколение агентов живет N_T периодов, в течение каждого периода выполняется t_{max} итераций, всего поколений T_g.

Пусть количество агентов в популяциях N (инвесторы) и M (производители) и каждый из агентов имеет определенный капитал K_{inv} (инвесторы) или K_{pro} (производители). Изложим подробнее жизненный цикл популяции агентов в *течение одного поколения.*

В начале каждого периода T отдельный инвестор делает вклад в m наиболее выгодных производителей ($m \le M$). В конце периода производитель возвращает каждому инвестору капитал, вложенный инвестором, а также распределяет полученную им прибыль между инвесторами пропорционально их вкладам, при этом определенная доля прибыли остается у производителя.

В конце периода каждый инвестор принимает решение: какой капитал вложить в того или иного производителя в следующий период. Для того, чтобы принять решение с учетом намерений других инвесторов организуется итеративный процесс, который будет подробно описан ниже.

Обмен информацией между инвесторами и производителями в модели происходит с помощью агентов-посланников, аналогичных тем, которые использовались в работе [6]. Имеется два типа агентов-посланников: *агенты-разведчики* и *агенты намерений*.

Если капитал инвестора или производителя стал меньше определенного малого порога Th_{min_inv} или Th_{min_pro}, то такой инвестор или производитель прекращает свою деятельность.

Генотипы агентов и текущие степени доверия.

Каждый инвестор имеет параметр, отвечающий за степень доверия к каждому производителю, d_{ij} – текущая степень доверия j-го инвестора i-му производителю. Степени доверия в начале поколения формируют генотип сообщества инвесторов $G = (g_{ij})$, $i = 1, \ldots, M$, $j = 1, \ldots, N$. Каждый столбец матрицы G представляет собой генотип отдельного инвестора. В начале рассматриваемого процесса эволюции генотипы g_{ij} задаются случайным образом и равномерно распределены в интервале [0, 1]. Генотип задается в момент рождения агента и не меняется в течение его жизни. Генотипы потомков отличаются от генотипа родителя небольшими мутациями. Таким образом, генотипы g_{ij} изменяются только посредством эволюции, а текущие степени доверия d_{ij}

могут меняться от периода к периоду в **процессе обучения**. При этом в момент рождения текущая степень доверия агента-инвестора совпадает с генотипом $d_{ij} = g_{ij}$.

Математическая модель.

Считаем, что перед началом периода T i-й производитель имеет собственный исходный капитал C_{i0}. К капиталу каждого производителя добавляются вклады от инвесторов. Будем полагать, что производитель вкладывает в производство весь имеющийся у него к началу периода капитал C_i:

$$C_i = C_{i0} + \sum_{j=1}^{N} C_{ij} , \qquad (1)$$

где C_{ij} – капитал, вложенный j-м инвестором в i-го производителя в начале периода. Считаем, что зависимость прибыли производителя от его текущего капитала, определяется по формуле: $Pr_i(C_i) = k_i F(C_i)$, где функция F одинакова для всех производителей, а коэффициент k_i характеризует эффективность производства i-го производителя. В данной работе считалось, что функция $F(x)$ имеет вид:

$$F(x) = \begin{cases} ax, \text{ если } ax \le Th \\ Th, \text{ если } ax > Th \end{cases} , \qquad (2)$$

где a – положительный параметр, Th – порог функции $F(x)$.

В конце периода T производитель возвращает инвесторам вложенный ими капитал. Кроме того, производитель выплачивает инвесторам часть полученной им прибыли. Причем j-му инвестору отдается часть прибыли, пропорциональная сделанному им вкладу в данного производителя:

$$Pr_{ij} = k_{repay} Pr_i(C_i) \frac{C_{ij}}{\sum_{l=1}^{N} C_{il}} , \qquad (3)$$

где C_i – текущий капитал (в начале периода) i-го производителя, k_{repav} – параметр, характеризующий долю выплат прибыли инвесторам, $0 < k_{repay} < 1$. Сам производитель получит остальную часть своей прибыли Pr_i, равную:

$$Pr_pro_i = (1 - k_{repay}) Pr_i(C_i) . \qquad (4)$$

Схема итеративного процесса принятия решения инвесторами.

Итеративный процесс, в течение которого определяются вклады инвесторов в производителей, состоит в следующем. *На первой итерации* инвесторы рассылают агентов-разведчиков по всем производителям и определяют, какой капитал имеется у каждого производителя в данный момент времени. Причем на первой итерации не учитываются вклады других инвесторов в производителей. Далее инвесторы оценивают величины A_{ij}, характеризующие прибыль, ожидаемую от i-го производителя в течение нового периода T. Эти величины A_{ij} равны:

$$A_{ij} = d_{ij} Pr_{ij} = d_{ij} k_{repay} k_i F(C'_{i0}) \frac{C_{ij}}{\sum_{l=1}^{N} C_{il}} , \qquad (5)$$

где d_{ij} – текущая степень доверия j-го инвестора к i-му производителю, C_{il} – капитал, вложенный l-м инвестором в i-го производителя, C'_{i0} – предполагаемый исходный капитал i-го производителя в начале следующего периода (пока без учета вкладов инвесторов), k_{repay} – доля выплат инвестору.

Затем инвестор ранжирует всех производителей в соответствии с величинами A_{ij} и выбирает m наиболее выгодных производителей, т.е. тех производителей, которым соответствуют большие величины A_{ij}. Далее j-й инвестор *формирует намерение* распределить весь свой капитал $K_{inv\,j}$ по всем выбранным производителям, пропорционально полученным оценкам A_{ij}. А именно, намечается, что вклад j-го инвестора в i-го производителя C_{ij} будет равен:

$$C_{ij} = K_{inv\,j} \frac{A_{ij}}{\sum_{l=1}^{M} A_{lj}} . \qquad (6)$$

Для невыбранных производителей формально полагалось $A_{ij} = 0$.

На второй итерации каждый инвестор с помощью агентов намерений оповещает тех производителей, которых он выбрал для инвестиций, о величине капитала, который он намеревается вложить в каждого из производителей.

На основе этих данных производители оценивают свой *новый исходный капитал* C'_{i0}, который они ожидают после получения капитала от всех инвесторов, т.е. у производителя формируется оценка суммы $\sum_{l=1}^{N} C_{il}$ и новая оценка своего капитала в соответствии с выражением (1).

Затем инвесторы снова высылают агентов-разведчиков по всем производителям и

оценивают новые капиталы производителей C'_{i0} с учетом намерений других инвесторов, а также суммы $\sum_{l=1}^{N} C_{il}$. Инвесторы делают оценки ожидаемой прибыли согласно выражению (5), в котором уже учитывается сумма намеченных вкладов всех инвесторов $\sum_{l=1}^{N} C_{il}$. Далее каждый инвестор ранжирует производителей, и капитал инвестора распределяется пропорционально *новым полученным оценкам* A_{ij} согласно выражению (6). Инвесторы снова рассылают агентов намерений для того, чтобы сообщить производителям намеченные величины вкладов.

Делается достаточно большое число таких итераций, после чего итерации заканчиваются, и каждый инвестор принимает окончательное решение, какие вложения сделать на следующий период T. Окончательные вклады равны величинам C_{ij}, полученным инвесторами на последней итерации.

В конце каждого периода T капиталы производителей пересчитываются с учетом амортизации (например, это может быть амортизация оборудования производителя) $K_{pro}(T+1) = k_{amr}K_{pro}(T)$, где k_{amr} – коэффициент амортизации $(0 < k_{amr} \le 1)$. Аналогично учитываются расходы инвесторов (для удобства соответствующие величины будем называть коэффициентами инфляции) и пересчитывается капитал инвесторов $K_{inv}(T+1) = k_{inf}K_{inv}(T)$, где k_{inf} – коэффициент инфляции $(0 < k_{inf} \le 1)$.

Если капитал инвестора или производителя стал меньше определенного малого порога, то такой инвестор или производитель погибает. При этом возможно уменьшение численности инвесторов и/или производителей и даже разрушение всей экономической системы.

Эволюция.

Эволюция включает отбор инвесторов и производителей в соответствии с их приспособленностями и мутации. Рассматривается дарвиновская эволюция. Количество агентов от поколения к поколению не меняется (если не происходит вымирания всего экономического сообщества, см. выше). То есть в новое поколение отбирается ровно столько же агентов, сколько было в предыдущем. При этом агенты с более высокой приспособленностью имеют большую вероятность быть отобранными. Некоторые агенты могут быть отобраны в новое поколение несколько раз, это означает, что в новом поколении будет несколько потомков данного агента. Получая прибыль, инвесторы накапливают опыт и обучаются. Обучение происходит без учителя, путем настройки *степеней доверия к* производителям.

Схема эволюции.

Подробно изложим *эволюционный переход к следующему поколению*. Как было сказано выше, эволюция происходит в течение ряда поколений $t_g = 1, 2, \ldots T_g$, где T_g – количество поколений. В конце каждого поколения происходит отбор производителей и инвесторов в новое поколение в соответствии с их функциями приспособленностей $f(K_{pro})$ и $f(K_{inv})$. Будем считать, что **функция приспособленности** имеет вид $f(K) = K$, где K – конечный (в конце поколения) капитал инвестора или производителя. Для отбора используется *рулеточный метод* (roulette wheel selection) или метод отбора с вероятностями, пропорциональными приспособленностям (fitness-proportionate selection) [8]. Число производителей и инвесторов при отборе остается постоянным. Сначала делается отбор производителей, а затем инвесторов. Некоторые агенты могут иметь несколько «потомков». Конечный (в конце поколения) капитал «родителя» при этом делится между «потомками» поровну. Потомками производителей наследуется эффективность производства родителя с мутациями, т. е. $k_l(t_g + 1) = k_i(t_g) + rand_l$, где l – номер потомка в новом поколении, i – номер родителя, $rand_l = \tau P_{mut}$, где τ – нормально распределенная случайная величина с нулевым средним и стандартным отклонением $\tau_0 = 1$, P_{mut} – параметр интенсивности мутаций.

Аналогично делается отбор агентов инвесторов. Конечный капитал родителя инвестора делится между «потомками» поровну. Генотип родителя инвестора наследуется потомком с небольшими мутациями, $g_{il}(t_g + 1) = g_{ij}(t_g) + rand_l$, где l – номер потомка, j – номер родителя, i меняется от 1 до M, l и j фиксированы, а $rand_l = \tau P_{mut}$,

где τ − нормально распределенная случайная величина с нулевым средним и стандартным отклонением $\tau_0 = 1$, P_{mut} – параметр интенсивности мутаций. Если величина g_{ij} или k_i в результате мутаций стала отрицательной, то она становится равной 0.

Таким образом, генотип инвесторов есть начальные степени доверия g_{ij}, генотип производителя есть его эффективность k_i, а конечный капитал инвестора или производителя $K_{inv\,j}$, $K_{pro\,i}$ есть его фенотип, определяющий его приспособленность. Эволюция дарвиновская, так как по наследству передаются потомкам генотипы (с малыми мутациями), а не фенотипы.

Обучение инвесторов.

В течение жизни поколения инвесторы обучаются. Обучение происходит в конце каждого периода T. Получив прибыль от производителя в конце текущего периода T инвестор пересчитывает текущие степени доверия к производителям по следующему правилу:

$$d_{ij}(T+1) = d_{ij}(T) + \alpha Q\left(P_{ij}\right)\left[1 - d_{ij}(T)\right] - \\ -\gamma d_{ij}(T) \qquad (7)$$

где α ($\alpha > 0$) – параметр скорости обучения, $Q(x) = x / (H + x)$, P_{ij} – прибыль, полученная j-м инвестором от i-го производителя, H – параметр функции Q, этот параметр порядка характерной прибыли P_{ij}. Таким образом текущая степень доверия зависит от прибыли, которую инвестор получает от производителя. Последнее слагаемое характеризует уменьшение степени доверия, «забывание» навыка.

2. Результаты компьютерного исследования модели

Описанная выше модель была исследована с помощью компьютерного моделирования. Ниже приведены общие параметры моделирования, которые использовались при проведении расчетов:

- общее число поколений $T_g = 100$;
- число периодов внутри каждого поколения: $N_T = 100$;
- число итераций в каждом периоде: $t_{\max} = 1, \ldots, 50$;
- минимальные пороги капиталов производителей и инвесторов (если капитал становился ниже этих порогов, то соответствующий производитель или инвестор погибал): $Th_{min_pro} = Th_{min_inv} = 0.01$;
- количество производителей и инвесторов в каждом поколении: $M = N = 100$;
- максимальное число производителей m, в которое мог вкладывать капитал инвестор $m = 100$;
- доля выплат из полученной производителями прибыли инвесторам $k_{repay} = 0.6$;
- параметр функции $F(x)$, определяющей величину прибыли: $a = 0.01$;
- порог функции $F(x)$: $Th = 100$;
- $H = 100$ – параметр функции Q;
- параметр скорости обучения: $\alpha = 0.5$;
- параметр уменьшения степени доверия $\gamma = 0.1$;
- интенсивность мутаций $P_{mut} = 0.1$.

Величины k_i, характеризующие эффективность производителей, исходно (в начале расчета) были случайными, равномерно распределенными в интервале [0, 1].

Сходимость итеративного процесса для эволюционной модели.

Проверим сначала сходимость итеративного процесса. На Рис. 1 показаны результаты компьютерного моделирования при разных количествах итераций внутри одного периода. Полученные результаты относятся к первым двадцати поколениям. Видно, что, когда производители и инвесторы не делятся информацией между собой, суммарный капитал, который накоплен сообществом, значительно ниже. В частности, если количество итераций $t_{max} = 50$, суммарный капитал сообщества будет на 70% больше, чем в сообществе без итеративных оценок.

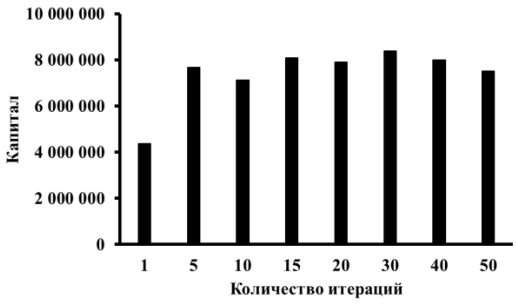


Рис. 1. Зависимость величины суммарного капитала (усл. единицы), накопленного сообществом от количества итераций t_{max} в периоде ($T_g = 20$).

Следует также отметить, что влияние итераций снижается с ростом капитала у сообщества, что будет показано ниже.

Динамика капиталов инвесторов и производителей.

На Рис. 2 представлена динамика суммарных капиталов инвесторов и производителей без усреднения для типичных параметров расчетов.

Видно, что капиталы инвесторов и производителей растут до 20 поколения, а далее рост прекращается. Это связно с тем, что функция прибыли $F(x)$ имеет линейно-пороговый вид. Когда капиталы производителей достигают порога, прибыль, получаемая производителями, не зависит уже от размера вкладов инвесторов. Для каждого периода она будет иметь постоянный размер. При этом при эволюции родитель делит свой капитал между потомками и ограниченный размер прибыли не дает возможности для дальнейшего роста при заданных параметрах модели.

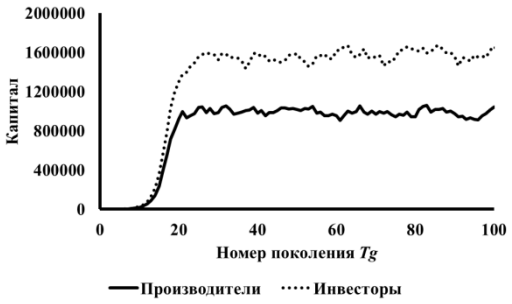


Рис. 2. Динамика суммарных капиталов (усл. единицы) инвесторов и производителей. Количество итераций в каждом периоде t_{max} = 50.

Генотипы производителей.

Ранее было сказано, что генотипы производителей — это их эффективности. В процессе эволюции генотипы меняются.

В каждое новое поколение отбираются агенты более приспособленные, те, которые имеют больший капитал.

Размеры капиталов зависят в свою очередь от эффективностей. Эксперименты показывают, что в результате *отбора и обучения* средний коэффициент эффективности производителей k_i растет. Ниже на Рис. 3 представлены результаты экспериментов с итеративными оценками.

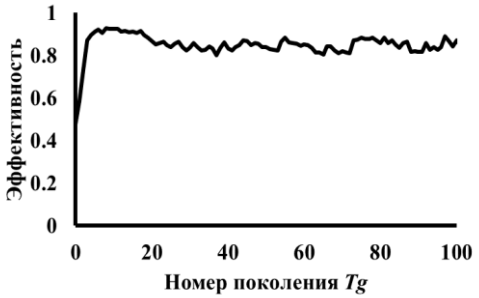


Рис. 3. Динамика среднего коэффициента эффективности производителей (t_{max} = 50).

Следует отметить, что средний коэффициент эффективности улучшается и при моделировании без итеративных оценок.

Влияние итеративных оценок.

Для того, чтобы показать, как итеративные оценки влияют на суммарный капитал инвесторов и производителей, моделирование проводилось в двух режимах 1) без итеративных оценок (t_{max} = 1) и 2) с итеративными оценками (t_{max} = 50).

Рис. 4. Влияние итеративных оценок на суммарные капиталы инвесторов и производителей.

На Рис. 4 представлено на какой процент увеличатся суммарные капиталы инвесторов и производителей, если инвесторы используют итеративные оценки. Видно, что с 1-го до 20-го поколения влияние итеративных оценок существенное. Далее, когда производители достигают предела роста, влияние итеративных оценок колеблется в некотором диапазоне.

На Рис. 5 показано как отдельный инвестор меняет размер своего вклада от итерации к итерации внутри одного периода в различных производителей.

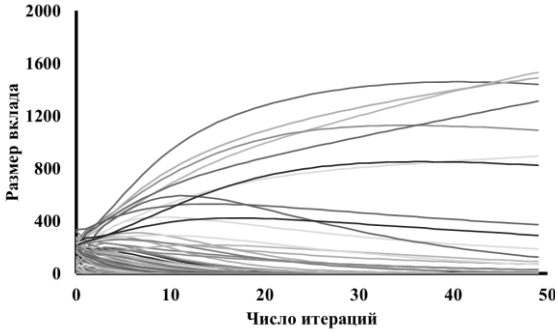


Рис. 5. Изменение размера вклада (усл. единицы) отдельного инвестора от итерации к итерации. Разные кривые соответствуют разным производителям.

Рис. 6. Количество инвесторов у каждого производителя (T_g = 100, N_T=100).

Аналогичное поведение наблюдается и у остальных инвесторов. При этом поведение инвесторов в эволюционной модели отличается от поведения инвесторов описанного в [2]. В данном случае производитель, который выбирается для максимального вклада у разных инвесторов, разный. В модели без эволюции инвесторы ранжировали производителей одинаково.

С этим результатом связан еще один интересный эффект, который наблюдается в эволюционной модели: инвесторы распределяют свои вклады таким образом, что в каждого производителя делают вклады не более одного-трех инвесторов, при этом у инвесторов есть возможность делать вклады во всех производителей. Т.е. в результате самоорганизации все сообщество разбивается на небольшие группы: 1) один производитель – один инвестор, 2) один производитель – два инвестора и 3) один производитель – три инвестора. При этом следует учесть, что это происходит в последних поколениях, где в основном все производители имеют высокую эффективность и большой капитал. Результаты представлены на Рис. 6. Видно, что в каждого производителя делают вклады не более 1-3-х инвесторов.

Заключение

Таким образом, исследованы принципы взаимодействия инвесторов и производителей в среде прозрачной экономической системы. Оригинальные черты модели: сотрудничество между инвесторами и производителями, открытость информации о капиталах и эффективностях производителей, а также о намерениях инвесторов вложить капиталы в тех или иных производителей, итеративный процесс формирования размеров вкладов. Наиболее важным результатом является разработка нового метода выгодного распределения капитала в конкурентной среде через сотрудничество. Разработаны принципы построения модели эволюции коллектива инвесторов и производителей, в которой инвесторы обучаются формировать степени доверия к производителям.

Данная работа выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «*35.16. Моделирование процессов формирования когнитивных способностей автономных агентов*» (№ 0065-2018-0002).

The study of the processes of self-organization in the evolutionary model of a transparent decentralized economy

Z.B. Sokhova, V.G. Red'ko

Abstract. The article suggests a model of interaction between investors and producers in a decentralized transparent economic system. The processes of evolution and self-organization in this community are analyzed. A specific mechanism for allocating investor capital between producers is proposed. An important feature of the model is transparency and decentralization. Investors and producers openly share information about their capital, efficiency and intentions with the help of light messenger agents. The model was studied using computer simulation, the results of which demonstrate the effectiveness of the mechanism under consideration.

Keywords: decentralization, multiagent systems, collective behavior, cooperation, evolution, transparent economy.

Литература

1. В.М. Полтерович. Позитивное сотрудничество: факторы и механизмы эволюции. «Вопросы экономики», (2016), № 11, 5 – 23.
2. V.G. Red'ko, Z.B. Sokhova. Iterative method for distribution of capital in transparent economic system. «Optical memory and neural networks (information optics)», v. 26 (2017), No. 3, 182 − 191.
3. В.И. Варшавский, Д.А. Поспелов. Оркестр играет без дирижера: размышления об эволюции некоторых технических систем и управлении ими. М., Наука, 1984.
4. В.Г. Редько, М.С. Бурцев, З.Б. Сохова, Г.А. Бесхлебнова. Моделирование конкуренции при эволюции многоагентной системы. «Искусственные общества» (электронный журнал), т. 2 (2007), № 2, 76 − 89. URL: http://abm.center/magazine/?ID=44
5. M. Burtsev, P. Turchin. Evolution of cooperative strategies from first principles. «Nature», v. 440 (2006). No. 7087, 1041 – 1044.
6. R. Claes, T. Holvoet, D. Weyns. A decentralized approach for anticipatory vehicle routing using delegate multiagent systems. IEEE Transactions on Intelligent Transportation Systems. v. 12 (2011), No. 2, 364 – 373.
7. T. Holvoet, P. Valckenaers. Exploiting the environment for coordinating agent intentions. In: Environments for Multi-Agent Systems III, Lecture Notes in Artificial Intelligence, Springer: Berlin et al. v. 4389 (2007). 51 – 66.
8. В.Г. Редько. Моделирование когнитивной эволюции. М., Ленанд, 2015.

Использование метода последовательных приближений при композиционном моделировании разработки месторождений углеводородов для определения концентраций компонентов

И.В. Афанаскин[1], С.Г. Вольпин[2], А.В. Королев[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]ivan@afanaskin.ru, [2]sergvolpin@gmail.com, [3]alexandre.korolev@mail.ru

Аннотация. Рассмотрена система уравнений сохранения для моделирования многокомпонентной фильтрации и обобщенное уравнение состояния. Подробно описан алгоритм определения концентраций компонентов в фазах с помощью метода последовательных приближений. Рассмотрен пример реализации указанной схемы для моделирования попеременной закачки газа и воды при разработке нефтяного месторождения. Получено косвенное подтверждение корректности модели.

Ключевые слова: композиционная модель, многокомпонентная фильтрация, фазовые переходы.

Введение

Композиционные (многокомпонетные) модели фильтрации используются при подробном моделировании залежей, содержащих легкие углеводороды (конденсат и газ) в том случае, когда необходимо тщательно описывать массообмен между фазами, либо когда пластовые флюиды содержат ценные неуглеводородные компоненты. Такие модели являются также основой для изучения множества методов увеличения нефтеотдачи.

Одна из первых изотермических композиционных моделей трехмерной трехфазной фильтрации (1980 г.) была предложена в работе Coats K.H. [1], численное решение системы нелинейных уравнений было осуществлено с помощью полностью неявной (SS) вычислительной схемы (Simultaneous Solution, современное название - Fully Implicit).

Важнейшим для композиционных моделей моментом, отличающим их от классических моделей трехфазной фильтрации несмешивающихся жидкостей (так называемая модель «черной нефти» или модель Маскета-Мереса) является прямой учет фазовых переходов компонентов между нефтяной и газовой фазами с помощью механизма уравнений состояния.

Эта работа посвящена одному из простых методов расчета концентраций компонентов в нефтяной и газовой фазах с применением уравнений состояния и метода последовательных приближений.

1. Система уравнений трехфазной многокомпонентной фильтрации нефти, газа и воды

Рассмотрим математическую модель трехфазной многокомпонентной фильтрации нефти, газа и воды с фазовыми переходами. Система уравнений представляет собой комбинацию уравнений материального баланса и некоторых ограничений [1, 2-5]:

$$\frac{\partial}{\partial t}\left[\phi\left(\xi_o S_o x_i + \xi_g S_g y_i\right)\right] + \\ + \nabla\left(\xi_o x_i \vec{W}_o + \xi_g y_i \vec{W}_g\right) + q_i = 0$$

$$i = 1,2,\ldots, N_c, \qquad (1)$$

$$\frac{\partial}{\partial t}\left[\phi \xi_w S_w\right] + \nabla\left(\xi_w \vec{W}_o\right) + q_w = 0, \qquad (2)$$

$$f_i^o - f_i^g = 0,\ i = 1,2,\ldots, N_c, \qquad (3)$$

$$\vec{W}_\alpha = -kk_{r\alpha}\mu_\alpha^{-1}\left(\nabla P_\alpha + \rho_\alpha g \nabla D\right), \quad \alpha = o, g, w, \tag{4}$$

$$\sum_{i=1}^{N_c} x_i = 1, \tag{5}$$

$$\sum_{i=1}^{N_c} y_i = 1, \tag{6}$$

$$S_o + S_g + S_w = 1, \tag{7}$$

$$P_o - P_w = P_{cwo}, \tag{8}$$

$$P_g - P_o = P_{cog}, \tag{9}$$

$$P_g - P_w = P_{cwo} + P_{cog}, \tag{10}$$

здесь N_c - общее число компонентов в нефтяной и газовой фазах, $k(X,P)$ и $\phi(X,P)$ - соответственно тензор абсолютной проницаемости и пористость пласта, $X=(x,y,z)$ - точка в трехмерном пространстве, t - время, P_α - фазовое давление, S_α - фазовая насыщенность, x_i и y_i - мольная концентрация i-ого компонента соответственно в нефтяной и газовой фазе, ξ_α - мольная фазовая плотность, μ_α - динамическая вязкость фазы α, ρ_α - массовая плотность фазы α, $P_{cwo}(S_w)$ - капиллярное давление в системе нефть-вода, $P_{cog}(S_o)$ - капиллярное давление в системе нефть-газ, $k_{r\alpha}$ - относительная фазовая проницаемость для фазы α, q_i - компонентная плотность источников и стоков, q_w - плотность источников и стоков по воде, $D(X)$ - превышение точки пласта над некоторой горизонтальной плоскостью, g - ускорение силы тяжести, f_i^o и f_i^g - летучести компонентов нефтяной и газовой фазы, ∇ - оператор Гамильтона.

В системе (1)-(10) количество уравнений составляет $2N_c+9$, что соответствует количеству неизвестных $x_1, x_2, \ldots, x_{N_c}$; $y_1, y_2, \ldots, y_{N_c}$; P_o, P_g, P_w; S_o, S_g, S_w; $\vec{W}_o, \vec{W}_g, \vec{W}_w$.

Количество уравнений и неизвестных можно сократить до $2N_c+4$, подставляя (4), а затем (8)-(10) в систему (1), (2).

За опорное давление принято давление в нефти $P \equiv P_o$.

Таким образом, в качестве неизвестных функций приняты:

$$x_1, x_2, \ldots, x_{N_c};\ y_1, y_2, \ldots, y_{N_c};\ P;$$
$$S_o, S_g, S_w. \tag{11}$$

В данной работе система уравнений (1)-(10) для отыскания неизвестных (11) решается способом, приведенным в [5].

2. Обобщенное уравнение состояния в рамках композиционной модели изотермической фильтрации

Для определения свойств нефти и газа, а также мольных концентраций компонентов в нефтяной и газовой фазах применяют уравнения состояния.

Использованию уравнений состояния при моделировании внутрипластовых процессов посвящено множество работ [6-10].

Наиболее «популярными» в нефтегазовой отрасли уравнениями состояния являются:

1. уравнение Пенга-Робинсона (PR);
2. уравнение Редлиха-Квонга (RK);
3. уравнение Соаве-Редлиха-Квонга (SRK).

Эти уравнения состояния при моделировании применяют в обобщенной форме с помощью уравнения Мартина. Воспользуемся формулировкой, приведенной в [11].

Обобщенная форма уравнения состояния выглядит следующим образом:

$$Z^3 + E_2 Z^2 + E_1 Z + E_0 = 0. \tag{12}$$

Коэффициенты этого уравнения E_2, E_1 и E_0 определяются по правилам, описанным ниже.

Максимальный положительный корень уравнения (12) Z_g равен коэффициенту сверхсжимаемости газа, а минимальный положительный корень Z_o – коэффициенту сверхсжимаемости нефти.

Для расчета коэффициентов уравнения (12) для каждого компонента необходимо задать пластовую температуру T и следующие константы:

1. T_{ci} - критическая температура компонента i;
2. P_{ci} - критическое давление компонента i;
3. ω_i - ацентрический фактор компонента i;
4. c_{ij} - коэффициенты парного взаимодействия компонентов i и j.

Для расчета коэффициентов уравнения (12) определим коэффициенты m_1, m_2, Ω_{a0} и Ω_b в зависимости от типа уравнения в Табл. 1.

Приведенные давление и температура для каждого компонента рассчитываются как:

$$P_{ri} = \frac{P}{P_{ci}},\ T_{ri} = \frac{T}{T_{ci}}. \qquad (13)$$

Таблица 1

Постоянные коэффициенты уравнений состояния

EOS	m_1	m_2	Ω_{a0}	Ω_b
RK	0	1	0,4274802	0,08664035
SRK	0	1	0,4274802	0,08664035
PR	$1+\sqrt{2}$	$1-\sqrt{2}$	0,457235529	0,07796074

В зависимости от типа уравнения параметр Ω_a вычисляется как:

- RK

$$\Omega_a(T,i) = \Omega_{a0} T_{ri}^{-1/2}, \qquad (14)$$

- SRK

$$\Omega_a(T,i) = \Omega_{a0} \times \left[1+\left(0{,}48+1{,}574\omega_i - 0{,}176\omega_i^2\right)\times \left(1-T_{ri}^{-1/2}\right)\right]^2 \qquad (15)$$

- PR

$$\Omega_a(T,i) = \Omega_{a0}\left[1+\left(0{,}37464+1{,}5422\omega_i - 0{,}37464+1{,}5422\omega_i\right)\left(1-T_{ri}^{-1/2}\right)\right]^2 \qquad (16)$$

Теперь определим следующие коэффициенты:

$$A_i = \Omega_a(T,i)\frac{P_{ri}}{T_{ri}^2},\ B_i = \Omega_b \frac{P_{ri}}{T_{ri}},$$
$$A_{jk} = \left(1-c_{jk}\right)\sqrt{A_j A_k}. \qquad (17)$$

Теперь мы можем рассчитать непосредственно параметры уравнения состояния:

$$A = \sum_{j=1}^{N_c}\sum_{k=1}^{N_c} x_j x_k A_{jk},\ B = \sum_{j=1}^{N_c} x_j B_j, \qquad (18)$$

$$E_2 = \left(m_1+m_2-1\right)B - 1, \qquad (19)$$

$$E_1 = A - \left[2\left(m_1+m_2-1\right)\right]B^2 - \left(m_1+m_2\right)B, \qquad (20)$$

$$E_0 = -\left[AB + m_1 m_2 B^2\left(B+1\right)\right]. \qquad (21)$$

Коэффициенты летучести рассчитываются с помощью следующих выражений:

$$\ln\left(\frac{f_i^o}{Px_i}\right) = -\ln\left(Z_o - B\right) + \frac{A}{\left(m_1-m_2\right)B}\times \times\left(\frac{2\Sigma_i}{A}+\frac{B_i}{B}\right)\ln\left(\frac{Z_o+m_2 B}{Z_o+m_1 B}\right) + \frac{B_i}{B}\left(Z_o-1\right) \qquad (22)$$

$$\ln\left(\frac{f_i^g}{Py_i}\right) = -\ln\left(Z_g - B\right) + \frac{A}{\left(m_1-m_2\right)B}\times \times\left(\frac{2\Sigma_i}{A}+\frac{B_i}{B}\right)\ln\left(\frac{Z_g+m_2 B}{Z_g+m_1 B}\right) + \frac{B_i}{B}\left(Z_g-1\right), \qquad (23)$$

где $\Sigma_i = \sum_{j=1}^{N_c} A_{ij} x_j$.

Чтобы термодинамическая система пребывала в равновесии, необходимо, чтобы летучести компонентов в жидкой и газовой углеводородных фазах были равны для каждого компонента:

$$f_i^o - f_i^g = 0,\ i = 1,2,\ldots,N_c. \qquad (24)$$

Мольная концентрация i-го компонента в углеводородной системе z_i определяется как:

$$z_i = x_i L + y_i V,\ L = \frac{\xi_o S_o}{\xi_o S_o + \xi_g S_g},$$
$$V = \frac{\xi_g S_g}{\xi_o S_o + \xi_g S_g},\ \sum_{i=1}^{N_c} z_i = 1. \qquad (25)$$

Константы равновесия для каждого компонента определяют как:

$$K_i = \frac{y_i}{x_i}. \qquad (26)$$

Тогда мольные доли каждого компонента в жидкой и газовой фазе задаются соотношениями:

$$x_i = \frac{z_i}{1+\left(K_i-1\right)V}, \qquad (27)$$

$$y_i = \frac{K_i z_i}{1+\left(K_i-1\right)V}. \qquad (28)$$

3. Определение концентраций компонентов в фазах с помощью метода последовательных приближений

Определение мольных концентраций компонента в жидкой и газовой фазах x_i и y_i, а также мольной доли «пара» V и жидкости L при заданных z_i, P и T можно реализовать следующим итерационным способом. Константы равновесия K_i, присваиваются каждому компоненту, i, из предыдущей итерации, а если их (предыдущих итераций на данном шаге по времени) нет, то из формулы Вильсона, приведенной ниже:

$$K_i = \frac{1}{P_{ri}} \exp\left[5{,}37(1-\omega_i)\left(1-\frac{1}{T_{ri}}\right)\right]. \quad (29)$$

Затем при известных z_i и K_i решается уравнение для мольной доли «пара» V:

$$g(V) = \sum_{i=1}^{N_c} \frac{z_i(K_i - 1)}{1 + V(K_i - 1)} = 0. \quad (30)$$

Рассмотрим возможные значения корней уравнения (30).

1. $V < 0$ - однофазное ненасыщенное жидкое состояние. Необходимый критерий того, что при заданных значениях K_i смесь находится в ненасыщенном жидком состоянии:

$$\sum_{i=1}^{N_c} z_i K_i < 1. \quad (31)$$

2. $V = 0$ - насыщенное жидкое состояние (точка кипения). Необходимое условие однофазного насыщенного жидкого состояния:

$$\sum_{i=1}^{N_c} z_i K_i = 1. \quad (32)$$

3. $0 < V < 1$ - двухфазное парожидкостное состояние. Необходимые условия двухфазного состояния имеют вид:

$$\sum_{i=1}^{N_c} z_i K_i > 1, \ \sum_{i=1}^{N_c} z_i / K_i > 1. \quad (33)$$

4. $V = 1$ - однофазное состояние в виде насыщенного пара (газа) (точка росы). Необходимое условие нахождения в точки росы имеет вид:

$$\sum_{i=1}^{N_c} z_i / K_i = 1. \quad (34)$$

5. $V > 1$ - однофазное состояние в виде ненасыщенного газа. Необходимое условие существования смеси в ненасыщенном газовом состоянии имеет вид:

$$\sum_{i=1}^{N_c} z_i / K_i < 1. \quad (35)$$

При расчете парожидкостного равновесия методом последовательных приближений решение уравнения (30) следует искать не на отрезке $0 \le V \le 1$, а на более широком интервале. После нахождения V решают уравнения относительно мольных долей каждого компонента в жидкой и газовой фазе (27) и (28).

Расчеты летучести можно проводить лишь в том случае, если вычисленные на каждой итерации составы имеют физический смысл $0 < x_i^l < 1$, $0 < y_i^l < 1$, где l - номер итерации.

Мольная доля жидкости L рассчитывается как $L = 1 - V$.

Затем решается уравнение состояния (12), корни которого определяют коэффициенты сверхсжимаемости газовой и жидкой фазы.

Далее рассчитываются коэффициенты летучести с помощью выражений (22) и (23).

Если смесь находится в равновесии при данном давлении и температуре, то летучести компонентов в жидкой и газовой фазах должны быть равны:

$$f_i^o = f_i^g, \ i = 1,2,..., N_c.$$

Для проверки этого предположения применяется критерий сходимости:

$$\sum_{i=1}^{N_c} \left(\frac{f_i^o}{f_i^g} - 1\right)^2 < \varepsilon, \quad (36)$$

где ε - малое число, обычно $\varepsilon = 10^{-5}$.

Если равновесие не достигается, то производится новая оценка констант равновесия, и итерационный процесс продолжается с новым решением уравнения парожидкостного уравнения (30) и т.д. С использованием последовательной подстановки новую оценку константы равновесия можно записать так:

$$K_i^{l+1} = K_i^l \frac{f_i^o}{f_i^g}, \quad (37)$$

где l - номер итерации.

В процессе итераций могут выполняться любые из критериев (31)-(35). Однако выполнение одного из этих критериев на промежуточной итерации не означает, что смесь действительно находится в соответствующем фазовом состоянии. Возможны случаи, когда решение исходной системы уравнений равновесия показывает, что смесь находится в двухфазном равновесии $(0 < V < 1)$, но в процессе итерации значение V может быть как больше единицы, так и меньше нуля. Это возможно, в частности, из-за неточных начальных приближений констант равновесия, входящих в уравнение (30). Часто такая ситуация возникает при расчете фазовых равновесий вблизи точек кипения или росы

4. Пример использования рассмотренной модели

В качестве примера использования рассмотренной математической модели приведем моделирование попеременной закачки газа и воды для элемента пятиточечной системы заводнения, рис. 1, одного из месторождений Западной Сибири. Пласт представлен отложениями ачимовской свиты.

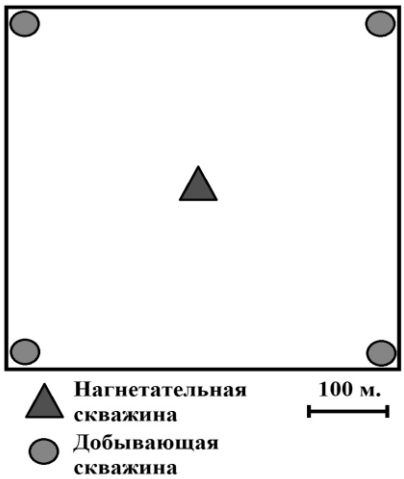


Рис. 1. Элемент пятиточечной системы расстановки скважин

Сетка модели прямоугольная, блочноцентрированная. Количество ячеек по осям X-Y-Z равно 20-20-12 штук соответственно, размеры ячеек равны 50-50-1,7 м. соответственно. Расстояние между добывающими скважинами 1000 м. Эффективная толщина пласта 20 м. Эффективная нефтенасыщенная толщина 10 м. По толщине пласт неоднороден по проницаемости и пористости. Поле проницаемости пласта характеризуется параметрами:

- ✓ математическое ожидание 70,4 мД;
- ✓ стандартное отклонение 151,3 мД;
- ✓ минимум 15,3 мД;
- ✓ максимум 460,6 мД.

Поле пористости характеризуется параметрами:

- ➢ математическое ожидание 0,192 д.ед.;
- ➢ стандартное отклонение 0,017 д.ед.;
- ➢ минимум 0,161 д.ед.;
- ➢ максимум 0,210 д.ед.

Коэффициент анизотропии проницаемости по вертикали 0,1 д.ед. Перфорирована верхняя, нефтенасыщенная часть пласта.

Начальная пластовая температура 92,2°С, начальное пластовое давление 255 ат. Вязкость нефти 0,62 сПз, объемный коэффициент нефти 1,16 ед., плотность нефти 847 кг/м^3. Объемный коэффициент воды 1,02 ед. Плотность газа 1,17 кг/нм^3. Давление насыщения нефти газом 98 ат., растворимость газа в нефти 65 м^3/м^3. Для моделирования PVT-свойств и фазовых равновесий использована модель Пенга-Робинсона. При расчете используются следующие компоненты:

1. N_2 - молекулярная масса 28,013,
2. CO_2 - молекулярная масса 44,01,
3. C_3 - молекулярная масса 44,097,
4. C_4 - молекулярная масса 58,124,
5. H_2O

и псевдокомпоненты:

1. C_1+C_2 - молекулярная масса 17,693,
2. C_5+C_6+C_7 - молекулярная масса 81,739,
3. C_{8+} - молекулярная масса 209,512.

Относительные фазовые проницаемости и капиллярное давление в системе вода-нефть приведены на Рис. 2, а в системе нефть-газ – на Рис. 3. Капиллярным давлением в системе нефть-газ пренебрегаем.

Внешние границы модели считаются непроницаемыми.

Добывающие скважины управляются дебитом жидкости. Скважины работают с различным дебитом жидкости. Нагнетательные скважины управляются закачкой. Дебиты и закачка соответствуют фактическим данным. Скважины в углу модели имеют дебит жидкости 1/4 от фактического. Вода и газ закачиваются по очереди циклами. Длительность периода закачки одной фазы равна одному месяцу. Расход газа 40 тыс.нм^3/сут. Расход воды 200 м^3/сут. Закачивается растворенный в нефти газ после сепарации или сухой газ, Табл. 2. Начальные мольные концентрации компонентов в углеводородной системе приведены в Табл. 3.

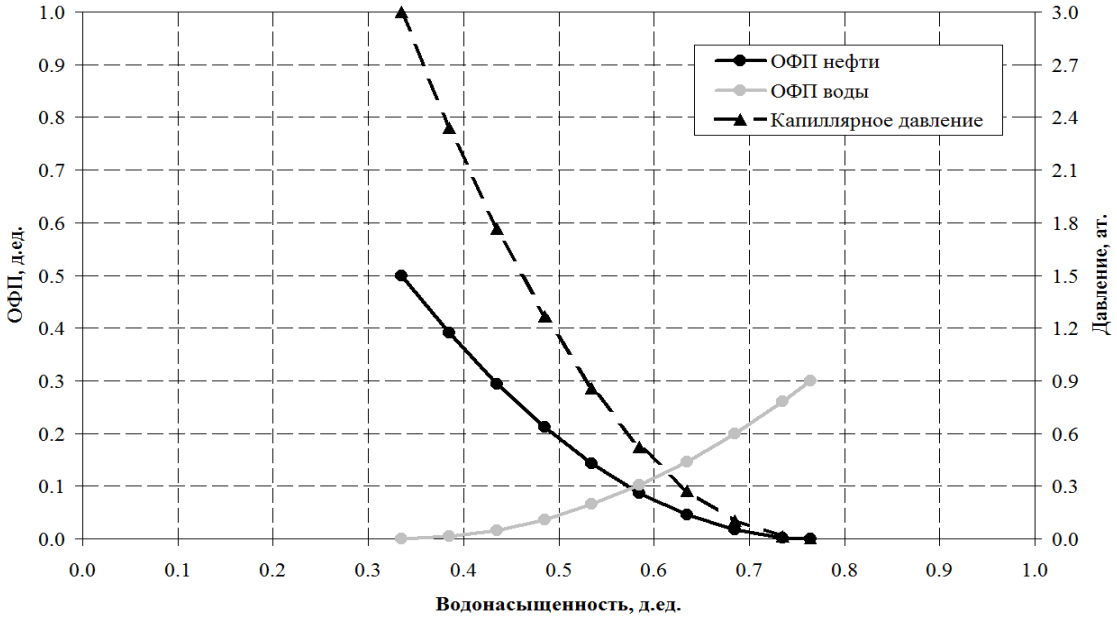


Рис. 2. Относительные фазовые проницаемости (ОФП) и капиллярное давление в системе вода-нефть

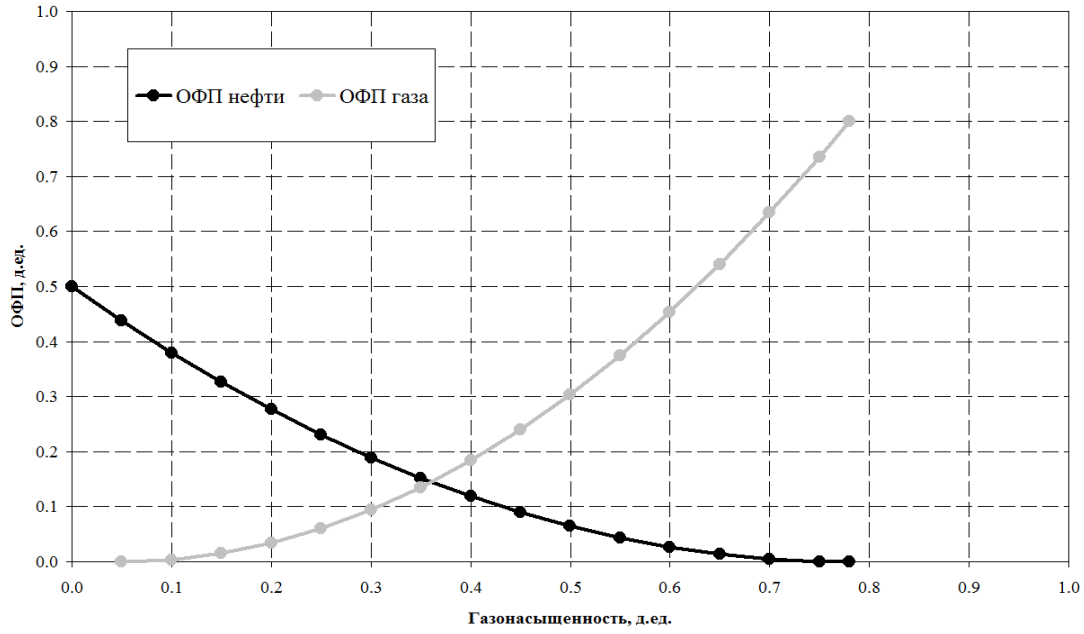


Рис. 3. Относительные фазовые проницаемости (ОФП) в системе нефть-газ

Таблица 2

Состав закачиваемого газа (мольные концентрации)

	N_2	CO_2	C_1+C_2	C_3	C_4	$C_5+C_6+C_7$	C_{8+}
Попутный газ	0,0191	0,0	0,9412	0,0122	0,0169	0,0106	0,0
Сухой газ	0,0	0,0	1,0	0,0	0,0	0,0	0,0

Таблица 3

Начальные мольные концентрации компонентов в углеводородной системе

	N_2	CO_2	C_1+C_2	C_3	C_4	$C_5+C_6+C_7$	C_{8+}
Углеводородная система	0,0039	0,0053	0,2615	0,0603	0,0679	0,1476	0,4535

Ограничения на работу добывающих скважин следующие:

1. Минимальный дебит нефти 1 $м^3$/сут.
2. Максимальная обводненность 98 %.
3. Максимальный газовый фактор 1000 $нм^3/м^3$.

Срок разработки – не более 25 лет.

Всего было рассмотрено 2 варианта в зависимости от рабочего агента.

Вариант 1. Рабочий агент – попутный газ и вода.

Вариант 2. Рабочий агент – сухой газ и вода.

Оба варианта закончены разработкой ранее 25 лет по причине прорыва закачиваемого газа в добывающие скважины в больших количествах.

На Рис. 4-8 приведены показатели разработки по группе добывающих скважин. Рис. 4. - суммарная добыча нефти. Рис. 5. - накопленная добыча нефти. Рис. 6. - газовый фактор. Рис. 7. - обводненность продукции. Рис. 8. - мольная концентрация компонентов $C_5+C_6+C_7$ и C_{8+} в добываемой нефти.

Резкие изменения параметров на Рис. 4-8 объясняются послойным прорывом закачиваемого газа и воды в скважины, а также отключением некоторых добывающих скважин из-за достижения экономических ограничений.

Видно, что вариант с закачкой попутного газа и воды дает лучшие показатели разработки, что косвенно говорит о корректности модели.

Заключение

Создана бета-версия гидродинамического симулятора для моделирования многофазной многокомпонентной (композиционной) фильтрации при разработке нефтяных, газовых, нефтегазовых и газоконденсатных месторождений. Приведен простой вариант реализации блока для моделирования фазовых переходов. Проведено тестирование алгоритма на синтетической модели, построенной на осредненных данных по реальному нефтяному месторождению.

Работа выполнена при поддержке гранта РФФИ 16–29–15105 офи_м.

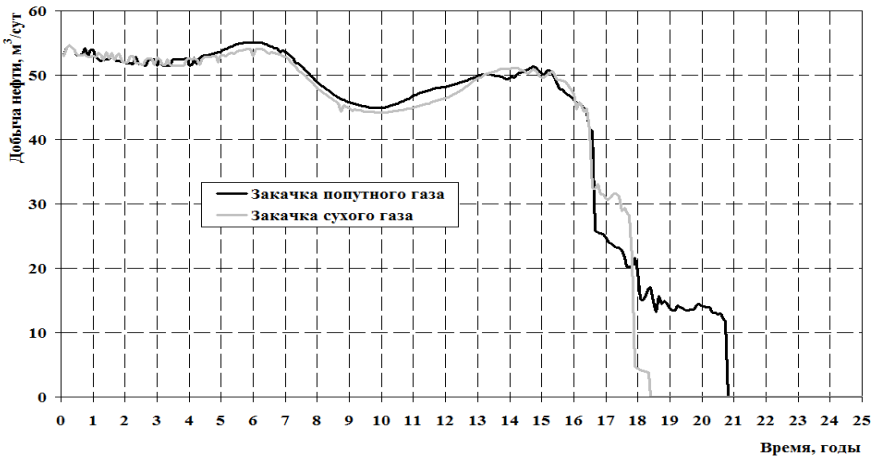


Рис. 4. Суммарная добыча нефти по группе скважин

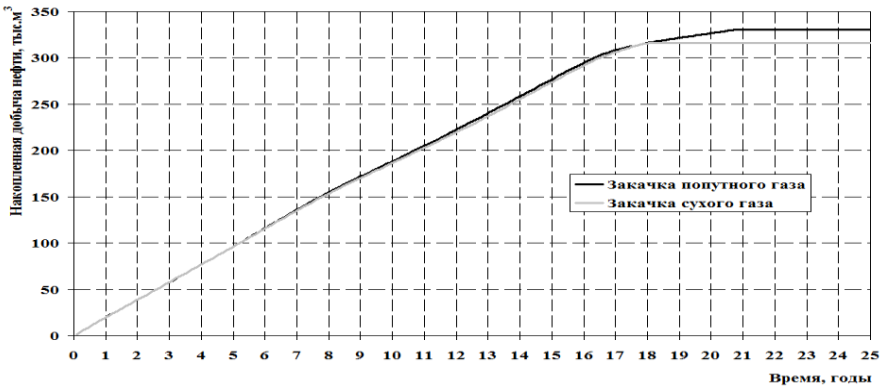


Рис. 5. Накопленная добыча нефти по группе скважин

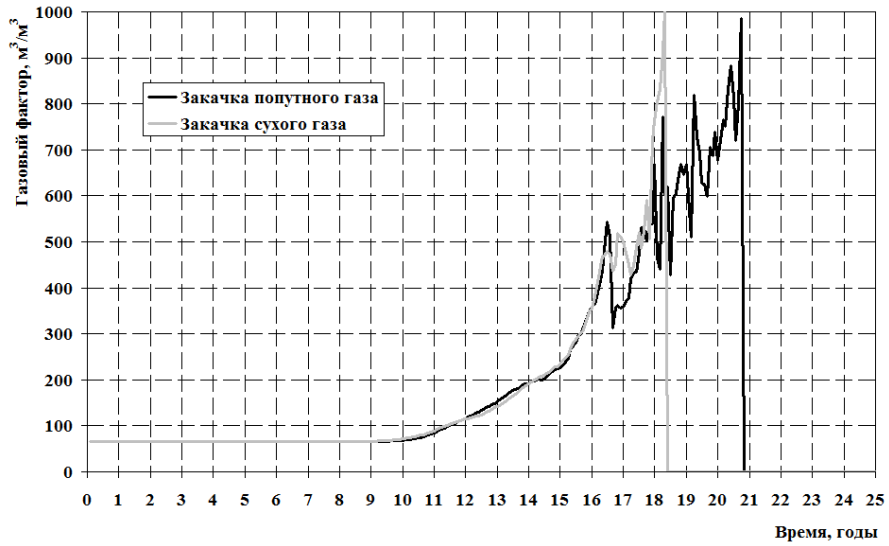


Рис. 6. Газовый фактор по группе скважин

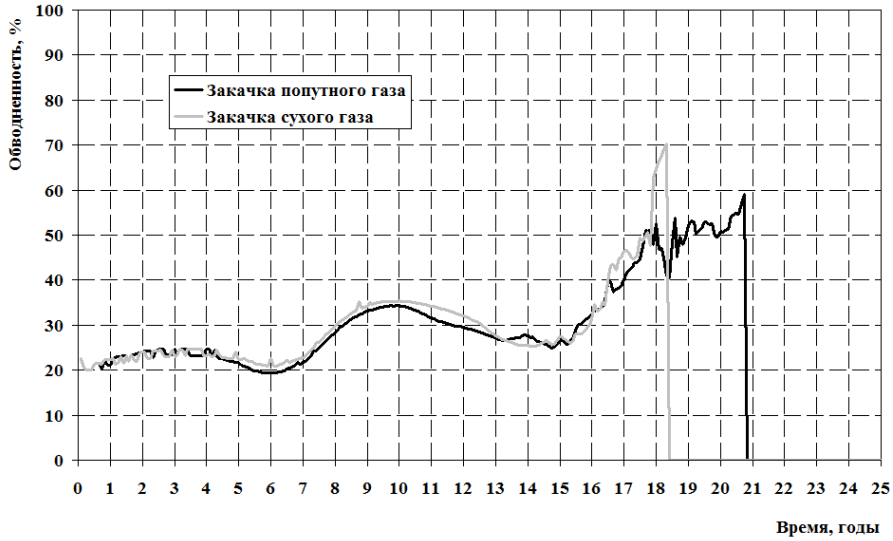


Рис. 7. Обводненность по группе скважин

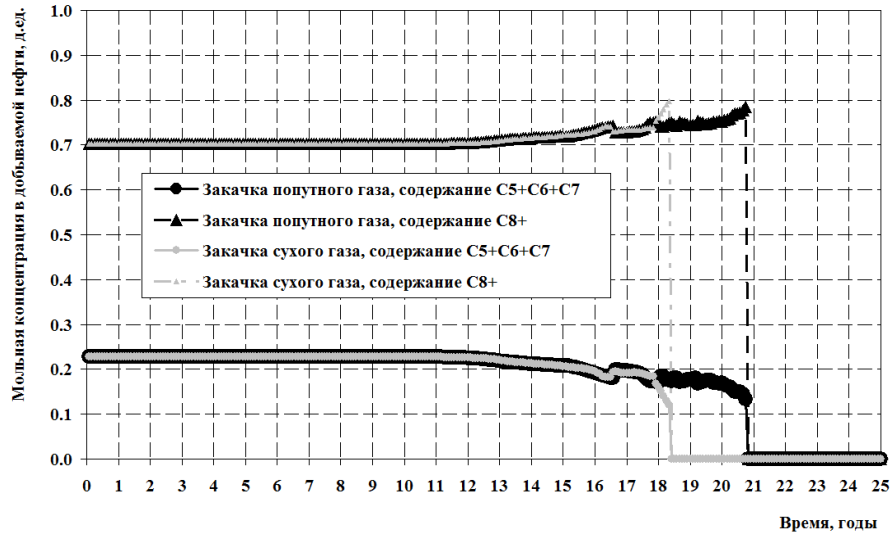


Рис. 8. Мольная концентрация некоторых компонентов в добываемой нефти по группе скважин

Successive substitution method for component concentration calculation during compositional modeling of hydrocarbon field development

I.V. Afanaskin, S.G. Volpin, A.V. Korolev

Abstract. A system of conservation equations for multicomponent flow modeling and a generalized equation of state are considered. An algorithm for determining the concentration of components in phases is described in detail using the method of successive substitution. An example of the implementation of this scheme for simulating alternate injection of gas and water in the development of an oil field is considered. An indirect confirmation of the model correctness is obtained.

Key words: compositional modeling, multicomponent flow, phase transitions.

Литература

1. K.H.Coats. An Equation of State Compositional Model. // Soc. Pet. Eng. J. (Oct. 1980), 363-376.
2. L.X.Ngiem, D.K.Fong and K.Aziz K. Compositional Modeling with an Equation of State // Soc. Pet. Eng. J. (Dec. 1981), 687-698.
3. K.H.Coats, L.K.Thomas and P.G.Pierson. Compositional and Black Oil Reservoir Simulation // SPE Reservoir Evaluation & Engineering (August 1998), 372-379.
4. H.Kazemi, C.R.Vestal and G.D.Shank. An Efficient Multicomponent Numerical Simulator. // Soc. Pet. Eng. J. (Oct. 1978), 355-368.
5. И.В.Афанаскин, А.В.Королев, С.Г.Вольпин и др. Модель многофазной многокомпонентной фильтрации для проектирования разработки месторождений углеводородов // Нефтепромысловое дело, № 2, 2017. – с. 5-12.
6. Г.Р.Гуревич, А.И.Брусиловский. Справочное пособие по расчету фазового состояния и свойств газоконденсатных смесей. М.: Недра, 1984. – 264 с.
7. С.Н.Закиров, Б.Е.Сомов, В.Я.Гордон и др. Многомерная и многокомпонентная фильтрация. М.: Недра, 1988. – 335 с.
8. М.Н.Муркес. Расчет фазовых равновесий при численном решении задач фильтрации многокомпонентных смесей в процессе разработки углеводородных залежей // Сб. тр. ВНИИ № 91, 1985. с. 32-37.
9. РД 39-1-579-81 Руководство по расчету фазовых превращений газоконденсатных вонефтегазоконденсатных систем и свойств фаз на ЭВМ.
10. M.L.Michelsen. The isothermal flash problem. Part I. Stability Fluid Phase Equilibria. Vol 9, P. 1-19, 1982.
11. Программа для моделирования процессов разработки нефте-газовых месторождений tNavigator версия 3.0. Техническое руководство. М.: Rock Flow Dynamics, 2009. – 782 с.

Развитие методики оценки фазового состояния парафинов в пластовых нефтях

К.Д.Ашмян [1], А.К.Пономарев[2], О.В.Ковалёва [3]

[1,2] ФГУ ФНЦ НИИСИ РАН и [3] АО «ВНИИнефть, Москва. Россия,

E-mail's: [1]kdashmyan@yandex.ru, [2]akponomarev@mail.ru, [3]olgakovaleva57@ mail.ru

Аннотация: В процессе разработки нефтяных месторождений выпадение «парафина» (асфальто-смоло-парафиновые отложения) может происходить как в пластах, так и скважинах. Для предотвращения образования отложений парафина или борьбы с этими отложениями необходимо знать условия, при которых происходит образование твердой фазы «АСПО», т.е. необходимо определить температуру насыщения нефти парафином при снижении пластового давления.

Ключевые слова: нефть пластовая, фазовое состояние, температура насыщения нефти парафином, пластовое давление, давление насыщения, газосодержание.

Введение

При разработке нефтяных залежей часто возникают технические осложнения, связанные с нарушением фазового состояния пластовых флюидов – нефти, свободного газа и пластовой воды. Фазовое состояние пластовых флюидов изменяется в результате воздействия новых термобарических условий залегания, при этом в нефтенасыщенном пласте происходит преобразование пластовой нефти из однофазной системы в двух или трехфазную.

Как правило, появление газовой фазы (свободного газа) происходит за счет выделения растворенного в нефти газа при снижении пластового давления ниже давления насыщения (Рпл. < Рнас.), что приводит к образованию в пласте двухфазной системы: жидкая нефтяная фаза + свободный газ.

Образование в пластовых условиях в нефти твердой фазы, как правило, происходит при изменении пластовой температуры и пластового давления. Твердая фаза, выделившаяся из нефти, состоит из высокомолекулярных парафиновых углеводородов с адсорбированными на них асфальтеновыми и смолистыми веществами, которые входят в состав пластовой нефти, а также частиц механических примесей и пластовой воды.

Выделившиеся из нефти в виде кристаллов тяжелые углеводороды парафинового ряда (C_{16+}) могут находиться в пластовой нефти как отдельная фаза и даже участвовать в процессе фильтрации к добывающей скважине или же, объединяясь между собой, образуют частицы большого размера, которые уже не могут пройти через поровое пространство породы и осаждаются на породе.

Это снижает фильтрационные свойства породы и приводит к уменьшению добычи, вплоть до полного прекращения. Причем, в пластовых условиях этот процесс необратимый.

В нефтяной промышленности принято техническое название выпавшего конгломерата - «парафин».

Для предотвращения выпадения «парафина» или же для борьбы с уже выпавшими отложениями (АСПО) необходимо знать условия, при которых образуется твердая фаза, ее состав и выявить зависимости этих характеризующих твердую фазу параметров.

Такие исследования достаточно сложны и трудоемки, а, следовательно, длительны и дорогостоящи, поэтому была предпринята успешная попытка проанализировать все имеющиеся собственные исследования и построить графоаналитическую зависимость, позволяющую проводить предварительные оценки фазового состояния пластовой нефти в области жидкость – твердая фаза [1].

Исходными экспериментальными данными являлись собственные исследования высокопарафинистых нефтей с содержанием парафина 20 ÷ 30% мольных (Харьяга, Узень, Белый Тигр и др.).

Полученная корреляционная кривая хорошо описывает зависимость

температуры насыщения нефти парафином от количества растворенных в нефти высокомолекулярных (парафиновых) компонентов нефти [1].

Дальнейшее развитие нефтяной отрасли, изменение качества сырьевой базы, а также освоение новых месторождений Восточной Сибири показали, что вопрос выпадения твердой фазы из пластовых нефтей имеет значительно более широкое значение, чем только для высокопарафинистых, для которых превалирующим условием при выпадении парафина является снижение пластовой температуры. Применение существующей методики [1] для малопарафинистых месторождений необоснованно, т.к. при определении корреляционной зависимости температуры насыщения нефти от содержания «парафина» не участвовали данные по малопарафинистым нефтям.

Исследования процесса выпадения парафина в малопарафинистых нефтях задача новая и не имеет еще пока достаточного объема данных для построения корреляционной зависимости. Однако, используя опыт построения зависимостей, характеризующих влияние различных факторов на образование твердой фазы в пластовой нефти для парафинистых и высокопарафинистых нефтей, возможно при уже имеющемся объеме экспериментальных данных получить эти зависимости и для малопарафинистых нефтей

В данной статье представлены результаты проведенных авторами экспериментальных и аналитических расчетных исследований малопарафинистых нефтей Восточной Сибири на примере месторождения Энское.

Определение температуры насыщения нефти парафином в малопарафинистых нефтях

Известно, что образование твердой фазы в пластовой нефти может происходить вследствие не только понижения пластовой температуры, но и изменения давления Общий вид зависимости температуры насыщения пластовой нефти парафином от давления представлен на Рисунке 1 [2] .

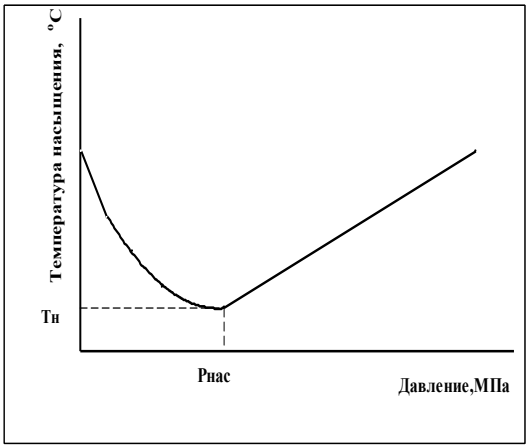


Рис. 1. Общий вид зависимости температуры насыщения пластовой нефти парафином от давления

Как видно из Рис. 1, повышение давления приводит к увеличению температуры насыщения нефти парафином.

С другой стороны, если понижать давление ниже давления насыщения $\mathbf{P_{нас}}$, то вследствие выделения из нефти растворенного в ней газа температура насыщения тоже увеличивается.

Поэтому, если нефть в пластовых условиях насыщена или близка к насыщению парафином, то при увеличении давления или снижении его ниже давления насыщения температура насыщения нефти парафином окажется выше температуры пласта, и парафин будет выпадать из нефти.

То же самое произойдет при понижении температуры пласта ниже температуры насыщения нефти парафином.

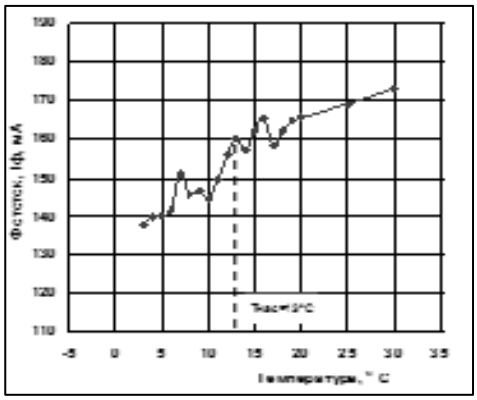
Рассмотрим влияние давления на температуру насыщения нефти парафином на примере исследования проб нефти из пласта $ВЧ_2$ блока I Энского месторождения.

В экспериментах температура насыщения нефти парафином определялась как в поверхностной пробе нефти, так и в нефти после дегазирования глубинных проб. Результаты представлены на Рис. 2.

Следует отметить, что температура насыщения нефти парафином для дегазированной и поверхностной проб т нефти отличаются на 3^oC.

Температура насыщения нефти парафином по дегазированной пробе составила 13 ^oC, а по поверхностной пробе экспериментально установлена 10 ^oC.

а)



б)

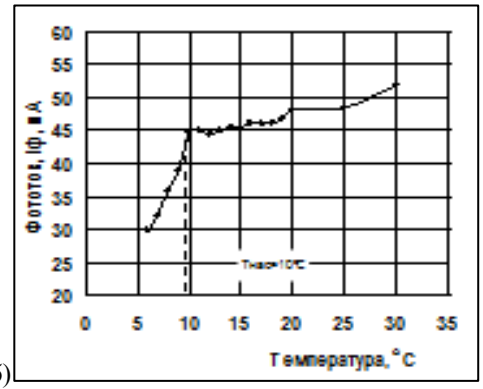


Рис. 2. Температура насыщения нефти парафином при атмосферном давлении (фотометрический метод): а) дегазированной, б) поверхностной

Температуру насыщения нефти парафином определяли фотометрическим методом на приборе ПТП. Исследования проводили в диапазоне температур от (+40°С) до (+2°С). Было установлено, что при давлении 16,1МПа температура насыщения пластовой нефти парафином составляет 9 °С. Эта же пластовая нефть была исследована при давлениях 12, 4, 0,1 МПа для установления зависимости температуры насыщения нефти парафином, как в однофазной области, так и при давлениях ниже давления насыщения нефти газом, т.е. в двухфазной системе жидкая фаза (нефть) + газовая фаза (выделившийся газ).

Зависимость температуры насыщения нефти парафином от давления представлена на Рис. 3.

Анализ результатов исследования глубинной пробы нефти от давления показал незначительное снижение температуры насыщения нефти парафином при снижении давления от пластового до давления насыщения, дальнейшее снижение пластового давления приводит к газоотделению и увеличению температуры насыщения нефти парафином.

Давление, кгс/см²	Температура насыщения нефти парафином,°С	Температурный диапазон исследования,°С
161($P_{пл}$)	8	50 – (+5)
140	7	40 – (+3)
120	7	40 – (+3)
96,8(P_s)	6 (экстраполяция)	–
60	8 (расчет)	–
40	9(расчет)	–
0	13	30 – (+3)

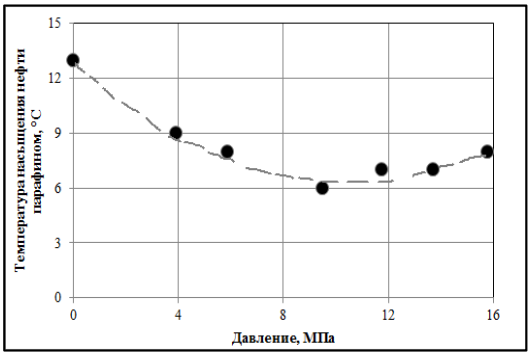


Рис. 3. Зависимость температуры насыщения пластовой нефти парафином от давления.

Ранее были исследованы глубинные пробы нефти Энского месторождения с целью изучения выпадения парафина из скважины 37, пласт $ВЧ_1$+$ВЧ_2$ и скважины 87, пласт $ВЧ_1$+$ВЧ_2$.

Для сравнения результаты этих исследований, а также результаты исследования глубинных проб из скважины 1028 представлены в Таблице 1.

Таблица 1. Зависимость температуры насыщения пластовой нефти парафином от температуры пласта. Месторождение Энское, блок I, пласт $ВЧ_2$, блок II, пласт $ВЧ_1$+$ВЧ_2$ и блок V, пласт $ВЧ_1$+$ВЧ_2$

№№ скв.	1028	37	87
Блок	I	II	V
Пласт	$ВЧ_2$	$ВЧ_1$+ $ВЧ_2$	$ВЧ_1$+ $ВЧ_2$
Пластовое давление,МПа	15,8	15,07	15,3
Пластовая температура,°С	20	20	13
Давление насыщения,МПа	9,5	12,9	10,8
Содержание парафина, % масс.	**1,84**	**2,23**	**1,68**
Температура насыщения нефти парафином, °С	**8**	**12**	**10**

Как видно из Таблицы 1, при различной температуре пластов значения температуры насыщения нефти парафином близки, потому что содержание парафина в пробе из скважины 1028 и скважины 87 примерно одинаковое. Увеличение температуры насыщения нефти парафином связано с увеличением содержания парафина в нефти (1,68 масс % в нефти из скв. 87; 2,23 масс % в нефти из скв.37) и различным составом пластовой нефти.

Кроме определения температуры насыщения нефти парафином фотометрическим методом, использовали также визуальный метод определения [2]. Эти методы позволяют при понижении температуры (начиная от пластовой) в пробе исследуемой нефти определить температуру насыщения нефти парафином, т.е. **температуру начала массовой кристаллизации содержащихся в нефти высокомолекулярных парафиновых углеводородов**. Появление кристаллов парафина в нефти можно наблюдать с помощью специально оборудованного микроскопа. Кристаллы парафина хорошо видны в проходящем поляризованном и обыкновенном свете, что позволяет не только наблюдать их появление, но и измерять линейные размеры кристаллов, а также производить микрофотографирование и изучать кинетику процесса кристаллизации.

Общий вид установки для визуального наблюдения представлен на Рисунке 4.

Рис. 4. Общий вид установки для визуального наблюдения

Для наблюдения за кристаллизацией парафина исследуемую нефть помещают в камеру высокого давления (КВДМ) под микроскоп. Устройство камеры позволяет производить эти наблюдения при различных давлениях и температуре, в том числе моделировать пластовые условия.

Исследования под микроскопом были проведены с дегазированной нефтью. Из фотоснимков нефти под микроскопом (Рисунок 5) видно, что при постоянном давлении и снижении температуры от (+25°C) образование кристаллов происходит при (+10°C). Количество кристаллов парафина и их размеры увеличиваются по мере снижения температуры до (+5°C).

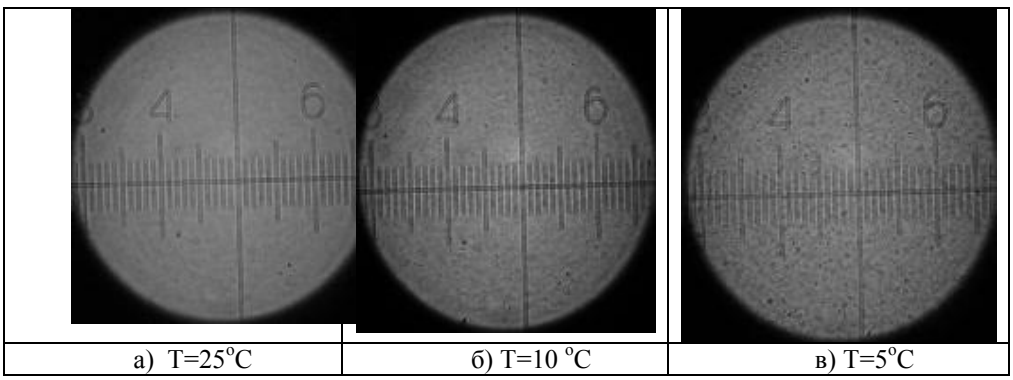
При температуре(+25°C) в поле зрения наблюдаются единичные частицы твердого парафина, размер которых колеблется в диапазоне от 0,0025 мм до 0,005 мм, с преобладанием частиц размером 0,005 мм.

При температуре (+10°C) твердых частиц парафина выпало почти вдвое больше и в поле зрения множественные частицы парафина с нечетко сформированной структурой по всему полю образца. Диапазон размеров выпавших парафинов колеблется от 0,0025 до 0,01 мм с преобладанием частиц, размером 0,005 мм.

Заключение

Рассмотрен вопрос влияния снижения пластового давления в месторождениях с малым содержанием «парафинов». Проведены экспериментальные исследования (совместно с АО «ВНИИнефть») влияния давления для малопарафинистых месторождений. Определены параметры выпадения парафина для таких нефтей (Таблица 2). Полученные зависимости будут использованы при создании методики «Определения фазового состояния (жидкость – твердая фаза) пластовых флюидов при добыче нефти».

Работа выполнена при поддержке гранта РФФИ 18-07-00679А.



а) T=25°C	б) T=10 ^{o}C	в) T=5°C

Рис. 5. Фотографии процесса выпадения парафина из поверхностной нефти при снижении температуры от 25 ^{o}C до 5 ^{o}C. (Месторождение Энское. Скважина 1028, блок I, пласт $ВЧ_2$.)

Таблица 2. Основные результаты исследования пластовых и дегазированных проб Верхнечонского месторождения, пласты $ВЧ_1$+$ВЧ_2$.

Параметры	Значения			
	2005 г	2006 г	2010 г	2011г.
№ скв.	**57**	**12**	**87*)**	**1028**
Дата отбора	13.04.05	04.07.06	24.03.10	07.03.11
Пластовое давление, МПа	15,73	16,06	15,3	15,79
Пластовая температура, °С	10,6	12,1	13	20
Давление насыщения, МПа	15,44	15,41	10,82	-
Газосодержание (стандартная сепарация), $м^3/м^3$	107,08	104,18	74,07	64,58
Плотность пластовой нефти, $кг/м^3$	785	776	798	801
Плотность сепарированной нефти при 20°С, $кг/м^3$	859,9	858,9	855,8	856,7
Температура насыщения нефти парафином, °С	**12**	**12**	**10**	**8**
Исследования поверхностных проб нефти				
Содержание, масс.доли, %				
воды	отс.	0,77	отс.	отс
серы	0,46	0,45	0,69	0,7
асфальтенов	0,2	0,28	0,11	0,03
смол	8,3	8,68	7,6	6,09
парафина	**1,5**	**1,71**	**1,68**	**1,84**
Температура, °С				
плавления парафина	53	51	55,5	57
насыщения нефти парафином	**8**	**8**	**8**	**10**
застывания нефти	(-53)	(-43)	(< -26,5)	(-63)
*) – определено по поверхностной пробе.				

Development of paraffin phase state in reservoir oil evaluation method

K.D.Ashmyan, A.K.Ponomarev, O.V.Kovaleva

Abstract. During the oilfield development “paraffin” (asphaltene-resin-paraffin deposits) deposition may occur in reservoirs and wellsites. To prevent the formations paraffin deposits or control of these deposits it’s necessary to know the conditions under which the formation of the solid phase of asphaltene-resin-paraffin deposits occurs. It means that it’s necessary to determine the saturation temperature of oil with paraffin while lowering the reservoir pressure.

Keywords. Reservoir oil, phase state, paraffin oil saturation temperature, reservoir pressure, saturation pressure, gas content.

Литература

1. РД 39-0147035-226-88 «Методическое руководство по выявлению залежей нефтей, насыщенных парафином»/ Руководитель разработки К.Д. Ашмян.

2. Экспериментальные методы исследования парафинистых нефтей/ Под ред. К.Д. Ашмяна. М.: ОАО «ВНИИнефть», 2004. – 108 с.

Методы стереовизуализации результатов моделирования неустойчивого вытеснения нефти из пористых сред

М.В. Михайлюк[1], А.В. Мальцев[2], П.Ю. Тимохин[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]mix@niisi.ras.ru, [2]avmaltcev@mail.ru, [3]webpismo@yahoo.de

Аннотация: В работе предлагаются методы и алгоритмы стереовизуализации данных, полученных в результате пошагового моделирования процесса неустойчивого вытеснения нефти из пористых сред. Задача рассматривается на примере рендеринга изменяющейся от шага к шагу трехмерной виртуальной модели изоповерхности насыщенности вытесняющей жидкости. Описываются подходы для реализации нескольких различных типов стерео. Также рассматривается режим многопортовой визуализации данной виртуальной модели.

Ключевые слова: визуализация, стерео, многопортовый режим, нефть, пористая среда, моделирование.

Введение

Как известно, человеческое зрение является бинокулярным. Это позволяет человеку оценивать расстояния до объектов и воспринимать их правильное взаиморасположение. Имитация бинокулярного зрения в системах виртуального окружения производится с помощью технологии стереовизуализации. Изучение и анализ результатов научного моделирования на стереоизображении обеспечивает исследователю более глубокое понимание структур, образующихся в ходе проведения экспериментов, а также происходящих при этом процессов.

В данной работе рассматриваются методы визуализации в стереорежиме результатов компьютерного моделирования неустойчивого вытеснения нефти из пористых сред на примере изоповерхности насыщенности вытесняющей жидкости [1-4].

1. Стереовизуализация

Технология стеровизуализации включает много различных методов моделирования бинокулярного зрения. В данной работе рассматривается ряд эффективных методов, суть которых состоит в создании стереопары – пары изображений виртуальной сцены для левого и правого глаза. При направлении этих изображений в соответствующие глаза наблюдателя, он начинает ощущать эффект объема визуализируемой среды и воспринимать глубину виртуального пространства. Рассматриваемые методы создания стерео включают несколько этапов: установка двух виртуальных камер с нужными параметрами, получение с их помощью стереопары и ее отображение в глаза наблюдателя. Опишем эти этапы подробнее.

1.1. Установка виртуальных камер

Рассмотрим трехмерную сцену, в которой представлены результаты некоторой последовательности шагов моделирования неустойчивого вытеснения нефти из пористых сред [1]. Точка P_E расположения виртуального наблюдателя в сцене и единичный вектор $\boldsymbol{V}$ направления его взгляда (Рис. 1) выбираются так, чтобы сцена целиком попадала в поле зрения. Наблюдатель имеет собственную локальную систему координат (СК) VCS с центром в точке P_E и осью Z, направленной противоположно $\boldsymbol{V}$. Ось X направлена вправо, ось Y – вверх.

Виртуальные камеры для реализации стерео устанавливаются в точках P_L и P_R на оси X системы VCS, которые расположены симметрично (слева и справа соответственно) относительно ее центра P_E. Длина отрезков P_EP_L и P_EP_R выбирается равной 0.032 м – половине среднего расстояния между глазами среднестатистического человека. Направления

взглядов $\boldsymbol{V}_L$ и $\boldsymbol{V}_R$ обеих камер параллельны и совпадают с $\boldsymbol{V}$. Горизонтальные растворы камер (*field of view*, *FOV*) устанавливаются равными 54 градуса (Рис. 1).

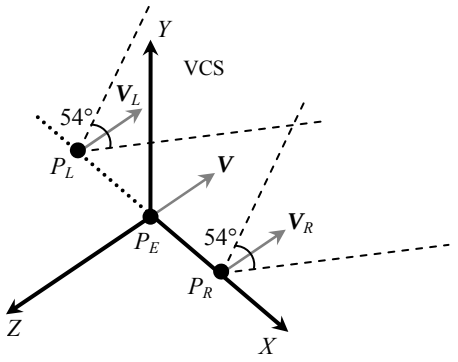


Рис. 1. Установка виртуальных камер для стереовизуализации

1.2. Синтез и отображение стереопары

Для получения стереопары производится визуализация виртуальной сцены из каждой установленной ранее камеры в текущий буфер кадра. Формат записи и техника отображения результатов рендеринга зависят от выбранного метода стеровизуализации. В данной работе рассматриваются анаглифический и поляризационный методы получения стерео.

Анаглифический метод. Этот метод получения стереоизображения основан на цветовом разделении двух изображений стереопары. Для левого глаза виртуальная сцена визуализируется из левой камеры в оттенках красного цвета, а для правого – из правой камеры в оттенках синего и зеленого цветов. Обе картинки, получившиеся в результате рендеринга, передаются на одно и то же устройство отображения (монитор, проектор). Поскольку виртуальные камеры для левого и правого глаза, установленные в точках P_L и P_R, разнесены по оси X системы VCS, то два изображения из стереопары (красное и сине-зеленое) также будут иметь некоторый горизонтальный параллакс на экране (Рис. 2). Чтобы просматривать данный тип стерео, исследователь должен использовать специальные очки с установленными в них фильтрами, которые пропускают свет только в определенном диапазоне длин волн: для левого глаза – красный фильтр, для правого – бирюзовый (сине-зеленый). Для рендеринга стереопары при этом будем использовать следующий

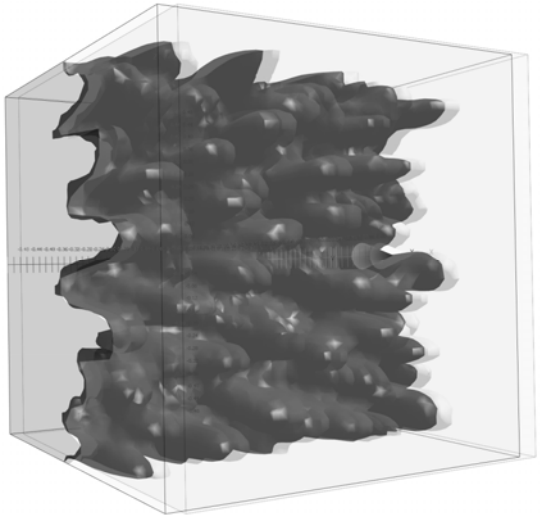


Рис. 2. Анаглифическое стереоизображение

Алгоритм синтеза анаглифического стереоизображения

1. С помощью функции *glViewport* (x, y, w, h) графической библиотеки OpenGL задаем координаты x, y левого нижнего угла, а также ширину w и высоту h (в пикселах) той области окна приложения, в которую будет осуществляться вывод изображений стереопары;

2. Устанавливаем использование камерами перспективного проецирования с горизонтальным углом зрения FOV = 54°, используя функцию *gluPerspective* (54.0·h /w, w /h, n_p, f_p), где n_p и f_p – расстояния до ближней и дальней отсекающих плоскостей соответственно;

3. Блокируем вывод в буфер кадра всех цветовых компонент, кроме красной. Это можно сделать функцией

glColorMask (1, 0, 0, 0);

4. Визуализируем сцену из левой камеры;

5. Очищаем буфер глубины с помощью функции *glClear* (*GL_DEPTH_BUFFER_BIT*);

6. Функцией *glColorMask* (0, 1, 1, 0) блокируем вывод в буфер кадра всех цветовых компонент, кроме синей и зеленой;

7. Визуализируем сцену из правой камеры;

8. Снимаем блокировку вывода цвета со всех каналов, используя функцию

glColorMask (1, 1, 1, 1).

Поляризационный метод. Этот метод создания стерео использует два идентичных проектора L и R, выводящих на один и тот же экран изображения соответственно для левого и правого глаз. На линзу каждого из них устанавливается светофильтр, преобразующий световой поток в плоскополяризованный. Фильтры закрепляются таким образом, чтобы их плоскости поляризации были перпендикулярны друг другу (например, для проектора L – вертикальная, для проектора R – горизонтальная). Экран вывода отражает падающие на него световые потоки, сохраняя исходные плоскости поляризации. Чтобы

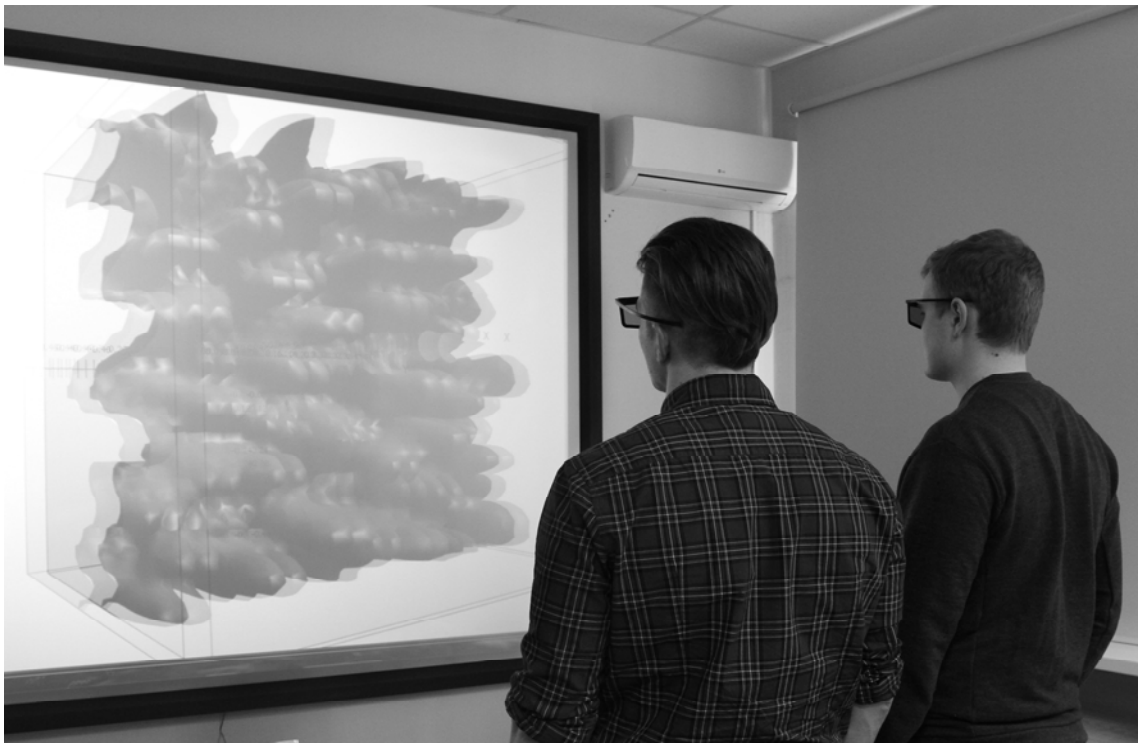


Рис. 3. Поляризационное стереоизображение

воспринимать стереоэффект, пользователю необходимо использовать поляризационные стереоочки (Рис. 3), светофильтры которых также имеют взаимно перпендикулярные плоскости поляризации (в нашем случае, вертикальную – для левого глаза, чтобы пропускать изображение с проектора *L*, и горизонтальную – для правого, чтобы пропускать изображение с проектора *R*). Пусть каждый из проекторов имеет разрешение *w* x *h* (в пикселах). Тогда для получения стереопары в данном случае будем применять следующий

Алгоритм синтеза поляризационного стереоизображения

1. С помощью настроек операционной системы устанавливаем расширенный рабочий стол размером 2*w* x *h*, левая половина которого отображается проектором для левого глаза, а правая половина – проектором для правого глаза;

2. Переводим общую область вывода окна приложения в полноэкранный режим;

3. Для каждой камеры устанавливаем перспективное проецирование с горизонтальным углом зрения FOV = 54°. Для этого используется оператор

$$gluPerspective\,(54.0 \cdot h/w, w/h, n_p, f_p);$$

4. Оператором *glViewport* (*0*, *0*, *w*, *h*) устанавливаем левую половину буфера кадра в качестве текущей области для рендеринга;

5. Визуализируем сцену из левой камеры;

6. Оператором *glViewport* (*w*, *0*, *w*, *h*) устанавливаем правую половину буфера кадра в качестве текущей области для рендеринга;

7. Визуализируем сцену из правой камеры.

2. Многопортовая визуализация

Многопортовая визуализация представляет собой рендеринг виртуальной сцены

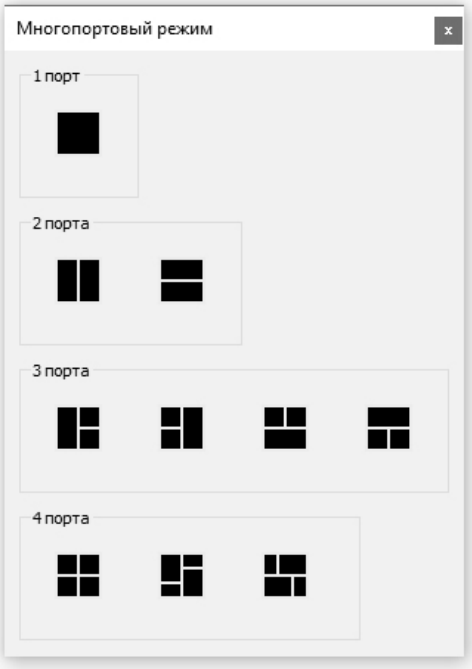


Рис. 4. Окно выбора конфигурации портов

одновременно с нескольких ракурсов, соответствующих расположению такого же числа виртуальных камер. При переходе в этот режим выдается окно с возможными конфигурациями портов вывода (Рис. 4). Пользователь с помощью мышки выбирает одну из них. После этого на экране отображаются изображения виртуальной модели изоповерхности насыщенности вытесняющей нефть жидкости во всех портах, предусмотренных выбранной конфигурацией.

Клик мышью по одному из имеющихся портов делает его активным: появляется возможность включения полноэкранного моно- или стереорежима отображения данного порта, а также возможность управления соответствующей виртуальной камерой.

С помощью клавиш можно перемещать точку P_E расположения виртуальной камеры порта влево/вправо, вверх/вниз, вперед/назад, а также по большой (горизонтальной или вертикальной) дуге сферы с центром в центральной точке O объема моделирования и радиусом R, равным расстоянию от O до P_E. Новое положение P_E' и направление взгляда камеры задаются с помощью функции

glLookAt (e_x, e_y, e_z, r_x, r_y, r_z, u_x, u_y, u_z)

графической библиотеки OpenGL, где e_x, e_y, e_z – координаты точки P_E', r_x, r_y, r_z – координаты точки, куда будет направлена камера, u_x, u_y, u_z – координаты вектора, указывающего верх камеры (все координаты задаются в текущей СК VCS). Например, сдвиг камеры вправо по оси X на Δx задается оператором

glLookAt(Δx, 0, 0, Δx, 0, -1, 0, 1, 0).

Рассмотрим перемещение камеры по большой окружности вдоль оси X. Из Рис. 5 видно, что

$e_x = R \sin\alpha$, $e_y = 0$, $e_z = -R \cos\alpha$,
$r_x = O_x$, $r_y = O_y$, $r_z = O_z$,
$u_x = 0$, $u_y = 1$, $u_z = 0$.

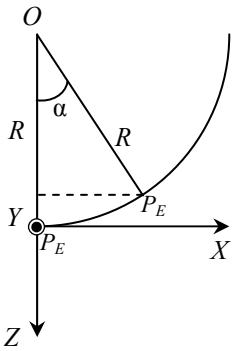


Рис. 5. Перемещение камеры по большой окружности

Аналогично выписываются формулы для движения камеры по большой окружности в направлении оси Y.

Заключение

В данном исследовании предложены методы и алгоритмы рендеринга в стереорежиме трехмерных виртуальных моделей изоповерхностей насыщенности жидкости, вытесняющей нефть из пористых пород. Предложенные решения обеспечивают визуализацию изоповерхностей на основе числовых данных, получаемых в результате пошагового компьютерного моделирования процесса вытеснения. Также предложены подходы к реализации многопортового режима для одновременного просмотра визуализируемых виртуальных моделей с различных ракурсов. На основе представленных методов и алгоритмов созданы программные модули для системы моделирования процесса неустойчивого вытеснения нефти из пористых сред.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-29-15099-офи_м.

The methods for 3D stereo visualization of data obtained in simulation of unstable oil displacement from porous media

M.V. Mikhaylyuk, A.V. Maltsev, P.Yu. Timokhin

Abstract: The paper proposes methods and algorithms for 3D stereo visualization of resulting data obtained in step-by-step simulating of the process of unstable oil displacement from porous media. As an example for rendering task a 3D-virtual model representing the isosurface of the saturation of displacing liquid is chosen. The approaches for implementation of several different types of stereo are described. A multiport mode of visualization of given virtual model is considered as well.

Keywords: visualization, stereo, multiport mode, oil, porous media, simulation.

Литература

1. N.N.Smirnov, J.C.Legros, V.F.Nikitin, E.Istasse, L.Schramm, F.Wassmuth, D.A.Hart. Filtration in artificial porous media and natural sands under microgravity conditions (2003). Microgravity Science and Technology, 14 (2), pp. 3-28.
2. N.N.Smirnov, V.F.Nikitin, O.E.Ivashnyov, A.Maximenko, M.Thiercelin, A.Vedernikov, B.Scheid, J.C.Legros. Microgravity investigations of instability and mixing flux in frontal displacement of fluids (2004) Microgravity Science and Technology, 15 (2), pp. 35-51.
3. М. В. Михайлюк, П. Ю. Тимохин, А. В. Мальцев, В. Ф. Никитин, Е. И. Скрылева, В. В. Тюренкова. Моделирование и визуализация процесса вытеснения нефти из пористой среды // Вестник кибернетики. – 2016. – № 3. – С. 35-41.
4. И.В. Козлов, Е.И. Скрылева. Математическое моделирование и обработка эксперимента по вытеснению нефти водой из неокомских песчаников // Вестник кибернетики. 2016. № 2. С. 138-145.

О множестве вида A/A для плотных подмножеств интервала [1, n]

Ю.В. Кузнецов[1], М.М. Петрунин[2], Ю.Н. Штейников[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1,2] yukuz61@rambler.ru, [3] yuriisht@gmail.com,

Аннотация: Изучается плотность множеств A/A для подмножеств интервала [1,n] и вычисляются некоторые его характеристики.

Ключевые слова: плотность, функция Эйлера.

Введение

Напомним вопрос, который был представлен в работе Х.Силлеруело, Д.С. Рамана и О. Рамаре [1]. Ниже дается его формулировка.

Вопрос.

Пусть $A \subseteq [1,n]$ и $|A/A| \sim 6n^2/\pi^2$ при n стремящимся к бесконечности. Верно ли, что имеет место соотношение $A \sim n$?

Утверждение 1. Сформулированный Вопрос имеет положительный ответ.

Мы приводим доказательство Утверждения 1 и даём явные оценки на функцию в неравенстве для плотности множества A/A.

Доказательство Утверждения 1.

Приведем сначала краткую идею доказательства дополнительного утверждения, которое эквивалентно положительному ответу на поставленный Вопрос.

Утверждение 2.

Для каждого $\alpha > 0$ имеется константа $c(\alpha) > 0$ такая, что если

$A \subseteq [1,n]$ и $|A| < (1-\alpha)n,$

то неравенство

$|A/A| < 6(1-c(\alpha))n^2/\pi^2$

справедливо для всех достаточно больших n.

Доказательство.

Определим $A_{0:=}[1,n]$, и пусть $a_1,..,a_s$ - все элементы (занумерованные в некотором порядке) интервала [1,n], которые не принадлежат A, и, тем самым, $A=A_0 \setminus \{a_1,...,a_s\}$.

Пусть, далее, мы имеем множества A_t для t=0,… , i-1. Пусть, наконец, i < s, так что определены множества $A_i=A_{i-1} \setminus \{a_i\}$.

Для произвольного элемента a_i тогда справедливо:

1) если

$a_i \le n/2$ и $n/2 \le b \le n$,

причём $\gcd(a_i,b)=1$, то a_i/b не принадлежит A_i/A_i и значит a_i/b не принадлежит A/A;

2) если

$a_i > n/2$ и $1 \le b \le n$,

причём $\gcd(a_i,b)=1$, то a_i/b не принадлежит A_i/A_i и значит a_i/b не принадлежит A/A.

Отсюда вытекает что для любого a_i существует не менее

1) $[n/2a_i]\,\varphi(a_i)$, при $a_i \le n/2$;

2) $\varphi(a_i)$, при $a_i > n/2$

различных b , которые образуют несократимые дроби вида a_i/b, принадлежащие A_0/A_0, но не принадлежащие A/A.

Заметим, что различным a_i соответствуют непересекающиеся множества таких дробей.

Таким образом, далее достаточно убедиться в справедливости следующего утверждения:

для любого $\alpha > 0$ существует константа $c(\alpha) > 0$ такая, что для произвольного множества

$A' \subseteq [1,n]$ с $|A'| = \alpha n$,

имеет место неравенство

$$\sum_{a \in A', a \le n/2} [n/2a]\varphi(a) + \sum_{a \in A', a > n/2} \varphi(a) \ge c(\alpha)n^2.$$

Но данное утверждение было доказано в работе [2].

Этим мы завершаем доказательство соответствующего Утверждения 2..

Численные расчеты.

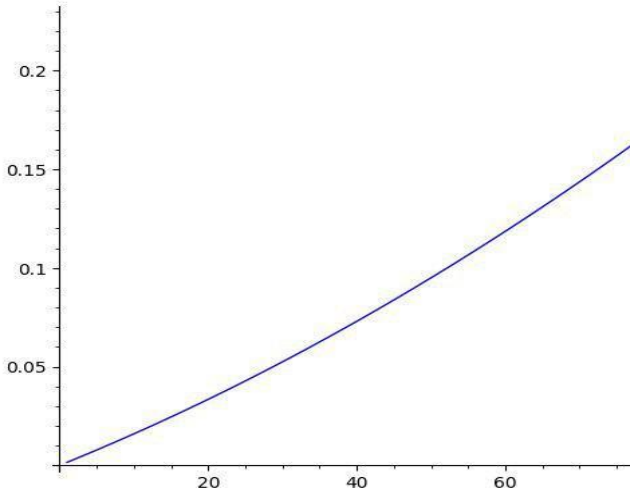


Рис. 1. График функции f(i).

Нами были посчитаны значения специальных нормированных сумм функции Эйлера на сплошном интервале от 1 до 100. Компьютерные вычисления осуществлялись на аппаратном комплексе НИИСИ РАН при помощи системы компьютерной алгебры SageMath, построенной на базе языка Python и множества программных пакетов (NumPy, SciPy и других).

В результате был построен график зависимости нормированного значения соответствующей суммы от натурального числа N, то есть график функции

$$f(i) = S(i) / N^2, \quad 1 \le i \le 100,$$

где

$$S(i) = \sum_{[N/2] \le k \le [N/2]+[iN/200]} \varphi(k),$$

иллюстрирующий результаты настоящей статьи. Он приведен на Рисунке 1.

Публикация выполнена в рамках Государственного задания по проведению фундаментальных научных исследований по теме (проекту) «Исследование групповых алгебраических многообразий и их связей с алгеброй, геометрией и теории чисел» (0065-2018-0019).

On sets of the form A/A for a dense sets of the interval [1,n]

Yu.V. Kuznetsov, M.M.Petrunin, Yu.N.Shteinikov

Abstract: We study the density of the sets A/A for dense subsets of the interval [1,n].

Keywords: density, Euler function.

Литература

1. Х. Силлеруело, Д. С. Рамана, О. Рамаре. «Частные и произведения подмножеств нулевой плотности множества натуральных чисел». *Тр. МИАН, 2017, т. 296, с. 58–71.*
2. Ю.Н.Штейников. «Дополнение к работе Х. Силлеруело, Д.С. Раманы, О. Рамаре “Частные и произведения подмножеств нулевой плотности множества натуральных чисел”». *Тр. МИАН, 2017,* т. 296, *с. 260–264.*